Data augmentation using GANs

Directed Research in Computer Science B

Fábio Henrique K. dos S. Tanaka

January 2019

Abstract

In the field of data science, having a good database is essential for training any model. For different reasons it may be hard to have access to one of these, some common reasons are: the access to data may be regulated because of privacy concerns or the database could be highly imbalanced for the target class.

In this scenario the generation of synthetic data could help both cases, it could balance a database by oversampling the minority class or create an entirely new dataset based on the original while retaining important information. This research uses GANs (Generative Adversarial Neural networks) to produce this synthetic data and shows that it can be suitable for the task. Then, the efficiency of this strategy is compared to other existing methods like SMOTE and ADASYN.

This is a report for the subject Directed Research in Computer Science from October 2018 to February 2019

1 Introduction

When working with machine learning, it is important to have a high-quality dataset to train the algorithm. This means that the data should not only be sufficiently large, to cover as many cases as possible, but be a good representation of the reality. Having a good dataset permits the program to have a better understanding of the underlying characteristics of the data and making easier to generalize these traits. In this scenario, the creation of synthetic data can useful for several reasons like oversampling minority classes and generating new datasets to keep the privacy of the originals.

The first reason, oversampling minority class, is relevant when trying to learn from imbalanced datasets. In many instances, it is common for databases to have classes that are underrepresented, for example, when dealing with credit card frauds the ratio between normal and fraudulent transactions can be 10000 to 1, the same can happen when analysing medical information where the number of healthy patients is much higher than affected ones. When this happens, classification algorithms may have difficulties to identify the minority classes since the program would still have a low error even if it classifies all the minority classes wrong. To avoid this problem it is possible to augment the minority data, this is, create new entries by tweaking the original in meaningful ways, this not only increases the representation but it may help to avoid overfitting as well.

The second reason, generating new datasets, is used to avoid using the original data for privacy reasons. It is possible that a database contains sensitive information and working on it directly could have risks of it being misused or breached. For example, medical records could have many personal information about the patients, even without the names it could be possible to identify them using attributes like date of birth, weight, height, etc. Because of this, it is understandable that many regulations exist on this kind of database and its spread is controlled. One possible approach to this problem is to not use the original data but generate a synthetic dataset sufficiently realistic based on it.

On this research, Generating Adversarial Networks (GANs) are used to try to deal with the previous mentioned cases. Both of them will have synthetic data generated by a GAN and then the quality of it will be tested and compared using it to train a decision tree and testing on real data. Finally, this approach will compare its efficiency to other algorithms like SMOTE [5] and ADASYN [9].

2 Background

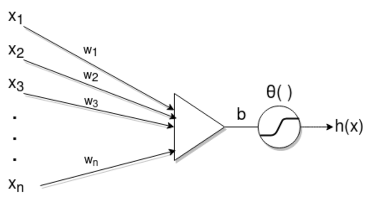
In this section, it will be explained the basic concepts and algorithms needed to understand this research.

2.1 Artificial Neural Networks (ANNs)

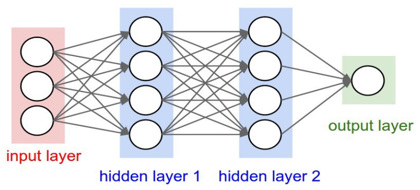
Before explaining the main method of how data is produced on in this research, it is important to have a basic understatement of how Artificial Neural Networks (ANNs) works. They are a model inspired by the biological neural networks that constitute animal brains.

An ANN is a collection of smaller units called Neurons. Each of these units receive as input an numeric array X, multiply it by a weight vector W and sum its values. Then it is added a bias b and finally, to make the result be between 0 and 1, it is applied an activation function θ. In the end, each Neuron computes the operation $\theta(W^T X + b)$ and outputs the result.

These Neurons are organized in a layered network such that the first layer receives the original input. The intermediate layers, also know as hidden layers, receive as input the results from the previous level. Finally, the last layer returns the value of its Neurons as an array as the output.



