Oversampling Tabular Data with Deep Generative Models: Is it worth the effort?

Ramiro D. Camino, Radu State
SnT, University of Luxembourg, Luxembourg
ramiro.camino,radu.state@uni.lu

Christian A. Hammerschmidt
Delft University of Technology, Netherlands
c.a.hammerschmidt@tudelft.nl

Abstract

In practice, machine learning experts are often confronted with imbalanced data. Without accounting for the imbalance, common classifiers perform poorly, and standard evaluation metrics mislead the practitioners on the model's performance. A standard method to treat imbalanced datasets is under- and oversampling. In this process, samples are removed from the majority class, or synthetic samples are added to the minority class. In this paper, we follow up on recent developments in deep learning. We take proposals of deep generative models and study these approaches' ability to provide realistic samples that improve performance on imbalanced classification tasks via oversampling. Across 160K+ experiments, we show that the improvements in terms of performance metric, while shown to be significant when ranking the methods like in the literature, often are minor in absolute terms, especially compared to the required effort. Furthermore, we notice that a large part of the improvement is due to undersampling, not oversampling.

1 Introduction

With recent advances in the field of generative adversarial networks (GANs) [13], such as learning to transfer properties [19], advances in practical aspects [14], or advances in understanding theoretical aspects [27], it is tempting to apply GANs as tools to problems in data science tasks. Data available in real-world settings often have quality issues, like class imbalances and the lack of ground truth. Generative methods learning the distribution of the data using deep learning, such as GANs and variational autoencoders (VAEs) [20], can help to build solutions. While deep generative models (DGM) work well on continuous-domain problems such as images and video, they struggle generating samples in the form of discrete sequences or mixed distributions. This situation occurs partly due to the inherent difficulty of training networks with discrete outputs: sampling from discrete distributions is a non-differentiable operation, making it impossible to train the network using backpropagation. To use DGMs for imputation, simulation, feature extraction, transfer learning, or sampling artificial data points, these limitations need to be addressed.

In the context of data science and business-oriented applications, data is often tabular, i.e., contains multiple categorical and numerical variables. Discrete variables are also crucial in natural language modeling problems and reinforcement learning tasks. While initial proposals focused on continuous variables, following proposals addressed generating multivariate binary samples [6] or sequential samples from a single categorical variable [21, 37, 18, 14, 2, 3].

Based on these methods, several papers addressed the issue of generating samples over distributions of discrete or tabular data, including generating airline data [30] or medical time series [7] as

1st I Can't Believe It's Not Better Workshop (ICBINB@NeurIPS 2020), Vancouver, Canada.

well as frameworks to solve tasks such as oversampling [9] or multiple dependent tasks such as HexaGAN [15]. Nevertheless, comparing the performance of different architectures on a single task is difficult: Besides using different datasets and multiple setups for hyperparameter search, the protocols to solve tasks, such as the under- and oversampling ratios, vary from proposal to proposal.

In the following sections, we present related work (Section 2), our protocol and comparision approach (Section 3), the experiments (Section 4) and our conclusions (Section 5).

2 Related Work

Unsupervised learning from datasets is of interest in many fields, serving different purposes. Examples include (Bayesian) inference tasks as in [25], where the authors learn Bayesian models using non-parametric priors over multivariate tabular data, as well as unsupervised feature discovery, extraction, and transfer [19, 34, 16], and synthetic data for privacy aware applications [6]. In the context of imbalanced datasets, SMOTE [4] is a well-known method for generating synthetic samples which combined with an over- and undersampling protocol, can improve classifier performance. Several variants of SMOTE were proposed [29], each addressing different shortcomings or extending the applicability of the method to other types of datasets and variable distributions.

On continuous data, GANs [13] have proven to be good at learning data distributions in an unsupervised fashion. By feeding in additional information (such as labels), conditional GANs [28] can learn to generate samples for specific inputs. On discrete data (e.g., text sequences, categorical data), GANs face problems leading to comparatively worse performance. Several approaches exist to deal with discrete data. The Gumbel-Softmax [17] and the Concrete-Distribution [24] were simultaneously proposed to tackle this problem in the domain of variational autoencoders (VAE) [20]. Later, [21] adapted the technique to GANs for discrete sequences.

