Balancing imbalanced Text, Image and Time series data using GANs: A Comparative Study

Arjun Ramesh
1225371667
arames65@asu.edu

Haritha Athinarayanan
1225396744
hathinar@asu.edu

Nitheese Thirumoorthy
1225417336
nthirumo@asu.edu

Surya Siddharthan Sivakumar
1225488758
ssivak18@asu.edu

ABSTRACT

Imbalanced datasets are a common problem in machine learning, leading to biased models and poor performance on minority classes. Methods such as resampling, cost-sensitive learning, and ensemble methods have been proposed to address this issue. Generative Adversarial Networks (GANs) have emerged as a powerful tool for generating synthetic data to balance imbalanced datasets for various data types. This project investigates the efficacy of using GANs to create synthetic data, that can balance imbalanced datasets. The aim is to use GANs to produce synthetic data that can enhance the minority class, thereby increasing the number of under-represented samples in the dataset. The effectiveness of GANs is evaluated by comparing the performance of models trained on the original imbalanced dataset and the balanced dataset generated using GANs.

KEYWORDS

GAN, CNN, RNN, LSTM, Boosting, Data Augmentation, Time Series

1 INTRODUCTION

Imbalanced datasets are a common problem in machine learning applications. Imbalanced datasets occur when the distribution of samples across different classes is skewed, with some classes having a significantly smaller number of samples than others. This can result in models that are biased towards the majority class and perform poorly on the minority class. To address this issue, various methods have been proposed, including resampling techniques, cost-sensitive learning, and ensemble methods. This problem is particularly prevalent in real-world applications such as medical diagnosis, fraud detection, and anomaly detection, where the under-represented classes are often of high importance.

In recent years, the use of GANs has become increasingly popular due to their ability to generate synthetic data, which can be utilized to improve the representation of the minority class and achieve a balanced dataset. GANs are a type of neural network that can learn the underlying distribution of the data and generate new samples that resemble the original data. By generating synthetic data for under-represented classes, GANs can balance the imbalanced dataset, thereby improving the performance of the models. Despite the popularity of GANs for generating synthetic data to improve the representation of minority classes and balance datasets, there is still a limited amount of research and work focused on assessing the effectiveness of GANs in enhancing the performance of machine learning models. In this project, the goal is to investigate the efficacy of GANs in balancing imbalanced datasets for text, image, and time series data. We will compare the performance of models trained on the original imbalanced datasets with models trained on the balanced datasets generated using GANs. Specifically, we will focus on enhancing the minority class in the datasets to evaluate the impact of synthetic data on the performance of models on under-represented classes. The results of this study will provide insights into the effectiveness of GANs in addressing the issue of imbalanced datasets and their potential applications in real-world scenarios.

2 RELATED WORK

GANs have gained significant attention in recent years for their ability to generate synthetic data that closely resembles the original data. Several studies have explored the use of GANs in generating synthetic data for imbalanced datasets. In particular, Conditional GANs have been used to introduce the concept of supervised learning in GANs by passing the labels along with the feature space.

[12]Ramponi G. et al., 2018, [14]Lei Xu et al., 2019, and [10]Giovanni Mariani et al., 2018 all used Conditional GANs and passed either the classification labels or the time steps as the conditional feature. This allowed the GAN networks to understand the temporal aspects of the data and improve the performance of models on under-represented classes.

[2]Christopher Bowles et al., 2018, and [16]Jinsung Yoon et al., 2019 both tested the ability of their model to generalize on other datasets by testing the proposed architecture on different datasets. Both studies showed that the approach could be generalized to other datasets, highlighting the potential of GANs in generating synthetic data for various applications.

[5]Fabio et al., 2019 and [1]Stavroula et al., 2021 both have a fixed standard dataset where different models are applied to check how different GAN architectures perform in generating synthetic data. In contrast, [7]Touseef et al., 2022 and Connor et al., 2019 explore different GAN architectures on different datasets to check if GAN models can be generalized. These studies demonstrate the potential of GANs in generating synthetic data that can be used as a substitute for real data. They also highlight the importance of evaluating the quality of synthetic data using a variety of metrics and increasing the diversity of generated data.

However, the ethical concerns of generating synthetic data have not been extensively discussed in these studies. It is important to consider the potential implications of using synthetic data and the potential consequences of relying solely on synthetic data in various applications.

3 DATASET

Four distinct datasets, including two datasets comprised of images, as well as one dataset for time series and another for text were utilized.

3.1 Tabular Data

The dataset used for tabular data is the Credit Card Fraud Detection dataset[4], which is a widely used dataset for analyzing fraudulent credit card transactions. The dataset contains credit card transactions made by European cardholders in September 2013 and has been collected from Kaggle. The dataset has a total of 30 features and 284,807 rows. It is an imbalanced dataset with only 492 positive class labels and 284,315 negative class labels. The dataset does not contain any missing or na values, but it does have 1081 duplicate values. All the features are integers, except for the 'Time' and 'Amount' features. The 'Class' feature is the response variable and has a value of 1 for fraud and 0 otherwise. Prior to its availability on Kaggle, the dataset underwent PCA transformation to obtain the most important features.The class distribution of tabular data is represented as a histogram in fig.1.