(a) Neuron



(b) Artificial Neural Network (from [1])

Figure 1: Examples of a Neuron and an ANN

In the beginning, an ANN is initialized with random weights (W) and bias (b) and because of that its early results are not precise. To evaluate the quality of the outputs it is used an error or cost function, one commonly used is the mean squared error.

$$loss = \frac{1}{n}\sum_{i=0}^{n-1} \parallel \hat{y}_i - y_i \parallel^2.$$

In this function, n is the size of input, $\hat{y}_i$ is the expected output of of the example i and y_i is the predicted result by the network. The values of W and b are modified after some executions with the goal to minimize the error function. Gradient descent and backpropagation are some techniques used to perform this operation, these definitions can be found in [18].

2.2 Generating Adversarial Networks (GANs)

Generating Adversarial Networks, or GANs for short, were first introduced by Ian Goodfellow in 2014 [8]. Since then, many researches have been done using the framework and Facebook's AI research director Yann LeCun recognized it as "the most interesting idea in the last 10 years in machine learning" [14]. GANs are a type of generative model, this means that it can produce new content based on its training data.

GANs can have a variety of applications, developing a new molecules for oncology [2] and increasing resolution [15] are some of them. The most common use is to generate new images, below it is possible to see an example of faces generated by a GAN based on a dataset composed by photos of famous people [11].

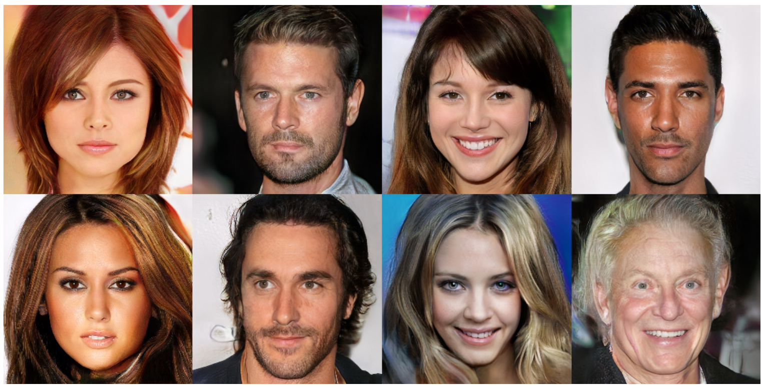


Figure 2: Faces generated by a GAN [11]

A GAN is made of 2 ANNs that compete against each other, the generator and the discriminator. The former generate new data instances while the latter evaluates them for authenticity.

The discriminator is responsible to evaluate the quality of the images created by the generator. It receives as input some data that can be from the original database or from the generator and tries to predict the chances of it being real, with 0 being false and 1 being real. The pseudo-code of its training can be seem below:

```
Function train_discriminator(originalData, fakeData):
    predictionReal = discriminator(realData)
    errorReal = loss(predictionReal, 1)
    errorReal.backpropagate()

    predictionFake = discriminator(fakeData)
    errorFake = loss(predictionFake, 0)
    errorFake.backpropagate()
return
```

The generator is a generative network, it learns to map the latent space into the distribution of the data that it is interested in, this is, it tries to predict the features of the data. The generator is evaluated by the discriminator, this means that its goal is to create data that close to the original and by doing this, make the other network predict it as real. Below it is the pseudo-code of the training process of the generator, it is important to note that it tries to make its output be predicted as 1:

```
Function train_generator(latentSpace):
    fakeData = generator(latentSpace)
    prediction = discriminator(fakeData)
    error = loss(prediction, 1)
    error.backpropagate()
return
```

By training the two networks at the same time both of them will get better by competing against one another, hence the name Generative Adversarial Networks. The discriminator will try to get better at distinguishing fake and real data and the generator is going to output data closer to the real one.