Addressing the same problem, a reinforcement learning approach called SeqGAN [37] interprets the generator as a stochastic policy and performs gradient policy updates to avoid the problem of backpropagation with discrete sequences. The discriminator outputs the reinforcement learning reward for a full sequence, and several simulations generated with a Monte Carlo search are executed to complete the missing steps.

Adversarially Regularized Autoencoders (ARAE) [18] transform sequences from a discrete vocabulary into a continuous latent space while simultaneously training both a generator (to output samples from the same latent distribution) and a discriminator (to distinguish between real and fake latent codes). The approach relies on Wasserstein GAN (WGAN) [1] to improve training stability and obtain a loss more correlated with sample quality. The main two changes in WGAN consist of replacing the discriminator with a critic and limiting the critic parameters' size by a constant.

MedGAN [6], while architecturally similar, is used to synthesize realistic health care patient records. The method pre-trains an autoencoder, and then the generator returns latent codes as in the previous case, but they pass that output to the decoder before sending it to the discriminator; therefore, the discriminator receives either fake or real samples directly instead of latent codes. They propose using shortcut connections in the generator. Additionally, the authors present the mini-batch averaging technique to evaluate better the generation of a whole batch instead of individual isolated samples. Before feeding a batch into the discriminator, mini-batch averaging appends the batch's average value per dimension to the batch itself.

An improved version for WGAN is presented in [14] to address the difficulty of training GANs by adding a gradient penalty to the critic loss (WGAN-GP) and removing the size limitation of the critic parameters. The authors present a use case for the generation of word sequences, where they claim that during training, discrete samples can be generated just by sending the outputs of softmax layers from the generator to the discriminator without sampling from those outputs.

3 Comparison

3.1 Datasets

Datasets like MNIST, ImageNet, and CIFAR are very well known in the domain of computer vision. Thanks to this, countless studies could compare their findings against others using widely accepted

benchmarks. On the other side, when applying deep learning to tabular data, no framework is well defined. Several papers opted to work with datasets from the UCI Repository [10] on tasks related to this domain: tabular data imputation [36, 31, 12, 26], imbalanced classification using a latent space [32], oversampling from deep generative models [9, 35] or all the previous tasks at the same time [15]. Nevertheless, we want to point out several interesting aspects of the datasets selected across these studies. First of all, the datasets are usually presented with short names or aliases, leading to confusion. For example, one dataset referred to as "breast" or "breast-cancer" presents four versions online with different features, and sometimes it is not very clear which one the study is referring to. Second, some datasets contain less than a thousand samples or just a few features. While this can be reasonable for other machine learning models, deep learning models may not reach their full potential when training with datasets of such dimensions. Finally, most of the classification tasks associated with imbalanced UCI datasets can be easily solved with state-of-the-art machine learning algorithms like XGBoost [5] without much need of any additional treatment. For example, one study [32] showed that 14 datasets from the UCI Repository could be classified with $ROCAUC > 0.9$ (and for most of the cases very close to 1) with a simple Random Forest [23]. Before starting this study, we used XGBoost to classify the ten most used datasets from the UCI repository among the related work. Some of them are multi-class problems, so we transformed them into several one-vs-all binary classification problems and several one-vs-one binary classification problems by pairing two classes at a time. We found only two cases where the test f1 score was less than 0.95: "Adult"[1] and "Default of Credit Card Clients"[2].

3.2 Deep Generative Models

In this study, we compare two well known deep generative architectures: GAN [13] and VAE [20]. We include as well two variants of GAN that involve autoencoders, ARAE [18] and MedGAN [6], adapting the reconstruction loss by separating the features per variable (e.g., all the features from a one-hot-encoded categorical variable). Each separated reconstruction loss depends on the variable type: cross-entropy for categorical variables, binary cross-entropy for binary variables, and mean squared error for numerical variables. This approach is based on previous work using GAN only with categorical variables [2], and a more complex version can be found in HI-VAE [31]. We further include Multi-Variable [2, 3] (MV) versions of the previous models. We selected gumbel-softmax [17, 24] for the categorical activations. For all the Multi-Variable versions of GAN, we use WGAN [1] instead, and we also add an alternative with WGAN-GP [14].