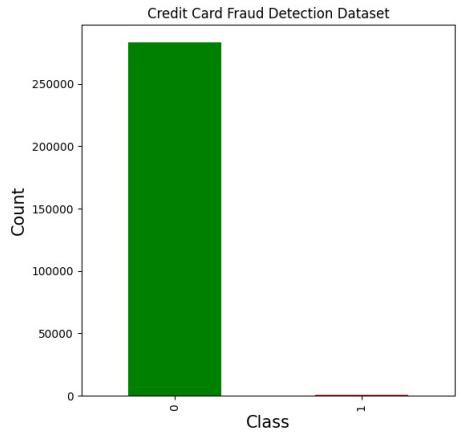

Figure 1: Credit Card Fraud Detection Dataset Distribution

3.2 Image Data

The Chest X-Ray image dataset[8] comprises high-resolution images of X-Ray scans of the lungs. The dataset includes a total of 5863 images that fall into two categories, Pneumonia and Normal. It is worth noting that this binary image dataset is imbalanced, with the Pneumonia category containing 4273 images, whereas the Normal category only has 1590 images. To exacerbate the imbalance, we down-sampled the minority class, resulting in only 402 Normal images.

The second dataset used is CIFAR-10 dataset[9], which is a popular benchmark dataset in machine learning and computer vision. It was created by the Canadian Institute for Advanced Research (CIFAR) and consists of 60,000 color images of size 32x32 pixels, each belonging to one of 10 classes. The classes are: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The dataset is split into 50,000 training images and 10,000 testing images, with equal distribution of the classes in both sets. The images were collected from a variety of sources and contain a wide range of object sizes, backgrounds, and orientations. The dataset is relatively small, making it an ideal choice for testing and benchmarking new machine learning algorithms. It is because of this reason, that we choose this already balanced dataset and create imbalance in it artificially. We select 5 classes and from the training set and downsample these classes to only contain 500 images each. Now the dataset is highly imbalanced. The classes airplane, bird, deer, frog and ship are the minority classes now. Prior to training, the images in these two datasets were normalized, and resized to meet the input requirements of the CNN architecture. The class distribution of image data is represented as a histogram in fig.2 and 3 .

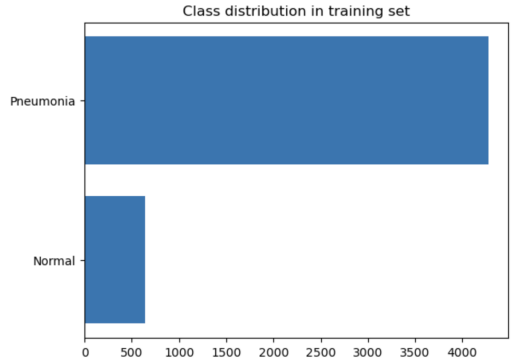

Figure 2: Chest X-Ray Pneumonia Dataset Distribution

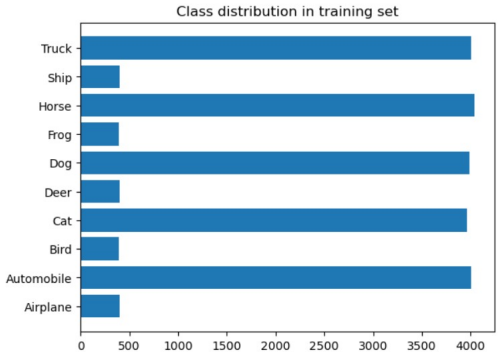

Figure 3: CIFAR-10 Dataset Distribution

3.3 Time series Data

The time series dataset used is called ECG5000[6]. It is a 20 hour long electrocardiogram (ECG) recording taken from a single patient and it has 140 data points where each data point is 10ms apart. The dataset has 4998 records and it has two classes: normal rhythm and abnormal rhythm. For people with normal rhythm, the ECG wave will follow a sinus rhythm. If the rhythm is abnormal, it can be due to various reasons most of which are dangerous to health. The data is fairly clean and it is preprocessed to extract each heartbeat

separately and made to equal length using interpolation. The class distribution of time series data is represented as a histogram in fig.4.

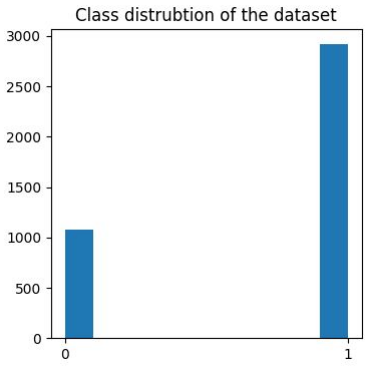

Figure 4: ECG5000 Dataset Distribution

4 METHODOLOGY

GAN is a deep learning architecture made up of two neural networks: a generator and a discriminator. The generator network creates synthetic data by getting noise as an input, while the discriminator network compares it to the actual dataset to determine if the created synthetic is real or fake. In an adversarial approach, the two networks are trained concurrently, with the generator attempting to generate synthetic data that is indistinguishable from actual data and the discriminator attempting to accurately categorize whether the input is genuine or fake. This repeated procedure continues until the generator generates data that the discriminator finds difficult to differentiate from genuine data.