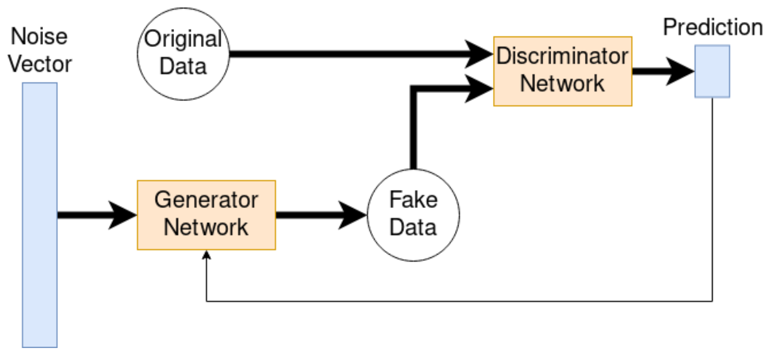


Figure 3: How a GAN works

The use of GANs have many advantages, they can create high quality data and be modeled to deal with different problems. On the other hand, their use have some disadvantages and difficulties as well, some of them are: it is hard to generate discrete data (like text), they are hard to train and require large processing power, and like any ANN its model can unstable or it can overfit the data.

For the applications addressed in this research (balancing data and generating synthetic datasets) some others researches have already been done using

GANs [17] [23] [7], but they all were related to images while this project the focus are fully numerical databases. Even if the approaches are similar, there is a difference in the implementation since the use convolution and other techniques does not apply to non-images datasets.

2.3 Decision Tree Classifier

In order to evaluate the accuracy of the data created by the GAN this research uses an classification algorithm. Classification is the process of predicting the class of the data based on its features. After training using a database containing a variety of observations, it is expected that the program will be able to identify the category of given example by its characteristics. In this research, Decision Trees Classifiers is the chosen strategy.

A decision tree classifier is a non-parametric supervised learning method used for classification, it tries to create a model that can distinguish the class of an input using simple decision rules. A decision tree is built like a flowchart where each internal node is a question, each branch one possible answer and each leaf node represents a class.

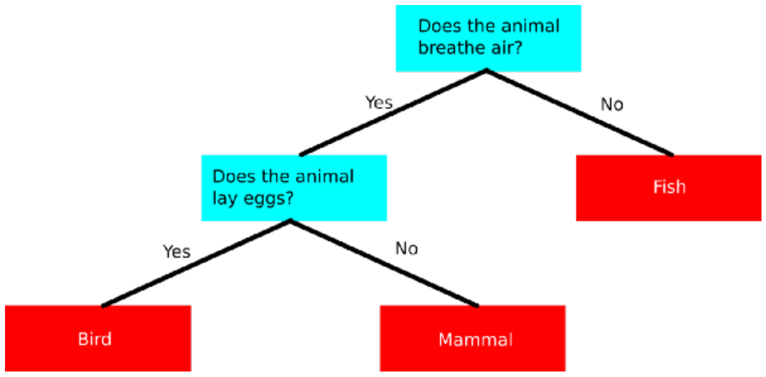


Figure 4: Simple decision tree to classify the type of animal (from [4])

There are different algorithms to construct decisions trees [20][21] and in this research its used CART (Classification and Regression Trees) [3] since it is the one implemented in the scikit-learn [19] library. It builds the tree using questions with binary answers and partitions the data using them, the questions are chosen based on gini impurity and information gain.

Gini impurity of a set is a measure of how often a randomly chosen element from it would be labeled wrong if it its label was chosen randomly according to the distribution of labels in the subset. Information gain of a question is the difference between the current gini impurity and the weighted average of the subsets created by it. It is chosen the question with the higher information gain.