3.3 Sampling from Deep Generative Models

Among the literature, we found two alternatives to generate synthetic samples with deep generative models for later use in imbalanced classification tasks. The authors in [11] train a GAN only using the minority class samples in order to improve the detection of credit card frauds. We call this technique "minority". The second method [9] uses a GAN with a condition (the label) as an additional input for the generator and discriminator. We will refer to this alternative as "conditional". We propose an additional technique that we call "rejection", that consists of training any deep generative model attaching the label to the rest of the features as an additional variable. Afterward, all the samples that do not belong to the desired class are discarded during the synthetic generation. For conditional and rejection, the discriminator received the label as an input. The difference is that when using the conditional strategy, the label is only a generator input, but when using the rejection strategy, the label is only a generator output. Additionally, iterative draws are necessary for obtaining a synthetic sample of a specific size with the rejection strategy, and the procedure could never end. Therefore, a limit on the number of draws or execution time is needed.

4 Experiments

In this section, we provide the details and the analysis of our experiments. The goal is to provide an empirical comparison of how different undersampling and oversampling techniques affect the

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

[2] https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients

classification of imbalanced datasets. The code for classification, undersampling, oversampling, and all the tasks related to deep generative models can be found online [3].

Table 1: Classification experiments for the "Credit card fraud" dataset.

Part I: Classifier				
IR			**Train f1**	**Test f1**
0.001			0.904 ± 0.005	0.814 ± 0.046
Part II: Undersampling → Classifier				
USR			**Train f1**	**Test f1**
0.007			0.861 ± 0.009	0.813 ± 0.057
Part III: Undersampling → SMOTE Oversampling → Classifier				
Oversampling	**USR**	**OSR**	**Train f1**	**Test f1**
BorderlineSMOTE	0.004	0.005	0.877 ± 0.010	0.811 ± 0.055
RandomOverSampler	0.002	0.007	0.895 ± 0.006	0.822 ± 0.052
SMOTE	0.002	0.007	0.891 ± 0.006	0.831 ± 0.043
SVMSMOTE	0.002	0.003	0.899 ± 0.006	0.816 ± 0.062
Part IV: Undersampling → DGM Oversampling → Classifier				
DGM	**Sampling**	**USR**	**OSR**	**Train f1**	**Test f1**
vae	Minority	0.002	0.009	0.905 ± 0.006	0.820 ± 0.039
arae	Minority	0.004	0.005	0.873 ± 0.014	0.814 ± 0.052
medgan	Minority	0.002	0.010	0.896 ± 0.006	0.822 ± 0.042
gan	Minority	0.006	0.008	0.860 ± 0.013	0.820 ± 0.069
vae	Conditional	0.006	0.009	0.871 ± 0.008	0.817 ± 0.062
arae	Conditional	0.002	0.007	0.902 ± 0.004	0.816 ± 0.048
medgan	Conditional	0.003	0.007	0.886 ± 0.007	0.810 ± 0.056
gan	Conditional	0.004	0.010	0.877 ± 0.011	0.815 ± 0.051
vae	Rejection	*Timeout*			
arae	Rejection	*Timeout*			
medgan	Rejection	*Timeout*			
gan	Rejection	*Timeout*			

4.1 Datasets

We select two datasets from the UCI Repository [10]: "Adult" and "Default of credit card clients". We also include the "Credit card fraud" dataset presented in [8], which we obtained from the Kaggle repositories[4]. This dataset only contains numerical features (most of them from coming from a PCA transformation) and is highly imbalanced ($< 0.001\%$ of the cases are labeled as frauds). In all three cases, we apply the same preprocessing procedure. We use one-hot-encoding for all the categorical variables, but we represent binary variables with one binary feature. Additionally, all the numerical variables are scaled to fit inside the range $[0; 1]$. We generate metadata indicating for each variable the type (categorical, binary, or numerical) and the size (number of features). This information is used by the SMOTE-NC method, the autoencoder reconstruction loss, and the multi-variable architectures. The code for the preprocessing is available online [5].