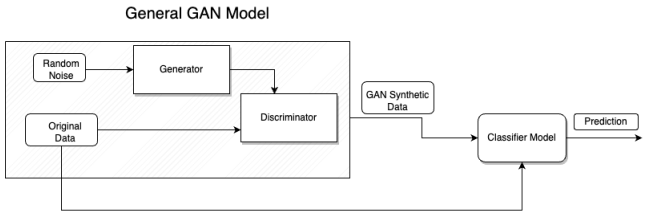

Figure 5: General Framework Implemented

We utilized a standardized approach to obtain our results, which involved using GANs to generate synthetic data and merging it with the imbalanced dataset to create a balanced dataset. This balanced dataset was then used to train the prediction models that were used in the baseline to generate the results. Fig.5 gives a visual depiction of the framework that was evaluated.

4.1 Techniques used to deal with tabular data

4.1.1 Baseline method. We investigate the performance of traditional tabular data classification models, namely random forest and decision tree with AdaBoost, on imbalanced datasets. Specifically, we aim to evaluate their performance on the minority class without applying any data augmentation techniques. To establish a reliable baseline, we conducted essential data preprocessing steps on the tabular dataset, such as removing duplicate values, scaling the data, and splitting it into training and testing sets. Our primary objective is to compare the performance of these classifiers with and without addressing the high class imbalance present in the data. We will analyze how the proposed techniques for handling class imbalance improve the models' performance in comparison to the traditional models.

4.1.2 GAN Based method. We use two types of GANs to deal with the class imbalance issue. We will first try to balance the data using these GAN models and then classify using the above models such as random forest and decision tree with Adaboost.

TableGAN is a Generative Adversarial Network (GAN) designed for generating synthetic data for imbalanced tabular datasets. It is an extension of the popular Deep Convolutional GAN (DCGAN) architecture, which is commonly used for image data. TableGAN comprises three neural networks: a generator (G), a discriminator (D), and a classifier (C). The generator produces synthetic records that have the same distribution as real records, while the discriminator distinguishes between the real and synthetic records. The addition of a classifier network helps maintain the consistency of values in the generated records, preventing inconsistencies such as a record with Age <=18 and CreditCardIssued = "Yes".

The classifier learns about consistency from the original table, and studies have shown that adding more classifiers can significantly improve the quality of the generated records. The generator and discriminator perform deconvolution or convolution operations to generate or classify a record. The final loss after the sigmoid activation is back-propagated to the generator. The dimensions of the latent vector input and intermediate tensors should be configured based on the number of attributes in the dataset. In tableGAN, where the generator generates T features one by one, and the discriminator concatenates all features together and uses a Multi-Layer Perceptron (MLP) with LeakyReLU activation function to distinguish real and fake data.

CTGAN is a GAN based approach designed to generate synthetic tabular data that closely resembles real-world tabular data. CTGAN models the conditional distribution of each column in the dataset given the values of the other columns, which helps to capture the complex interdependencies between the columns. The architecture of CTGAN can be seen in fig.6

To overcome the non-Gaussian and multimodal distribution in a dataset, CTGAN employs a unique mode-specific normalization feature. This is achieved by using the variational Gaussian mixture model (VGM) to estimate the number of modes and fit a Gaussian mixture for each continuous column Ci in the dataset. For each value ci,j in Ci, the probability of ci,j coming from each mode is computed using the weights and standard deviations of each mode. To normalize the value, CTGAN samples one mode from the given probability density and uses it to represent the value with a scalar α i,j. The representation of the row is then a concatenation of the continuous and discrete columns, as well as the one-hot vector β i,j indicating the selected mode.

To train the model, CTGAN uses a technique called training-by-sampling. This involves sampling the conditional and training data according to the log-frequency of each category, which ensures that CTGAN can evenly explore all possible discrete values and generate high-quality synthetic data for each category. For the neural network architecture of CTGAN, it utilizes two fully-connected hidden layers in both generator and critic. Batch-normalization and ReLU activation function are used to improve the performance of the generator, while Leaky ReLU function and dropout on each hidden layer are used in the critic to regularize the hidden layer activations and prevent overfitting.