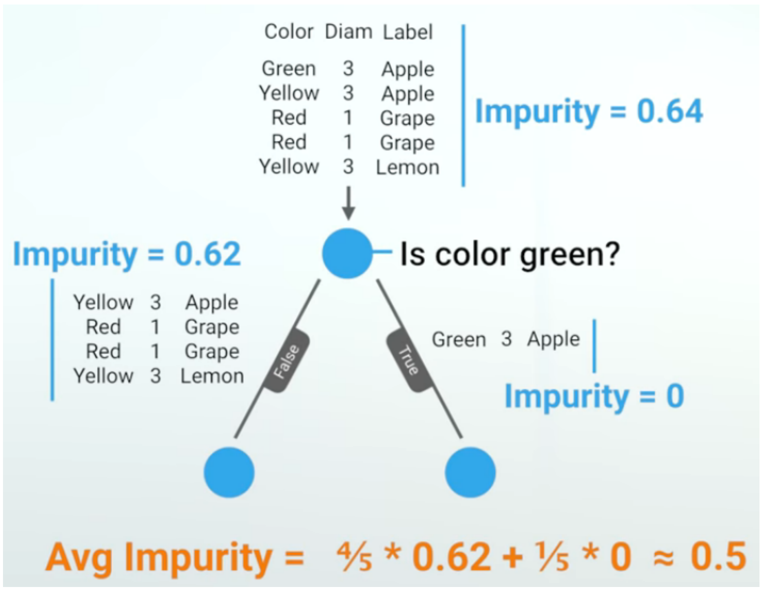


Figure 5: Calculation of the gini impurity. In this case the information gain would be $0.64 - 0.5 = 0.14$ (from [6])

Decisions trees were chosen because they are simple to understand and interpret since they can be visualized and because it requires little data preparation. On the other hand, it can create over-complex models that do not generalize well or be unstable because small variations in the data might result in a completely different tree being generated.

2.4 SMOTE and ADASYNC

SMOTE [5] and ADASYN [9] are two approaches to oversampling the minority classes with the goal to balance numerical datasets. They are both implemented in the imbalanced-learn [16] python library.

Synthetic Minority Over sampling Technique (SMOTE) create synthetic samples based on the position of the data. First it randomly selects a point in the minority class, them finds the k nearest neighbors of the same class. For each of these pairs a new point is generated in the vector between them, this new point is located in a random percent of the way from the original point.

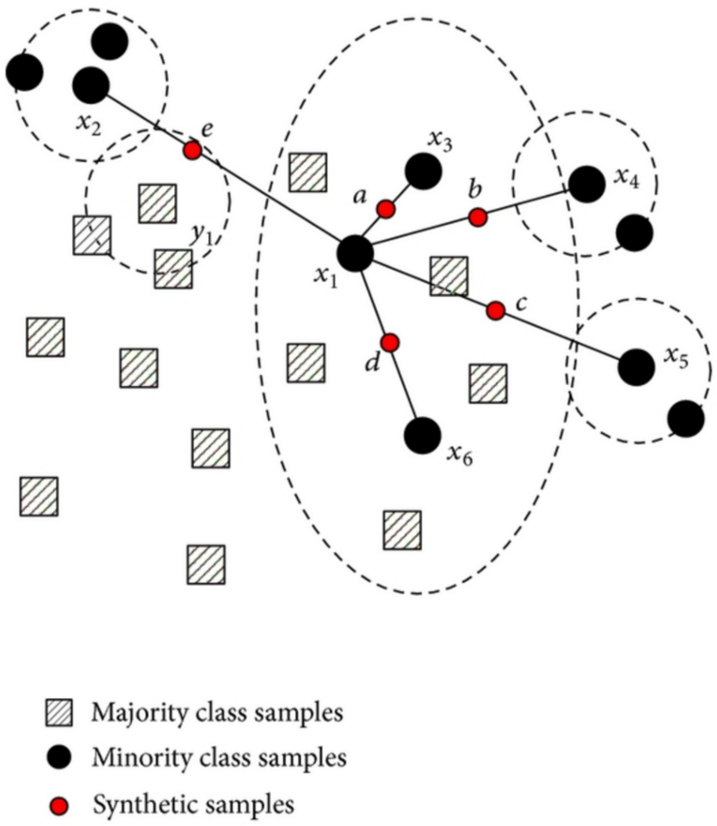


Figure 6: Example of SMOTE (from [10])

ADASYN is similar to SMOTE, and is derived from it. They function on the same way but after creating the samples, ADASYN adds a random small bias to the points, making them not being linearly correlated to their parents. Even though this is a small change it increases the variance in the synthetic data

3 Methods

In this project it will be discussed two different uses for generating synthetic data, creating new databases to preserve privacy and oversampling minority classes to balance datasets. The methods used for the experiments will be discussed in this section.

