2021

Improving a Network Intrusion Detection System's Efficiency Using Model-Based Data Augmentation

Vinicius Waterkemper Lodetti
Technological University Dublin

Recommended Citation

Lodetti, V. (2021). Improving a Network Intrusion Detection System's Efficiency Using Model-Based Data Augmentation. Technological University Dublin. DOI: 10.21427/EBCG-WS30

Improving a Network Intrusion Detection System's Efficiency Using Model-Based Data Augmentation

Vinicius Waterkemper Lodetti

A dissertation submitted in partial fulfilment of the requirements of

Technological University Dublin for the degree of

M.Sc. in Computing (Data Science Stream)

March 2021

Declaration

I certify that this dissertation which I now submit for examination for the award of MSc in Computing (Data Science stream), is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Technological University Dublin and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institute's guidelines for ethics in research.

Signed:Vinicius W. Lodetti

Date:03/March/2021

Abstract

A network intrusion detection system (NIDS) is one important element to mitigate cybersecurity risks, the NIDS allow for detecting anomalies in a network which may be a cyberattack to a corporate network environment. A NIDS can be seen as a classification problem where the ultimate goal is to distinguish between malicious traffic among a majority of benign traffic. Researches on NIDS are often performed using outdated datasets that don't represent the actual cyberspace. Datasets such as the CICIDS2018 address this gap by being generated from attacks and an infrastructure that reflects an up-to-date scenario.

A problem may arise when machine learning classification algorithms are trained on a dataset that presents class imbalance towards a majority, which is the case of CICIDS2018 data where the majority class is skewed to legitimate traffic. Such problem can be tackled by modifying a dataset probability distribution by augmenting the existing data to achieve balance in the dataset. Many different methods can be used to do so, ranging from naive approaches like random oversampling or undersampling; Machine learning with SMOTE and Decision Trees; Or even sophisticated deep learning models such as the GAN and CTGAN.

An evaluation of the different data-augmentation methods for training a random forest classifier task showed that ROS and SMOTE are competitive in detecting attacks, while CTGAN demonstrated to better recognize benign samples and provide a balance between security and functionality for the network, however at a computational resource expense.

Keywords: CICIDS2018, Network Intrusion Detection System, Imbalanced Learning, Data Augmentation, Deep Learning

Acknowledgments

I would like to thank Dr. Jack O'Neill for the endless support, insights, motivation and guidance that allowed me to produce such research.

A big thanks to my flatmates, family and especially to my partner that were always there to support me.

Contents

List of Figures

List of Tables

List of Acronyms

CART	Classification and Regression Trees
CTGAN	Conditional Tabular Generative Adversarial Neural Network
DDoS	Distributed Denial of Service
FN	False Negative
FP	False Positive
GAN	Generative Adversarial Neural Network
IDS	Intrusion Detection System
IPS	Intrusion Protection System
NIDS	Network Intrusion Detection System
RF	Random Forest
ROS	Random Oversampling
RUS	Random Undersampling
SMOTE	Synthetic Minority Over-sampling Technique
TN	True Negative
TP	True Positive
WGAN	Wasserstein Generative Adversarial Neural Network
WGAN-GP	Wasserstein Generative Adversarial Neural Network with Gradient Penalty
VA	Variational Autoencoder

Chapter 1

Introduction

An anomaly-based network intrusion detection system looks for malicious traffic patterns in a network, these malicious samples will be a minority regarding the legitimate traffic produced by business softwares, internet browsing or any other non-malicious network activity. A NIDS can be implemented as a classification algorithm to distinguish between malicious and benign traffic, but since minority of the traffic is malicious, the classifier predictions may be biased towards the benign samples. This problem is known as imbalanced learning or class imbalance problem.

1.1 Background

Cyber threats are a risk for the many business that rely on computer networks — such as the internet — to perform day-to-day operations. These networks have many benefits; they allow for integrate with other companies which potentially increases business opportunities, market availability, and ultimately, revenue. On the other hand, being part of a network also increases information security risks capable of doing financial and reputation damage or even shutting down an entire company.