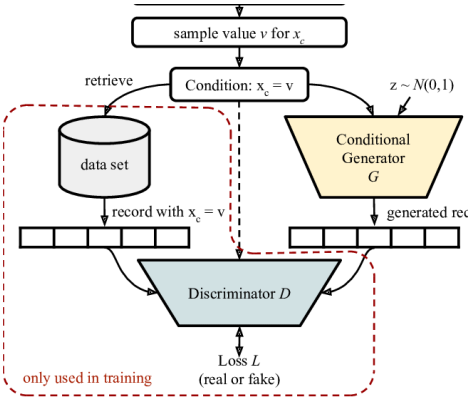

Figure 6: Architecture for CTGAN, source:[14]

4.2 Techniques used to deal with Image Data

4.2.1 Baseline method. Our aim is to evaluate the imbalanced binary dataset (X-ray Dataset) and the multi-class dataset(CIFAR-10) using two classifiers, without accounting for the imbalance in them. We use this method as it establishes a baseline, which could be compared against the future methods. We use two CNNs to train the datasets. One is a custom desgined CNN. The other network is the EfficientV2B0 whcih is trained using transfer learning. The datasets are split into train, val and test subsets. The performace of the two networks on the datasets are evaluated using metrics such as precision, recall and F1-score.

4.2.2 Traditional Image Augmentation. Traditional augmentation methods include Random rotation, Random Zoom, Horizontal flipping, Vertical Flipping, Rescaling etc. These augmentation techniques are used to increase the variety and number of samples in the training set, so that the model can learn from a diverse set of data. This also helps us in balancing the class distribution by generating more samples for the underrepresented classes. The models trained using this method are then evaluated on the test data and is also evaluated based on specific metrics.

4.2.3 GAN Based Augmentation. In this method, we plan of using GANs to generate synthetic image data to balance the imbalanced data. We plan to use Balancing GAN (BAGAN). BAGAN is a type of generative adversarial network (GAN) that is designed to balance imbalanced datasets. It extends the traditional GAN architecture by adding a balance constraint to the generator and discriminator networks, which helps to ensure that the generated samples are balanced across the different classes in the dataset.

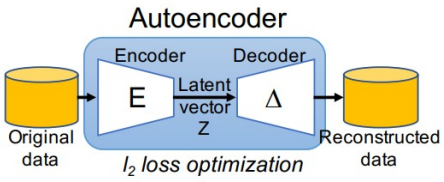

Figure 7: Autoencoder training, source:[10]

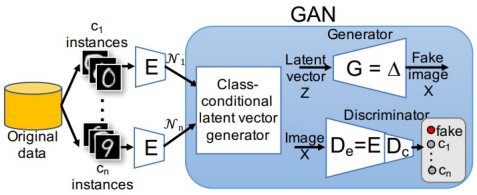

Figure 8: GAN initialization, source:[10]

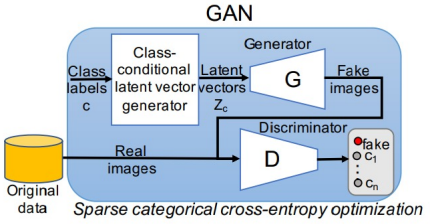

Figure 9: GAN training, source:[10]

The BAGAN training approach consists of three steps: autoencoder training (Fig.7), GAN initialization (Fig.8), and adversarial training (Fig.9). The autoencoder is trained using all images in the training dataset, while the generator and discriminator in the GAN have explicit class knowledge. The generator is initialized with the weights of the autoencoder decoder, and the discriminator is initialized with the weights of the encoder. The class-conditional latent vector generator is also initialized with probability distributions matching the distribution of latent vectors for each class. During adversarial training, the generator and discriminator are fine-tuned by processing batches of real and fake images, with the generator outputting images based on the class-conditional latent vectors drawn at random from the generator. The training aims to optimize the sparse categorical cross-entropy loss function to match the class labels for real images and the fake label for generated ones.

4.3 Techniques used to deal with Time Series Data

4.3.1 Baseline without augmentation. Initially, without accounting for the imbalance in the dataset, the models are trained to get a baseline value. The data is cleaned to make sure that there are no missing/duplicate values in the dataset. Then the data is split into train and test sets and passed onto to the model for learning. To evaluate the base dataset without any augmentation, A custom LSTM (Long Short Term Memory) model is used. The LSTM network has two LSTM cells with 50 units each, followed by a dense layer. These results are saved and later compared with the same models on dataset with augmented data.

4.3.2 Using GANs for Augmentation. To augment time series data, TimeGAN and T-CGAN. In TimeGAN, a supervised loss function is used which uses the original data and captures the stepwise conditional distributions in the data. This utilizes all the information that can be obtained from the dataset other than just the time series features. Fig.10 shows the architecture diagram for the TimeGAN model. Additionally, they convert the data into latent representations. This latent representation helps ease the training process by reducing the number of dimensions of the original data while still capturing the distribution of the data. Converting the data to latent representation helps in improving the performance of the model by computing for values in a lower dimension. In T-CGAN, Conditional GANs are modified to work on Time series data. Here the GANs are conditioned on the time step. Fig.11 shows the architecture diagram for the T-CGAN model.In Generator network, a CNN is used which takes the time series data along with the time steps. In the discriminator network, the real data, augmented data and the timesteps are passed and a binary value is given as output which indicates whether the data is real or synthetic. The discriminator is also a convolutional network with a full connected layer at the end.