4.2 Classification Protocol

All of our experiments involve binary classification tasks implemented with XGBoost [5]. We compute the mean and standard deviation of the f1 score for the train and test sets over ten equally sized and disjoint folds. For each dataset, we further separate 25% of the train set as a validation set, and we run a grid search over several XGBoost hyperparameters (e.g., the max depth and number of estimators). In the Part I of Tables 1, 2 and 3 we indicate the IR for each dataset along with the best results. The selected XGBoost hyperparameters are fixed for the experiments in the following parts.

[3] https://github.com/rcamino/deep-generative-models

[4] https://www.kaggle.com/mlg-ulb/creditcardfraud

[5] https://github.com/rcamino/dataset-pre-processing

Table 2: Classification experiments for the "Adult" dataset.

Part I: Classifier				
IR			**Train f1**	**Test f1**
0.33			0.743 ± 0.002	0.716 ± 0.005
Part II: Undersampling → Classifier				
USR			**Train f1**	**Test f1**
0.50			0.756 ± 0.002	0.731 ± 0.010
Part III: Undersampling → SMOTE Oversampling → Classifier				
Oversampling	**USR**	**OSR**	**Train f1**	**Test f1**
ADASYN	0.4	0.6	0.749 ± 0.001	0.727 ± 0.010
BorderlineSMOTE	0.5	0.6	0.751 ± 0.001	0.730 ± 0.008
KMeansSMOTE	0.5	0.6	0.752 ± 0.002	0.731 ± 0.009
RandomOverSampler	0.5	0.6	0.756 ± 0.002	0.732 ± 0.007
SMOTE	0.5	0.6	0.752 ± 0.002	0.732 ± 0.008
SMOTENC	0.6	0.7	0.662 ± 0.001	0.642 ± 0.010
SVMSMOTE	0.5	0.6	0.750 ± 0.001	0.730 ± 0.009
Part IV: Undersampling → DGM Oversampling → Classifier				
DGM	**Sampling**	**USR**	**OSR**	**Train f1**	**Test f1**
vae	Minority	0.5	1.0	0.756 ± 0.001	0.732 ± 0.010
mv-vae	Minority	0.5	1.0	0.755 ± 0.002	0.733 ± 0.010
arae	Minority	0.5	1.0	0.755 ± 0.001	0.733 ± 0.010
mv-arae	Minority	0.5	1.0	0.752 ± 0.001	0.732 ± 0.009
medgan	Minority	0.5	0.6	0.755 ± 0.002	0.732 ± 0.007
mv-medgan	Minority	0.6	0.8	0.752 ± 0.001	0.733 ± 0.010
gan	Minority	0.6	0.7	0.754 ± 0.001	0.734 ± 0.010
mv-wgan	Minority	0.5	1.0	0.752 ± 0.002	0.734 ± 0.008
mv-wgan-gp	Minority	0.6	0.7	0.754 ± 0.001	0.731 ± 0.009
vae	Conditional	0.6	1.0	0.754 ± 0.002	0.732 ± 0.008
mv-vae	Conditional	0.5	0.6	0.755 ± 0.001	0.733 ± 0.007
arae	Conditional	0.5	0.9	0.755 ± 0.002	0.734 ± 0.009
mv-arae	Conditional	0.6	0.9	0.752 ± 0.002	0.732 ± 0.010
medgan	Conditional	0.5	1.0	0.754 ± 0.002	0.733 ± 0.008
mv-medgan	Conditional	0.5	0.8	0.753 ± 0.002	0.732 ± 0.008
gan	Conditional	0.6	0.8	0.755 ± 0.002	0.733 ± 0.011
mv-wgan	Conditional	0.5	0.6	0.752 ± 0.002	0.733 ± 0.005
mv-wgan-gp	Conditional	0.6	0.8	0.752 ± 0.003	0.733 ± 0.008
vae	Rejection	0.5	1.0	0.756 ± 0.002	0.732 ± 0.008
mv-vae	Rejection	0.7	1.0	0.751 ± 0.002	0.732 ± 0.009
arae	Rejection	0.5	1.0	0.755 ± 0.002	0.733 ± 0.007
mv-arae	Rejection	0.6	0.9	0.753 ± 0.002	0.732 ± 0.011
medgan	Rejection	0.6	0.9	0.753 ± 0.002	0.733 ± 0.010
mv-medgan	Rejection	0.5	1.0	0.753 ± 0.001	0.732 ± 0.008
gan	Rejection	*Timeout*			
mv-wgan	Rejection	0.5	0.7	0.754 ± 0.002	0.732 ± 0.007
mv-wgan-gp	Rejection	0.5	0.9	0.754 ± 0.001	0.732 ± 0.008