3.1 Architecture of the GANs and Decisions Trees

In the two approaches a GAN is used to generate the new data. Both of them uses the same architecture with the following parameters:

- *leaky ReLU* as the activation function with a negative slope of 0.2.

- batch size = 5
- learning rate = 0.0002
- Use of dropout in the generator with a probability of 0.3.
- Binary cross-entropy as the loss function.
- Adam as the optimizer algorithm.
- No convolution layers.
- In the generator, if there are more than one layer, they are ordered in ascending size.
- In the discriminator, if there are more than one layer, they are ordered in descending size.

Leaky ReLU, Adam optimizer and the use of dropout were chosen because they are the standard for this kind of problem [22]. There are no convolutional layers because the input is not an image and the Binary cross-entropy loss is used because it is the most fit to measure the performance of a model whose output is a probability between 0 and 1. This architecture was tested using different numbers and sizes of layers, the results of it will be discussed on the next chapters.

The Decision Trees had a maximum depth of 3 to avoid overfitting the data and algorithm used to build it was the CART [3] since it is the one implemented in the scikit-learn [19] library.

3.2 Databases

The experiments on this research were done using the following 3 different databases:

- **Pima Indians Diabetes Database [13]:** This dataset consists of 8 independent variables and one target dependent class that represents if the patient has Diabetes or not. It is composed by 768 examples in which 268 patients presents the disease.

- **Breast Cancer Wisconsin (Diagnostic) Data Set [12]:** This database features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image and has a target class to determine if the cancer is benign or malign. Its distribution is 357 benign and 212 malignant examples.

- **Credit Card Fraud Detection [24]:** This dataset, contains transactions made by credit cards in September 2013 by European cardholders. It presents transactions that occurred in two days, where there are 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

The goal class on the *Breast Cancer Wisconsin (Diagnostic) Data Set* was represented by a char, "B" = benign and "M" = malignant but this label was changed to 0 and 1 respectively to be consistent with the other examples. With the exception of this case, all attributes are fully numeric and the databases were chosen because of this. Working with fully numeric examples allows the GANs to generate discrete results and they are faster to compute and train when compared to images. Each of these datasets was divided in two subsets, one for training the GAN (the first 70% of the original data) and other for testing it (formed by the remaining 30%).

Database Name	Number of features	Size	Class Distribution
Pima Indians Diabetes Database	9	768	No diabetes: 500, Diabetes: 268
Breast Cancer Wisconsin (Diagnostic) Data Set	32	569	Benign: 357, Malignant: 212
Credit Card Fraud Detection	31	284807	Non-frauds: 284315, Frauds: 492

Table 1: Databases used in this research

Notice that before operating in these databases their classes values were all scaled to be in the interval [0,1] by the min-max method. It was done because it makes the range for all attributes to be the same and avoids one of them to dominate the others because of its scale. This reduces the range of values that the generator has to produce as well.

4 Experiments and Results

In this section it will be explained the experiments and the results obtained from them. They were divided in 2 groups, one to test the generation of new synthetic databases and other to test the balance of datasets by oversampling the minority class.

4.1 New Synthetic Databases

To evaluate the creation of synthetic data the experiments were done using the following steps:

1. Trained the GAN using the full training subset of the original database for 1500 epochs.
2. Used the newly trained GAN to generate the new synthetic dataset with the exact size of the original.
3. Since the GANs generated the classification label as a continuous value between 0 and 1, this value has to be turned to a discrete by rounding it to the nearest integer.
4. The new dataset is used to train a classification tree.
5. The tree is tested using the test subset of the original dataset.