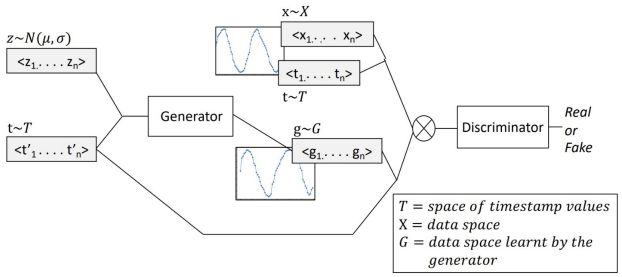

Figure 10: Architecture for TCGAN, source: [12]

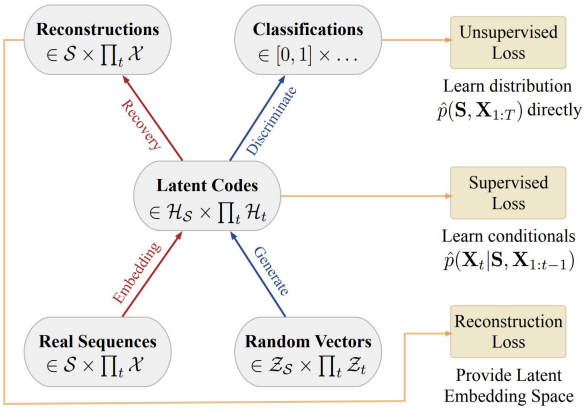

Figure 11: Architecture for Time GAN, source: [16]

5 IMPLEMENTATION

5.1 Tabular Data

For RandomForest ensemble method, we first import the RandomForestClassifier class from scikit-learn's ensemble module and the GridSearchCV class from scikit-learn's model selection module. We then define the hyperparameters we want to tune in a dictionary called params. Next, we create a RandomForestClassifier object with a random state of 0. We also create a GridSearchCV object with the RandomForestClassifier object, the params dictionary, a cross-validation of 5. The best hyperparameters found in this example are n estimators as 100, max depth as 8, max features as auto, and criterion as gini. This means that the random forest will consist of 100 decision trees, each with a maximum depth of 8, and the number of features considered at each split will be chosen automatically by the model. The gini criterion will be used to measure the quality of each split.

For the AdaBoost classifier using a decision tree, we define the base classifier, which in this case is a decision tree classifier. We then define the boosting classifier with the AdaBoostClassifier function and set the base estimator parameter to the decision tree classifier. We also set the n estimators parameter to the number of estimators as 100, and the learning rate parameter to control the contribution of each classifier. After defining the boosting classifier, we train it on the training data and labels using the fit method. Finally, we use the predict method to make predictions on the test data and evaluate the performance of the model using the classification report function, which provides metrics such as precision, recall, and F1-score for each class.

For the implementation of CTGAN model on the credit card dataset. We utilized the SDV library to generate synthetic data. The data preprocessing steps include removing duplicate rows and converting the class column to binary values. The metadata of the dataset is then extracted using the SingleTableMetadata function from the SDV library and validated to ensure it meets the required specifications. The CTGANSynthesizer function is then used with parameters such as metadata, number of epochs (100), and cuda to train the synthesizer on the dataset. The fit() function is then called to start the training process. Next, a Condition is created to specify the number of rows and column values that the synthetic data should match. A new synthetic data is created by sampling from the synthesizer using the samplefromconditions() function with the created condition. The synthetic data is then concatenated with the original data and passed to the baseline models to check the evaluation metrics.

We wrote a code from scratch for tableGAN referring to the paper [11]. The implementation process involves creating a TableGAN model to generate synthetic data using only the fraud data as input. The first step is data preprocessing, where the input data is prepared. The PowerTransformer method is used to transform the distribution of the input data into a Gaussian distribution. Next, the TableGAN model is defined. The generator architecture consists of four dense layers with LeakyReLU activation and the last layer with the data dimension as output. The discriminator architecture has three dense layers with LeakyReLU activation and one output layer with sigmoid activation. The hyperparameters of the model include a noise dimension of 256, a hidden layer dimension of 128, a batch size of 16, a learning rate of 0.0005, 401 epochs, and a logging step of 100. After defining the model, only the fraud data is given as input to the model. The TableGAN takes the fraud data as input and generates synthetic data as output. Overall, the implementation process involves data preprocessing, defining the TableGAN model, and generating synthetic data using the fraud data as input. The synthetic data is then concatenated with the original data and passed to the baseline models to check the evaluation metrics.

5.2 Image Data

The custom convolutional neural network's architecture consists of several layers, including convolutional layers with 32, 64, and 128 filters of size (3, 3), batch normalization layers, and max pooling layers with pool size (2, 2). Dropout layers with a dropout rate of 0.25 are added after each max pooling layer to prevent overfitting. The final layer is a fully connected layer with 128 neurons and a ReLU activation function, followed by a dropout layer with a rate of 0.25. The output layer is a dense layer with 10 neurons and a softmax activation function, representing the 10 possible classes that the images can be classified into. Transfer learning is used on EfficientNetV2B0. All the layers are made trainable. The output of the base model is flattened and is followed by a fully connected layer with 1024 neurons, followed by another layer with 10 neutrons and a softmax activation function. For CIFAR-10 dataset, both the models use "Adam" optimizer and the categorical cross entropy function with a learning rate of 0.001. Both the networks are trained on the multi-class dataset with 50 epochs in batches of 32. Similarly for Chest X-Ray dataset, both the models use "Adam" optimizer and the binary cross entropy function. Both the networks are trained on the binary dataset with 15 epochs in batches of 32. For generating synthetic data using BAGAN, we train the BAGAN model with the imbalanced data for 250 epochs. We do this for both the datasets and generate samples of minority classes, such that adding them will reduce the imbalance in the dataset.