4.3 Undersampling and Oversampling

All the under- and oversampling algorithms presented in the experiments that do not involve deep generative models are taken from the imbalanced-learn library [22]. Both datasets in the experiments are imbalanced, which means that the imbalance ratio (IR) is less than one, where IR is defined as:

$$IR = \frac{|\{\text{minority class samples}\}|}{|\{\text{majority class samples}\}|}$$

Table 3: Classification experiments for the "Default of credit card clients" dataset.

Part I: Classifier				
IR			**Train f1**	**Test f1**
0.28			0.479 ± 0.006	0.457 ± 0.036
Part II: Undersampling → Classifier				
USR			**Train f1**	**Test f1**
0.80			0.552 ± 0.004	0.534 ± 0.031
Part III: Undersampling → SMOTE Oversampling → Classifier				
Oversampling	**USR**	**OSR**	**Train f1**	**Test f1**
ADASYN	0.8	1.0	0.552 ± 0.004	0.537 ± 0.028
BorderlineSMOTE	0.6	0.7	0.550 ± 0.003	0.535 ± 0.027
KMeansSMOTE	0.7	1.0	0.549 ± 0.004	0.537 ± 0.029
RandomOverSampler	0.8	0.9	0.553 ± 0.005	0.538 ± 0.031
SMOTE	0.6	0.8	0.550 ± 0.004	0.535 ± 0.031
SMOTENC	0.9	1.0	0.488 ± 0.003	0.467 ± 0.029
SVMSMOTE	0.8	0.9	0.549 ± 0.004	0.534 ± 0.029
Part IV: Undersampling → DGM Oversampling → Classifier				
DGM	**Sampling**	**USR**	**OSR**	**Train f1**	**Test f1**
vae	Minority	0.8	0.9	0.550 ± 0.004	0.535 ± 0.030
mv-vae	Minority	0.8	0.9	0.550 ± 0.004	0.533 ± 0.031
arae	Minority	0.7	0.8	0.546 ± 0.004	0.533 ± 0.030
mv-arae	Minority	0.8	0.9	0.548 ± 0.005	0.535 ± 0.030
medgan	Minority	0.8	0.9	0.546 ± 0.005	0.534 ± 0.032
mv-medgan	Minority	0.7	0.8	0.543 ± 0.003	0.534 ± 0.032
gan	Minority	0.8	0.9	0.549 ± 0.004	0.535 ± 0.029
mv-wgan	Minority	0.8	0.9	0.545 ± 0.004	0.534 ± 0.030
mv-wgan-gp	Minority	0.8	0.9	0.548 ± 0.004	0.534 ± 0.031
vae	Conditional	0.7	0.9	0.539 ± 0.004	0.531 ± 0.030
mv-vae	Conditional	0.7	0.8	0.546 ± 0.005	0.532 ± 0.032
arae	Conditional	0.8	0.9	0.549 ± 0.006	0.535 ± 0.030
mv-arae	Conditional	0.8	0.9	0.547 ± 0.004	0.533 ± 0.030
medgan	Conditional	0.8	0.9	0.548 ± 0.003	0.535 ± 0.030
mv-medgan	Conditional	0.8	0.9	0.546 ± 0.004	0.535 ± 0.032
gan	Conditional	0.8	0.9	0.550 ± 0.003	0.535 ± 0.029
mv-wgan	Conditional	0.8	0.9	0.546 ± 0.004	0.536 ± 0.032
mv-wgan-gp	Conditional	0.9	1.0	0.545 ± 0.006	0.534 ± 0.028
vae	Rejection	0.7	0.8	0.543 ± 0.004	0.532 ± 0.031
mv-vae	Rejection	0.7	0.8	0.545 ± 0.004	0.534 ± 0.031
arae	Rejection	0.9	1.0	0.546 ± 0.004	0.532 ± 0.032
mv-arae	Rejection	0.8	0.9	0.547 ± 0.003	0.534 ± 0.032
medgan	Rejection	*Timeout*			
mv-medgan	Rejection	*Timeout*			
gan	Rejection	*Timeout*			
mv-wgan	Rejection	0.8	0.9	0.547 ± 0.004	0.535 ± 0.032
mv-wgan-gp	Rejection	0.9	1.0	0.547 ± 0.004	0.535 ± 0.033