Technologies such as a network intrusion detection system (NIDS) can be placed between the internet connection or in front of a company's most risky assets to detect, prevent and contain internal and external attacks. A NIDS can detect malicious network traffic in two ways, *signature-based* by comparing traffic to known attack patterns and *anomaly-based* for unknown attacks such as never seen before.

Network attacks generally follow similar patterns, often scripts and programmes available to public use. New attacks are being launched and discovered every day, signature-based detection requires regular maintenance on the NIDS signature database[1], obsolete signatures should also be deleted to assure a NIDS performance, by the other hand signature-based detection is less susceptible to False Positives.

Zero-day are attacks which haven't yet been seen and are immune to pattern-based recognition, worse still known attacks may be altered slightly to avoid detection. Machine learning models can be taught to recognize these anomalous patterns, anomaly-based detection looks for sudden unexplained differences in network traffic, meaning it can detect previously unseen patterns. The attack detection can be performed using different methods such as artificial neural networks, Bayesian nets, clustering, decision trees, ensemble models, support vector machines, association and fuzzy rules. (Buczak & Guven, 2016; Biswas, 2018).

It is expected that the major part of a company network traffic to be benign and ideally the NIDS must not interfere with it, which could impact operations by for example, disrupting critical business applications availability. However, machine learning models can have their prediction ability negatively impacted when trained on skewed data, thus favouring the most occurring class, this problem is also known as imbalanced data or class-imbalance problem (Weiss et al., 2007; Parkinson de Castro, 2020).

A solution to the imbalanced data problem is to increase the number of minority class samples, for example hiring an external company to exhaustively attack a network, which implies in costs and can impact business availability. That's where synthetic data generation can be an approach to achieve class balance and improve NIDS performance, ranging from more basic implementations such as random oversampling (ROS) and undersampling (RUS), to machine-learning based on k-nearest neighbours and decision tree like Synthetic Minority Over-sampling Technique (SMOTE) and Classification and Regression Trees (CART) respectively, to more advanced tech-

[1] https://www.ciscopress.com/articles/article.asp?p=133642

niques making use deep generative models such as Generative Adversarial Networks (GAN) and its derivatives like the Conditional Tabular GAN (CTGAN).

Goodfellow et al. (2014) proposed the GAN, a generative model trained in an adversarial way, the architecture consists of two Neural Networks, the generator and discriminator trained simultaneously while competing a min-max game where the generator objective is to maximize the discriminator loss and, the discriminator objective to minimize his own loss. The adversarial training can be exemplified as the generator being an art forger and the discriminator an detective specialized in art, at beginning the generator will reproduce the real arts in a very poorly manner and the discriminator will classify the generator work as fake, then eventually the generator will improve based on the discriminator's feedback. The discriminator will spot and exploit weaknesses in the generated samples, the generator is then forced to overcome that weaknesses to produce better samples. This training should continue until the discriminator is no longer able to distinguish between real and fake/generated samples, that's when the min-max loss is achieved. This website [2] shows pictures of people that doesn't actually exist, they are all generated from a type of GAN (Karras et al., 2019).

1.2 Research Project/problem

Companies must protect themselves against cyber-attacks to avoid reputational and financial damage, fines from compliance breaches and leakage of trade secrets[3]. A NIDS is one of the components to assure network security, it should prefer to misclassify benign traffic as malicious rather than malicious as benign, thus defining the main metrics as recall, precision and F1-Score. The metrics should be maximized, if recall is low it means attacks are not being detected, while lower precision will result in a security analyst wasting time reviewing false positives, the F1-Score will give a balanced score between recall and precision.

A NIDS can be implemented as a classification model to distinguish between mali-

[2] https://thispersondoesnotexist.com

[3] https://www.nytimes.com/2018/07/20/business/suppliers-data-leak-automakers.html

cious and benign traffic, major part of a company network traffic should be benign — so the traffic used to teach the NIDS — however machine learning models may bias its predictions towards the most common class, which is benign traffic. One approach to solve this bias/imbalanced learning problem is to increase the number of a minority class on the training data of a classification model.