5.3 Time Series Data

The Recurrent Neural Network (RNN) classification model employs several performance metrics, including Accuracy, Precision, Recall, and F1 Score, to evaluate its performance. The Long Short-Term Memory (LSTM) model consists of two LSTM cells, each comprising 50 units. Adam optimizer is used to optimize the model, and categorical cross-entropy loss function is employed to evaluate the classification model's performance. These techniques aid in assessing the accuracy and reliability of the LSTM model in performing classification tasks. Both the baseline dataset and the data augmented dataset. The TimeGAN network is trained for 10 epochs with learning rate 0.0001 and the network is optimized using Adam optimizer. The TCGAN network is trained for 20 epochs with learning rate of 0.001 and optimized using Adam optimizer.

6 RESULTS

6.1 Tabular Data

The study conducted a comparison between baseline raw data and GAN generated data to evaluate the performance of the model on minority classes. The results indicated in Table 1 show that the GAN-based models performed better on the minority classes. The baseline model had good precision and recall scores for the majority class, but the scores were relatively low for the minority class. The f1 scores were also lower for the minority class compared to the perfect score for the majority class. TableGan generated synthetic data did not perform well, particularly with the minority class, as compared to the baseline model. However, CTGAN generated synthetic data helped the model perform accurately. The precision, recall, and f1 scores for both the majority and minority classes were perfect. In summary, the study demonstrated that GAN-based models are effective in improving the performance of the model on minority classes. CTGAN generated synthetic data, in particular, helped to enhance the model's accuracy, whereas TableGan did not perform well in this regard. These findings suggest that using GANs for generating synthetic data could be a promising approach to address the challenge of imbalanced data in machine learning.

6.2 Image Data

Table 2 and 3 shows the performance of two classifiers, Custom CNN and EfficientNetV2, was evaluated on imbalanced CIFAR-10 and Chest X-Ray datasets after training them for 50 and 15 epochs respectively. On CIFAR-10, Custom CNN achieved an accuracy of 70%, while EfficientNetV2 produced an accuracy of 50%. The F1-score obtained for Custom CNN was 0.69, and for EfficientNetV2, it was 0.51. Upon applying traditional data augmentation, Custom CNN showed a moderate improvement with 73% accuracy and an F1-score of 0.71, while EfficientNetV2 produced an accuracy of 49% and an F1-score of 0.43. However, using GAN for data augmentation significantly improved the performance of both models. Custom CNN achieved an accuracy of 76% and an F1-score of 0.79, while EfficientNetV2 produced an accuracy of 69% and an F1-score of 0.68. In conclusion, the use of GAN-based augmentation resulted in the best performance for both classifiers.

For the Chest X-ray dataset, the Custom CNN classifier achieved an accuracy of 70% and an F1-Score of 0.57, while EfficientNetV2 achieved an accuracy of 80% and an F1-Score of 0.75. By applying traditional data augmentation techniques, Custom CNN's accuracy improved to 86% with an F1-Score of 0.85, while EfficientNetV2's accuracy remained at 80% with an F1-Score of 0.75. However, when GAN-based augmentation was used, the performance of both classifiers decreased. Custom CNN achieved an accuracy of only 71% and an F1-Score of 0.6, while EfficientNetV2 achieved an accuracy

of 79% and an F1-Score of 0.75. This could be attributed to the complexity of the Chest X-Ray dataset, which was not fully captured by the GAN model within the given number of epochs.

In Fig.12, the confusion matrix indicates that there is a high number of false classifications of minority samples, while the majority class samples were mostly classified correctly. Fig.13 presents the results of Custom CNN trained on traditionally augmented data, where the performance on majority class samples is satisfactory, but the number of correctly classified minority class samples, although better than the previous one, remains low. However, in Fig.14, the Custom CNN trained on GAN augmented data produces significantly better results on minority classes, while maintaining good performance on the majority classes.

6.3 Time series Data

After generating synthetic data and running the classifiers on the new balanced dataset, the results obtained are shown in 4. While both augmentation techniques performed better than the baseline, TimeGAN gave the highest improvement. Average f1 score went from 0.64 to 0.81 which is a significant improvement. The recall value for minority class shows that there is a 60% improvement over the baseline.