We undersample the majority class on the train set, applying a random under sampler, removing samples from the majority class with a uniform probability until reaching the desired sample size (smaller than the original). The result contains a larger IR that we call the "undersampling ratio" (USR). Afterward, we train and evaluate the classification model and compute the respective metrics. In the Part II of Tables 1, 2 and 3 we present the USR that provide the best mean test f1 score for each dataset. Furthermore, we oversample the minority class on the original train set and also on each train set that was previously undersampled. The result contains a larger IR that we call the "oversampling ratio" (OSR). We repeat the process for incrementally larger values for the OSR and different oversampling algorithms. Note that the OSR can reach values greater than 1 when we grow the minority class to be bigger than the majority class, but we will not change the classes' names for

clarity. In the Part III of Tables 1, 2 and 3 we present the USR-OSR combination that provide the best mean test f1 score for each algorithm on each dataset.

4.4 Oversampling with Deep Generative Models

For the final experiments, we combine nine deep generative models (DGM) with three different ways of sampling from them (presented in Section 3). All the models are implemented with PyTorch [33]. The models' training is done based on the training set. Additionally, the validation set is also used to evaluate the reconstruction's quality when an autoencoder is present in the architecture. After the model training, the undersampling is applied to the training set, and afterward, the trained model generates synthetic samples that are appended to the training set based on the OSR. In the Part IV of Tables 1, 2 and 3 we present the USR-OSR combination that provide the best mean test f1 score for each dataset. Note that the rejection sampling can "timeout" if it fails to obtain the desired number of samples after 10,000 iterations.

4.5 Results

We start analyzing the results shown in Table 1 for the dataset "Credit card fraud". Given that the dataset contains no categorical or binary features, we do not implement the multi-variable models for this scenario. The class imbalance is very high, and our experiments involving considerable changes to the imbalance ratio after under- and oversampling resulted in a significant deterioration of the classification scores. Small changes on the imbalance ratio after undersampling do not offer classification improvements either, but oversampling a bit appears to be useful in some cases. In our experiments with this dataset, SMOTE presents the best classification score, while most of the deep generative models offer a slight improvement over the classification without oversampling. Experiments with SMOTE and "minority" GAN were carried out with the same dataset in [11], where both models presented very close results. Besides the scores for SMOTE, our results are comparable to this study. We believe that the small discrepancies are related to the stochastic nature of the experiments. Additionally, we can see by the differences between the train and test metrics that the classification tends to overfit every case, which is expected considering the small number of positive samples. Finally, all the models implementing rejection sampling reached a timeout, which is expected given that the probability of drawing a sample that belongs to the minority class is very low.