One of the simplest ways of dealing with class imbalance is by randomly copying samples from the minority class into the dataset, or even randomly removing samples from the majority class. SMOTE and CART are also two methods to deal with class imbalance by augmenting the minority class in a machine learn fashion.

GANs have successfully generated synthetic images and tabular data but are known to be difficult to train (Arjovsky et al., 2017), approaches like Wasserstein GAN (WGAN), WGAN with Gradient Penalty (WGAN-GP) and CTGAN were designed to address known issues from the GAN such as mode collapse, vanishing gradients and non-convergence, further explained in section 2.6.1.

These deep generative methods will be explored as a synthetic data generator to achieve class balance in a classifier model. The goal of this project is to evaluate to what extent the different data generation methods will improve the NIDS recall, precision and F1-Score.

1.3 Research Objectives

The motivation behind including the deep learning approaches in the tests is that the GAN and its derivatives seems promising generator models given their ability to generate high quality images. Rigaki and Garcia (2018) have used a GAN to adapt malware to avoid being detected by a NIDS. Although it is necessary to simulate an attacker capability against a target, the defence for such state-of-the-art offensives must be planned. In how many other ways could such innovative use of GAN concepts contribute to the protection of the cyberspace?

The CICIDS2018 (Sharafaldin. et al., 2018) is a benchmark dataset for network intrusion detection systems. It was collected during a simulation of a company being attacked. This fictional organization has 420 workstations and 30 servers among 5 departments, while the attacking infrastructure is made of 50 computers.

In a real company network major part of the traffic is benign and a minority would be malicious, that's also reflected in the CICIDS2018. A classifier trained on imbalanced data can have its predictions biased towards the majority class, and may not detect malicious traffic to an acceptable degree. Although the dataset is imbalanced, with 87% instances belonging to the negative class (benign); correctly classifying the minority class (attack) is of great importance to NIDS. A false negative, allowing malicious traffic is potentially catastrophic to a network. While a false positive, would imply in inadvertently blocking benign traffic, usually causing minor inconveniences.

Modifying the training dataset distribution by augmenting the malicious samples is one way to workaround the imbalanced learning issue. This experiment will compare the performance of a NIDS implemented as a random forest (RF) classifier trained on CICIDS2018, with different methods of data-augmentation such as ROS, RUS, SMOTE, CART, GAN and CTGAN, these methods will be evaluated regarding their precision, recall and F1-Score.

It is important that the generated samples follow the same distribution and feature-interdependencies of the real data, leading to **sub-question 1 :** To what extent are the synthetic generated samples capable of capturing the original data distribution and correlations.

A random forest will distinguish between benign and malicious traffic regardless of which attack type the malicious traffic represents, leading to **sub-question 2 :** Will the data augmentation methods influence on the misclassification of each attack type?

Research objective: Compare the effect of adding GAN, CTGAN, ROS, RUS, CART or SMOTE generated samples to augment the minority classes of CICIDS2018 dataset to train a random forest classifier model in terms of model recall, precision and F1-score.

Research question: To what extent adding synthetic generated samples to augment the minority classes of CICIDS2018 dataset to train a random forest classifier model is capable of improving the model precision, recall and F1-Score.

1.4 Research Methodologies

This project is a secondary research since it relies on network flow traffic data collected by (Sharafaldin. et al., 2018) and available on Amazon[4], they developed a benchmark dataset with the latest attacks in a simulated organization, the CICIDS2018 allows building and evaluation of a NIDS on this empirical experiment to answer the research questions and hypothesis. Different NIDS models implemented as Random Forest classifier will be trained on the original data which is imbalanced and also on data balanced with samples generated from ROS, RUS, SMOTE, CART, GAN and CTGAN. Recall, precision and F1-Score of such models will then be compared, analysed and discussed to answer the research question.

1.5 Scope and Limitations

The scope of this research is the creation of synthetic attack samples to address the class imbalance issue present on CICIDS2018. Only a part of the dataset will be used, further details are given in section 2.1, this reduced dataset will decrease experiment time and facilitate GAN training which is known to be difficult and computationally expensive.