It is important to note that the GAN was trained using the label classes as well. This means that the the data generated can be have any of the classes and the GAN itself defines how each data point should be classified.

The tests were done in the diabetes and cancer databases, they both are not very unbalanced and have less than 1000 entries. Below, it is possible to see how the distribution of one some classes of the new synthetic cancer dataset compared to the original.

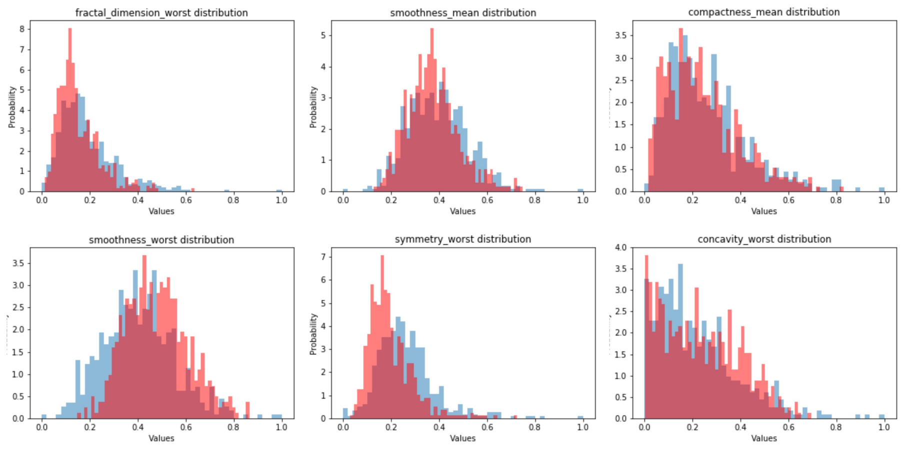


Figure 7: Comparison between the distribution of the data in the original (blue) and synthetic (red) cancer datasets

4.1.1 Synthetic Breast Cancer Wisconsin (Diagnostic) Data Set

These are some early results from tests using 1500 epochs:

Database	Label Proportion	Test Accuracy
original data	56.53/43.47	0.888
3 big layers	52.26/47.74	0.818
2 big layers	56.28/43.72	0.941
1 big layers	56.78/43.22	0.906
3 small layers	54.02/45.98	0.953
2 small layers	58.04/41.96	0.935
1 small layers	54.27/45.73	0.912

Table 2: Early results

Then, this is a more in-depth analysis of the results, it is the mean of 20 experiments using the synthetic dataset, the value between parenthesis is the standard deviation:

Database	Accuracy	Precision	Recall
Original data	0.888	0.679	0.974
3 Big layers	0.91 (0.042)	0.786 (0.127)	0.888 (0.054)
2 Big layers	0.935 (0.03)	0.853 (0.107)	0.896 (0.086)
1 Big layers	0.907 (0.053)	0.772 (0.126)	0.904 (0.071)
3 Small layers	0.869 (0.066)	0.702 (0.144)	0.821 (0.148)
2 Small layers	0.896 (0.048)	0.74 (0.122)	0.908 (0.086)
1 Small layers	0.906 (0.054)	0.775 (0.131)	0.894 (0.055)

Table 3: Mean of the results of 20 experiments

4.1.2 Pima Indians Diabetes Database

Results from a single test using 1500 epochs:

Database	Label Proportion	Test Accuracy
original data	64.8/35.2	0.748
3 big layers	71.69/28.31	0.7
2 big layers	67.23/32.77	0.548
1 big layers	67.6/32.4	0.748
3 small layers	60.15/39.85	0.661
2 small layers	65.18/34.82	0.739
1 small layers	68.9/31.1	0.697

Table 4: Early results

More in-depth analysis, done in the same way as the cancer database:

Database	Accuracy	Precision	Recall
Original data	0.748	0.784	0.367
3 Big layers	0.682 (0.064)	0.545 (0.093)	0.534 (0.206)
2 Big layers	0.706 (0.05)	0.582 (0.078)	0.584 (0.097)
1 Big layers	0.601 (0.097)	0.438 (0.118)	0.438 (0.213)
3 Small layers	0.685 (0.058)	0.568 (0.109)	0.544 (0.158)
2 Small layers	0.639 (0.094)	0.507 (0.106)	0.579 (0.185)
1 Small layers	0.653 (0.086)	0.51 (0.117)	0.462 (0.219)

Table 5: Mean of the results of 20 experiments

4.2 Oversampling the minority class

To evaluate the effects of balancing the dataset by oversampling the minority class the following steps were done:

1. Separated the training set based on the target class (For example, the credit card database was separated in non-frauds and frauds).
2. Trained the GAN using only the data in the set with the minority class. The label was used in the training as well.
3. Used the GAN to produce new entries to the training dataset until it is balanced.
4. Used the newly balanced training dataset to create a classification tree.
5. The trees were tested using two databases: the original test set and a balanced version of it obtained by undersampling the majority class.

Similar to the experiment before, the results shown are the mean from 5 experiments with the value in parenthesis being the standard deviation. Using the

SMOTE and ADASYN implementation in the imblearn library always resulted in the same outcome, so the standard deviation in this cases were 0.

Database	Accuracy	Precision	Recall
Original	0.999	0.896	0.556
SMOTE	0.958	0.026	0.861
ADASYN	0.958	0.026	0.861
1 Small layers	0.798 (0.372)	0.051 (0.029)	0.806 (0.042)
1 Big layers	0.986 (0.005)	0.077 (0.031)	0.789 (0.018)
2 Small layers	0.974 (0.01)	0.045 (0.02)	0.82 (0.028)
2 Big layers	0.964 (0.017)	0.033 (0.013)	0.808 (0.069)

Table 6: Outcomes resulted from the test on an **unbalanced** test set

Database	Accuracy	Precision	Recall
Original	0.782	1.0	0.565
SMOTE	0.912	0.959	0.861
ADASYN	0.921	0.979	0.861
1 Small layers	0.807 (0.165)	0.89 (0.202)	0.806 (0.042)
1 Big layers	0.894 (0.01)	0.998 (0.005)	0.789 (0.018)
2 Small layers	0.902 (0.012)	0.981 (0.015)	0.82 (0.028)
2 Big layers	0.888 (0.032)	0.962 (0.018)	0.808 (0.069)

Table 7: Outcomes resulted from the test on an **balanced** test set

5 Results Discussion

In this section it will be discussed the results of the experiments, this will be done in two parts, one for the creation of entirely new databases and other for balancing the training data.

5.1 New Synthetics Databases

The early results in both databases showed that GANs can produce data in approximately the same distribution of the original even if they aren't instructed to do so. This is a result of training the GAN with the complete training dataset, thered were no separation of the classes. Based on this, there is no need to treat or separate the data before training, meaning that the provider of the data don't have to provide additional information or take any action in this aspect.

When considering the mean results, the GAN with 2 big layers ($2^8, 2^9$) had the best overall outcomes in the two datasets. It is hard to precise explain why this architecture was better but it should be used in future experiments. The next considerations will be done using it as the parameter.

Analysing the accuracy, the results from the classification using the synthetic data as training set were not far from the original, in the Cancer database it was even better.Based on this parameter alone, using the new datasets should not affect the results too much, making them suitable for its goal.

But when using classification on medical process, as is the case in the databases, the most important aspect should be the recall since a false negative can be disastrous when deciding if a patient should be treated or not. In this aspect, the use of GAN is much more suited for the Diabetes database because even if the accuracy and precision decreased, the recall got better. The same argument can be applied to not use the GAN for the Cancer database.

By only these experiments alone, the GANs were able to reproduce the original data results fairly well. On the other hand, depending on the most valued aspect of the data (precision or recall) this approach could result in more false positives or false negatives. This cases should be evaluated individually for each case since based only on the results on this research, it is inconclusive what attributes make the GANs have better or worse precision and recall.