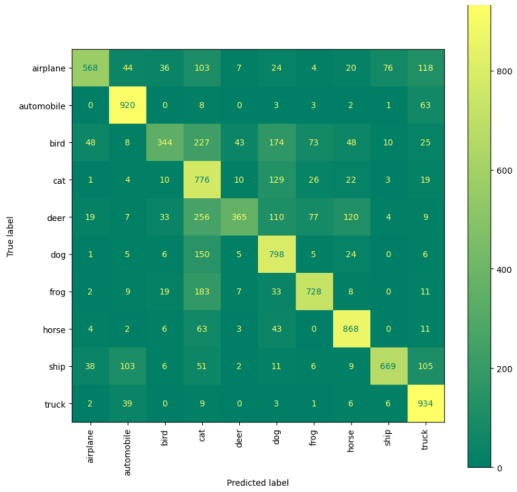

Figure 12: Confusion matrix of Custom CNN

7 CONCLUSION/FUTURE WORK

GANs for data augmentation have shown considerable potential in enhancing the generalization performance of machine learning models by creating synthetic data that improves minority class representation and balances the dataset. Various performance metrics were used in this project to evaluate the effectiveness of cutting-edge GANs in data augmentation, and the results indicate that incorporating synthetic data generated by GANs into the dataset results in a more balanced distribution, leading to significant improvements in model performance. The study discovered that when GANs were used to produce synthetic data, text-based data improved significantly.

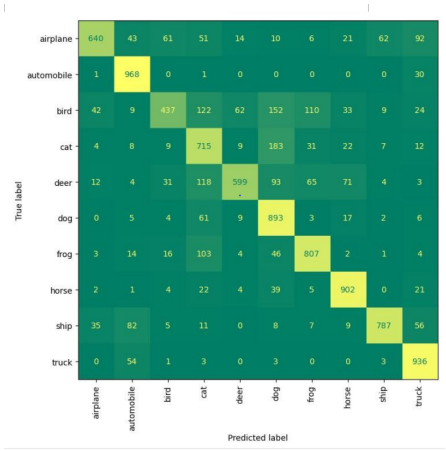

Figure 13: Confusion matrix of Custom CNN with Data Augmentation

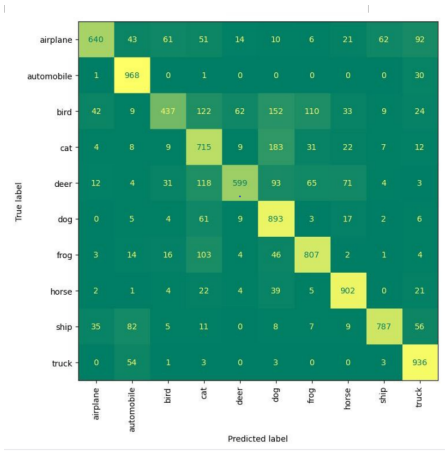

Figure 14: Confusion matrix of Custom CNN with Balanced GAN

Despite GANs' notable success in capturing the underlying distribution of datasets, their inability to generate completely novel data remains a limitation. GANs are also computationally costly, making their deployment in large-scale datasets difficult. Future research might concentrate on overcoming these limits by constructing more efficient GAN structures capable of producing fully unique data while reducing computing cost. Further research could look into the potential of GANs in areas other than image, text, and time series datasets, such as graph or tabular data. Finally, assessing the effectiveness of GANs in conjunction with other data augmentation techniques could be a promising area of future research.

REFERENCES

[1] Stavroula Bourou, Andreas El Saer, Terpsichori-Helen Velivassaki, Artemis Voulkidis, and Theodore Zahariadis. 2021. A Review of Tabular Data Synthesis Using GANs on an IDS Dataset. *Information* 12, 9 (2021). https://doi.org/10.3390/info12090375

[2] Christopher Bowles, Liang Chen, Ricardo Guerrero, Paul Bentley, Roger Gunn, Alexander Hammers, David Alexander Dickie, Maria Valdés Hernández, Joanna

Table 1: Results on Credit card dataset

		Accuracy	Precision		Recall		f1-score	
			0	1	0	1	0	1
Baseline	Bagging (Random forest)	1.00	1.00	0.97	1.00	0.8	1.00	0.88
	Boosting (DT and Ada Boost)	1	1.00	0.71	1.00	0.78	1.00	0.74
TableGAN	Bagging	1.00	1.00	0.71	1.00	0.95	1.00	0.81
	Boosting	1.00	1.00	0.71	1.00	0.95	1.00	0.81
CTGAN	Bagging	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Boosting	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table 2: Results on CIFAR-10 dataset

		Accuracy	Precision	Recall	f1-score
Baseline	Custom CNN	0.70	0.74	0.70	0.69
	EfficientNetV2B0	0.50	0.60	0.5	0.51
Data Augmentation	Custom CNN	0.73	0.76	0.72	0.71
	EfficientNetV2B0	0.49	0.65	0.46	0.43
BalancedGAN	Custom CNN	0.76	0.79	0.76	0.76
	EfficientNetV2B0	0.69	0.73	0.69	0.68

Table 3: Results on Chest X-Ray dataset

		Accuracy	Precision	Recall	f1-score
Baseline	Custom CNN	0.70	0.82	0.60	0.57
	EfficientNetV2B0	0.80	0.86	0.73	0.75
Data Augmentation	Custom CNN	0.86	0.86	0.84	0.85
	EfficientNetV2B0	0.80	0.86	0.73	0.75
BalancedGAN	Custom CNN	0.71	0.80	0.62	0.60
	EfficientNetV2B0	0.79	0.81	0.74	0.75