This research will not focus on the creation or tuning of classification models for the NIDS, it will use Random Forests (RF) trained on different data augmentation techniques to distinguish between benign or malicious network traffic, the different RF will then be evaluated regarding their recall, precision and F1-Score.

This work scope is dealing with imbalanced learning at data-level only, which implies in modify the distribution of the training dataset, such as adding samples to the training. The methods to deal with imbalanced learning at data-level are limited here to ROS, RUS, SMOTE, CART, GAN and CTGAN. Not in the scope of this work, but imbalanced learning can also be tackled at algorithm-level by for example setting weights to the classes during classifier training or setting decision thresholds on a classifier prediction.

[4] s3://cse-cic-ids2018/

Signature-based detection is out of the scope together with hybrid-based detection which is a mix of both anomaly and signature-based detections, although in a production scenario the is more likely a NIDS would use a hybrid-based approach. Host-based intrusion detection is also out of scope, it consists for example in analysing usage of computational resources such as memory, processor, storage, processes and network usage from an operating system or virtualizer perspective. This research will focus on network-based intrusion detection systems only.

1.6 Document Outline

The following chapters of this dissertation are outlined below

Chapter 2: This chapter presents an overview of the existing literature and discussions with respect to the imbalanced learned problem, how other research has dealt with such, and what are the opportunities for improvement in this area.

Chapter 3: Details the proposed experiment framework designed to answer the hypothesis and research questions towards the project objective.

Chapter 4: Presents an actual implementation of the proposed experiment, what went well, limitations, issues, workarounds and outcomes that allow to draw conclusions regarding the research.

Chapter 5: This chapter summarizes the results, observations and insights of the findings, with potential extension for this research.

Chapter 2

Review of existing literature

A NIDS is an important piece to mitigate information security risks, a minority of the traffic analysed by a NIDS would be malicious. The imbalance between minority and majority class — respectively malicious and benign — can bias a NIDS decision to the predict most frequent class. This research focuses on solving the class imbalance problem by investigating the effect of applying different data-augmentation methods to an imbalanced dataset.

In this chapter, it is explained the dataset used in the proposed experiment, the problems underlying imbalanced data and how other research dealt with similar problems. The data-augmentation techniques employed range from Naive approaches to sophisticated deep learning algorithms, where some techniques were applied to similar datasets by other research. The implementation of a NIDS as a binary classification problem using a Random Forest. Finally, the evaluation metrics, where conclusions will be based on.

2.1 Dataset

Researchers often use outdated network intrusion datasets such as DARPA, KDD-CUP and NSL-KDD (based on KDD-CUP), from 1998, 1999 and 2006 respectively (Khraisat et al., 2019), the internet has changed a lot since then and so have the cyber threats. Newer datasets such as the CICIDS2018 contemplate more recent attack types and techniques often seen today. The CICIDS2018 is part of a project that has created other IDS benchmark datasets[1] in the past such as the ISCXIDS2012 (Shiravi et al., 2012) and the CICIDS2017 (Sharafaldin et al., 2019).

The project creates datasets using a systematic approach based on profiles to generate benign (B-Profile) and malicious (M-Profile) traffic. B-Profile is composed of data generated from users or from the CIC-BenignGenerator (Sharafaldin et al., 2017) and M-Profile is derived from various attack types such as password guessing, web application exploitation, Distributed Denial of Service (DDoS), internal and external port scanning. The CICIDS2018 is available 10 different files in CSV format, which was parsed to a network flow format using CICFlowMeter[2] (Lashkari. et al., 2017; Draper-Gil. et al., 2016) from PCAP files, a standard format of low-level network traffic capture at interface level.