5.2 Oversampling Minority Classes

Before discussing this experiment, it is important to note that one of the GANs with 1 small layer was an outlier with bad results. Since only 5 experiments were done with each architecture, excluding this example could result in loss of information, this is the reason it was included in the calculation.

The first aspect to analyse is that independent of the method, oversampling the minority class give the classification algorithm a better recall value. This is to be expected because with not enough examples in the training set, the classification algorithm would have difficulties in identifying positive labels resulting in a large number of false negatives.

The second aspect to focus should be that the results are much better in a balanced test set. As before, this make sense because the classification algorithm can identify positive labels easier. Even so, the results in the unbalanced test set should have more importance because it is not reasonable to expect that real data will be balanced, especially when the training databases were so unbalanced.

With that in mind, when dealing with a unbalanced test set, the GAN had better accuracy and precision and worse recall when compared to ADASYN and SMOTE. The importance on each of these results may vary depending on the data set but in this case, fraud evaluation on credit cards, it is much more important to identify frauds, and as such, a good recall is desired. In this aspect the two algorithms are slightly better than the GAN.

Even so, the use of GAN was proved useful because it had increased the results when compared to the original dataset. It even could be better than the other algorithms depending on which aspect is more valuable.

6 Conclusion

The use of GANs to generate synthetic data proved itself viable for both balancing the dataset and creating a new database. Even if there are other methods available, it would be interesting to try using an autoencoder to initialize the GAN as suggested in BAGAN [17].

Since in this research only 3 databases were analysed, it is not possible to assure any connection between the number of classes in the set and the results. Other aspects not discussed about the databases were the quality, distribution and how they deal with wrong entries or missing data and how this can affect the GAN performance. These all would be interesting topics for a next research.

References

[1] Andrej Karpathy. Cs231n convolutional neural networks for visual recognition, 2018. [Online; accessed February 8, 2019].

[2] Andrey Kazennov Polina Mamoshina Quentin Vanhaelen Kuzma Khrabrov Alex Zhavoronkov Artur Kadurin, Alexander Aliper. The cornucopia of meaningful leads: Applying deep adversarial autoencoders for new molecule development in oncology.

[3] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Wadsworth and Brooks, Monterey, CA, 1984.

[4] Charlie Bickerton. A beginner's guide to decision tree classification, 2018. [Online; accessed February 8, 2019].

[5] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal Of Artificial Intelligence Research, Volume 16, pages 321-357, 2002*, 2011.

[6] Google Developers. Let's write a decision tree classifier from scratch - machine learning recipes 8.

[7] Maayan Frid-Adar, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. Synthetic data augmentation using gan for improved liver lesion classification, 2018.

[8] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

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

[10] Indresh Bhattacharyya. Smote and adasyn (handling imbalanced data set), 2018. [Online; accessed February 8, 2019].

[11] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2017.

[12] UCI Machine Learning. Breast cancer wisconsin (diagnostic) data set.

[13] UCI Machine Learning. Pima indians diabetes.

[14] Yann LeCun. What are some recent and potentially upcoming breakthroughs in deep learning?, July 2017.

[15] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network, 2016.

[16] 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.

[17] Giovanni Mariani, Florian Scheidegger, Roxana Istrate, Costas Bekas, and Cristiano Malossi. Bagan: Data augmentation with balancing gan, 2018.

[18] Michael A. Nielsen. *Neural Networks and Deep Learning.* Determination Press, 2015.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[20] J. R. Quinlan. Induction of decision trees. *MACH. LEARN*, 1:81–106, 1986.

[21] Steven L. Salzberg. C4.5: Programs for machine learning. *MACH. LEARN*, 16:235–240, 1994.

[22] Martin Arjovsky Michael Mathieu Soumith Chintala, Emily Denton. How to train a gan? tips and tricks to make gans work, 2016.

[23] Jost Tobias Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks, 2015.

[24] Machine Learning Group ULB. Credit card fraud detection.