Table 4: Results on ECG5000 dataset

	Accuracy	Precision		Recall		f1-score	
		0	1	0	1	0	1
Baseline	0.64	0.48	1.00	1.00	0.47	0.65	0.64
C-TGAN	0.73	0.62	0.99	1.00	0.54	0.76	0.70
TimeGAN	0.88	0.80	1.00	1.00	0.77	0.83	0.80

Wardlaw, and Daniel Rueckert. 2018. GAN Augmentation: Augmenting Training Data using Generative Adversarial Networks. arXiv:1810.10863 [cs.CV]

[3] Luis M. Candanedo, Véronique Feldheim, and Dominique Deramaix. 2017. Data driven prediction models of energy use of appliances in a low-energy house. *Energy and Buildings* 140 (2017), 81–97. https://doi.org/10.1016/j.enbuild.2017.01.083

[4] Andrea Dal Pozzolo. 2016. Credit Card Fraud Detection. https://www.kaggle.com/mlg-ulb/creditcardfraud. Accessed: 2023-04-13.

[5] Fabio Henrique Kiyoiti dos Santos Tanaka and Claus Aranha. 2019. Data Augmentation Using GANs. *CoRR* abs/1904.09135 (2019). arXiv:1904.09135 http://arxiv.org/abs/1904.09135

[6] A L Goldberger, L A Amaral, L Glass, J M Hausdorff, P C Ivanov, R G Mark, J E Mietus, G B Moody, C K Peng, and H E Stanley. 2000. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. *Circulation* 101, 23 (June 2000), E215–20.

[7] Touseef Iqbal and Shaima Qureshi. 2022. The survey: Text generation models in deep learning. *Journal of King Saud University - Computer and Information Sciences* 34, 6, Part A (2022), 2515–2528. https://doi.org/10.1016/j.jksuci.2020.04.001

[8] Daniel Kermany, Kang Zhang, and Michael Goldbaum. 2018. Chest X-Ray Images (Pneumonia). https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia

[9] Alex Krizhevsky. 2009. Learning Multiple Layers of Features from Tiny Images. arXiv:tinyimages [cs.CV] https://www.cs.toronto.edu/~kriz/cifar.html CIFAR-10 dataset.

[10] Giovanni Mariani, Florian Scheidegger, Roxana Istrate, Costas Bekas, and A. Cristiano I. Malossi. 2018. BAGAN: Data Augmentation with Balancing GAN. *CoRR* abs/1803.09655 (2018). arXiv:1803.09655 http://arxiv.org/abs/1803.09655

[11] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. 2018. Data synthesis based on generative adversarial networks. *Proceedings of the VLDB Endowment* 11, 10 (jun 2018), 1071–1083. https://doi.org/10.14778/3231751.3231757

[12] Giorgia Ramponi, Pavlos Protopapas, Marco Brambilla, and Ryan Janssen. 2019. T-CGAN: Conditional Generative Adversarial Network for Data Augmentation

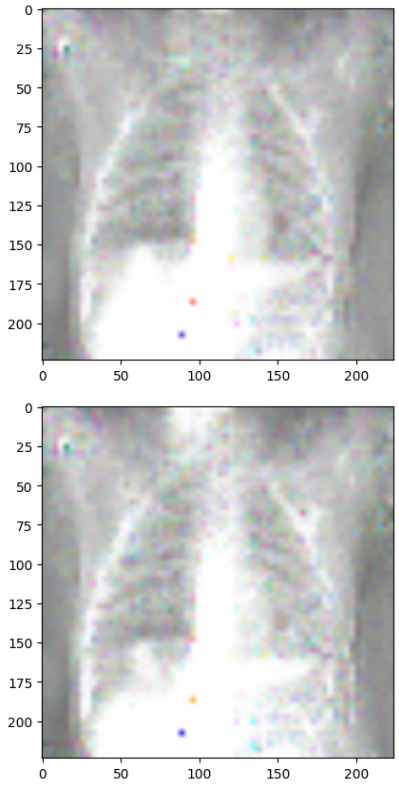

Figure 15: BAGAN generated images

in Noisy Time Series with Irregular Sampling. arXiv:1811.08295 [cs.LG]

[13] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. 2018. *Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization.* https://doi.org/10.1007/978-3-319-95729-6_13

[14] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. Modeling Tabular data using Conditional GAN. *CoRR* abs/1907.00503 (2019). arXiv:1907.00503 http://arxiv.org/abs/1907.00503

[15] Lei Xu and Kalyan Veeramachaneni. 2018. Synthesizing Tabular Data using Generative Adversarial Networks. arXiv:1811.11264 [cs.LG]

[16] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. 2019. Time-series Generative Adversarial Networks. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2019/file/c9efe5f26cd17ba6216bbe2a7d26d490-Paper.pdf