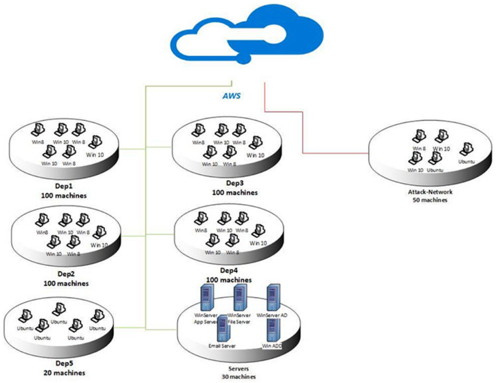


Figure 2.1: CICIDS2018 Network Topology (Sharafaldin. et al., 2018)

Leevy and Khoshgoftaar (2020) surveyed papers on classification models based on CICIDS2018 and found a gap on how the class balance is dealt, this impact on the researches reproducibility. The same author also mention works that used accuracy as a metric for a classification task on CICIDS2018, the accuracy metric can be misleading as it doesn't take in consideration what the model have failed to identify and could mask the problem of imbalanced learning thus an inefficient classifier for the given domain.

[1] https://www.unb.ca/cic/datasets/index.html

[2] https://github.com/ahlashkari/CICFlowMeter

2.2 Imbalanced Learning

The imbalanced learning problem occurs when one of the classes of a dataset is less frequent in proportion with the other(s), as shown in figure 2.2. For (Weiss et al., 2007) classifiers trained on imbalanced datasets can have their predictions biased towards the most common class, for example in credit card fraud detection, a minority of the samples will be fraud and a classifier may not be able to predict when it is a fraud or a just a normal transaction, to a level that matches a business risk appetite.

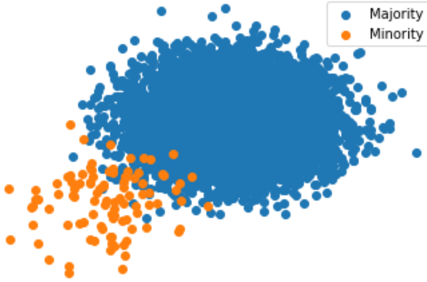


Figure 2.2: An imbalanced dataset with two classes

Prati et al. (2009) synthesizes the concepts, metrics and also methods to counteract underrepresentation of classes in datasets for inferring classification models, they explain that the problem of imbalanced data can be tackled at either data-level by modifying the train set distribution or at algorithm-level by providing misclassification cost information prior to training or posterior by defining decision thresholds. The same authors also mention a hybrid approach which is a mix of data and algorithm level methods.

Japkowicz and Stephen (2002) researched on how different types of imbalanced data in terms of training set size and imbalance proportion do affect classification algorithms such as C5.0, neural networks and support vector machines (SVM), and found that data and algorithm level methods to deal with class imbalance will have a different effect based on the dataset size and complexity. Buda et al. (2018) explored imbalanced data effects and possible counteracts for the problem at both data and algorithm levels on a convolutional neural network (CNN) classifier trained on three different image datasets, they found that ROS outperformed RUS and decision thresholding and then recommended a hybrid approach of both data and algorithm level methods to deal with class imbalance.

Seiffert et al. (2008) evaluated the effects of using four data-level manipulations: ROS, RUS, SMOTE and borderline-SMOTE, and also algorithm-level approaches: Cost-Sensitive Classifier and modifying the decision threshold of a classification algorithm. The tests were executed on 15 datasets, where RUS performed significantly better at higher imbalance ratios, and the algorithm-level approaches outperformed ROS, SMOTE and borderline-SMOTE. The two algorithm-level Cost-Sensitive Classifier and modifying the decision threshold had similar performance.

Pawlicki et al. (2020) tested a RF to classify CICIDS2017 traffic, the authors tested ROS, RUS, borderline-SMOTE and other undersampling techniques such as Tomek-Links, and also at algorithm-level a weighted classification in the RF. Surprisingly RUS outperformed the other data-level balancing methods. But in overall, the algorithm-level approach had the best recall values.

Although literature suggest an algorithm-level (Pawlicki et al., 2020; Seiffert et al., 2008) or hybrid (Prati et al., 2009) approach to the imbalanced learning problem, this work is focusing only on the data-level scope, although the classifier proposed here could be tested at different decision threshold levels, a data-level approach is agnostic to one objectives for using the data. A data-level method to deal with imbalance allows to train other classification models that are sensitive to class imbalance, such as neural networks (Buda et al., 2018).