Now we compare all the results from Table 2 and Table 3. Regarding the classification baseline without oversampling or undersampling, we can see that both problems are quite challenging since the mean test f1 score is considerably far from 1. Furthermore, adding only random undersampling to the pipeline shows some improvement in Table 2 but presents a considerable improvement in Table 3. Nevertheless, most of the oversampling techniques (both with and without deep generative models) only show a slight improvement on the third decimal of the mean test f1 score on top of the undersampling. The only oversampling algorithm that seems to worsen the classification quality is SMOTE-NC. Note also that in Table 3 the RandomOverSampler –which is the most straightforward oversampling technique– performs slightly better than the rest.

5 Discussion and Conclusion

Our experiments show several trends that persist across different generative methods and datasets: First, undersampling the majority class without any oversampling improves the classifier. Adding oversampling via a simple baseline such as SMOTE only leads to marginal improvements. Second, all generative models provide very close results to the SMOTE baselines. Finally, the sampling strategy does not substantially impact the results' quality, but rejection sampling is the slowest approach.

It is noteworthy that generative models require different under- and oversampling settings to archive the best performance. However, regardless of the method used, the absolute improvement on the classification metric (F1 score) is often minimal. This phenomenon occurs despite results showing that in a ranking of methods, deep generative methods' improvement is statistically significantly better [9]. Even if the f1 score is widely used in this domain, it could be argued that it is not fair to compare results with this metric between experiments that have different imbalance ratios. We also computed the area under the Precision-Recall curve for all the experiments, obtaining similar results. There is not a clear alternative in the literature, which implies a possible research opportunity.

It seems that the performance gain given by deep generative models for oversampling has to be seen in context. They possess a considerably more complicated setup and longer training time than the best-performing baseline, a simple random under- and oversampling approach.

Regarding the sampling strategies, it is reasonable to expect a lousy sample quality when training deep learning models with a small amount of data from the minority class. This situation directly affects the minority and the rejection strategies, but it is even worse for the latter because many draws are needed to obtain the desired output. On the other side, for the conditional strategy, it is not very clear if learning the majority class's distribution would help define better the distribution of the minority class. Finally, besides the samples' quality, it is not very clear either if powerful and complex machine learning algorithms like XGBoost can be benefited by injecting synthetic samples to the minority class. There might be a combination of simpler models and oversampling methods that could achieve better results, but following the evidence from this study, finding a useful combination might require many experiments. Brute forcing through combinations of tools, models, or techniques should not be the way of practicing science, but there is a lack of general understanding about what these generative models produce. In the meantime, more modifications to existing ideas keep emerging, claiming that they perform better than previous ideas by measuring convenient aspects of the experiments or presenting unfair comparisons. The lack of novelty or the not-so-exciting results is usually frowned upon by reviewers of top-conferences. Consequently, hours of work are never reported, forcing the scientific community to venture the same fruitless experiences repeatedly.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *arXiv preprint arXiv:1701.07875*, 2017.

[2] Ramiro Camino, Christian Hammerschmidt, and Radu State. Generating multi-categorical samples with generative adversarial networks. *arXiv preprint arXiv:1807.01202*, 2018.

[3] Ramiro D Camino, Christian A Hammerschmidt, and Radu State. Improving missing data imputation with deep generative models. *arXiv preprint arXiv:1902.10666*, 2019.

[4] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[5] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[6] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F. Stewart, and Jimeng Sun. Generating Multi-label Discrete Patient Records using Generative Adversarial Networks. *arXiv:1703.06490 [cs]*, March 2017. arXiv: 1703.06490.

[7] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F Stewart, and Jimeng Sun. Generating multi-label discrete patient records using generative adversarial networks. *arXiv preprint arXiv:1703.06490*, 2017.

[8] Andrea Dal Pozzolo, Olivier Caelen, Reid A Johnson, and Gianluca Bontempi. Calibrating probability with undersampling for unbalanced classification. In *2015 IEEE Symposium Series on Computational Intelligence*, pages 159–166. IEEE, 2015.

[9] Georgios Douzas and Fernando Bacao. Effective data generation for imbalanced learning using conditional generative adversarial networks. *Expert Systems with applications*, 91:464–471, 2018.

[10] D. Dua and E. Karra Taniskidou. UCI Machine Learning Repository. 2017.

[11] Ugo Fiore, Alfredo De Santis, Francesca Perla, Paolo Zanetti, and Francesco Palmieri. Using generative adversarial networks for improving classification effectiveness in credit card fraud detection. *Information Sciences*, 479:448–455, 2019.

[12] Lovedeep Gondara and Ke Wang. Mida: Multiple imputation using denoising autoencoders. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 260–272. Springer, 2018.

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[14] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5769–5779, 2017.

[15] Uiwon Hwang, Dahuin Jung, and Sungroh Yoon. Hexagan: Generative adversarial nets for real world classification. *arXiv preprint arXiv:1902.09913*, 2019.

[16] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. *arXiv:1611.07004 [cs]*, November 2016. arXiv: 1611.07004.

[17] Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparameterization with Gumbel-Softmax. *arXiv:1611.01144 [cs, stat]*, November 2016. arXiv: 1611.01144.

[18] Junbo, Zhao, Yoon Kim, Kelly Zhang, Alexander M. Rush, and Yann LeCun. Adversarially Regularized Autoencoders. *arXiv:1706.04223 [cs]*, June 2017. arXiv: 1706.04223.

[19] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to Discover Cross-Domain Relations with Generative Adversarial Networks. *arXiv:1703.05192 [cs]*, March 2017. arXiv: 1703.05192.

[20] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*, December 2013. arXiv: 1312.6114.

[21] Matt J. Kusner and José Miguel Hernández-Lobato. GANS for Sequences of Discrete Elements with the Gumbel-softmax Distribution. *arXiv:1611.04051 [cs, stat]*, November 2016. arXiv: 1611.04051.

[22] 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.

[23] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[24] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. *arXiv:1611.00712 [cs, stat]*, November 2016. arXiv: 1611.00712.

[25] Vikash Mansinghka, Richard Tibbetts, Jay Baxter, Pat Shafto, and Baxter Eaves. BayesDB: A probabilistic programming system for querying the probable implications of data. *arXiv:1512.05006 [cs]*, December 2015. arXiv: 1512.05006.

[26] Pierre-Alexandre Mattei and Jes Frellsen. Miwae: Deep generative modelling and imputation of incomplete data. *arXiv preprint arXiv:1812.02633*, 2018.

[27] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which Training Methods for GANs do actually Converge? *arXiv:1801.04406 [cs]*, January 2018. arXiv: 1801.04406.

[28] Mehdi Mirza and Simon Osindero. Conditional Generative Adversarial Nets. *arXiv:1411.1784 [cs, stat]*, November 2014. arXiv: 1411.1784.

[29] Ajinkya More. Survey of resampling techniques for improving classification performance in unbalanced datasets. *arXiv:1608.06048 [cs, stat]*, August 2016. arXiv: 1608.06048.

[30] Alejandro Mottini, Alix Lheritier, and Rodrigo Acuna-Agost. Airline passenger name record generation using generative adversarial networks. *arXiv preprint arXiv:1807.06657*, 2018.

[31] Alfredo Nazabal, Pablo M Olmos, Zoubin Ghahramani, and Isabel Valera. Handling incomplete heterogeneous data using vaes. *arXiv preprint arXiv:1807.03653*, 2018.

[32] Wing WY Ng, Guangjun Zeng, Jiangjun Zhang, Daniel S Yeung, and Witold Pedrycz. Dual autoencoders features for imbalance classification problem. *Pattern Recognition*, 60:875–889, 2016.

[33] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. 2017.

[34] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv:1511.06434 [cs]*, November 2015. arXiv: 1511.06434.

[35] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. In *Advances in Neural Information Processing Systems*, pages 7333–7343, 2019.

[36] Jinsung Yoon, James Jordon, and Mihaela Van Der Schaar. Gain: Missing data imputation using generative adversarial nets. *arXiv preprint arXiv:1806.02920*, 2018.

[37] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. In *AAAI*, pages 2852–2858, 2017.