2.3 Random Forest IDS

Detecting anomalies in a computer network can be a challenge, because of the high volume and dimensionality of data, false positive alerts caused by network traffic normal profile change and intruders adapting their attack techniques to evade detection, making past detection methods obsolete (Chandola et al., 2009).

Random Forest (Breiman, 2001) is a type of ensemble model that is computationally efficient and is less prone to imbalanced learning problems, RF can deal with high dimensional data (Chen et al., 2004), such as network intrusion. A RF is composed of many decision trees, as seen in figure 2.3, each tree will vote towards a target class, these votes are weighted by the probability estimative for the analysed sample be a malicious or benign network traffic, for example.

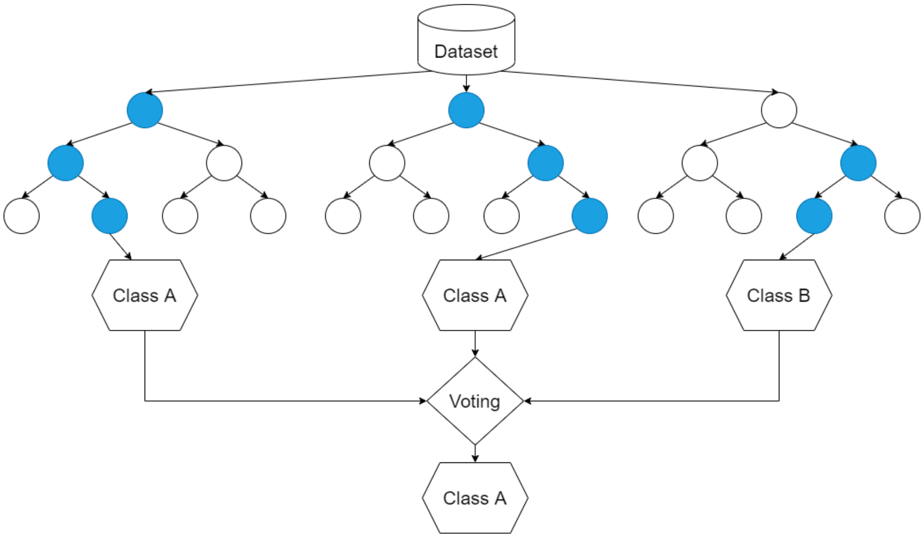


Figure 2.3: Random forest prediction.

RF have been used to classify IDS data on other similar datasets such as the CICIDS2017 (Lee & Park, 2019b, 2019a), and other datasets such Kyoto 2006+ in (Park et al., 2018) and NSL-KDD in (Tesfahun & Bhaskari, 2013), and outside of cybersecurity by (Khalilia et al., 2011) to predict disease risks from highly imbalanced data. In this work a RF will be implemented to act as the network intrusion detection system.

2.4 Naive Strategies

Random Oversampling (ROS) is one of the simplest approaches to deal with imbalanced learn, it consists in copying random samples from the minority class until the imbalance is solved, it is also known as sampling with replacement. The shortcoming of ROS is that it increases the chance of overfitting models. Random Undersampling

(RUS), is another naive approach that is based on randomly removing samples from the majority classes to achieve balance, with the shortcoming that valuable information can be lost (Mishra, 2017). Both ROV and RUS are very easy to implement because they don't perform any heuristical decision to deal with data and can serve as a baseline to compare more advanced methods.

2.5 Machine Learning

2.5.1 SMOTE

Proposed by (Chawla et al., 2002) SMOTE is one method to augment data to overcome the imbalanced learning problem, by creating synthetic samples from an interpolation between nearest neighbours.

A synthetic sample S is generated from the difference between a randomly chosen sample Si and one other sample in its nearest neighbour Sn multiplied by a random number between 0 and 1 and then added to the chosen sample, described in equation below.

$$S = Si + (|Sn - Si|) * rand[0, 1]$$

SMOTE consists in selecting samples from a minority class clustering them into k-nearest neighbours and creating new data points between the neighbours as shown in figure 2.4 where the minority class is in red and the synthetic samples sit in between the original samples.

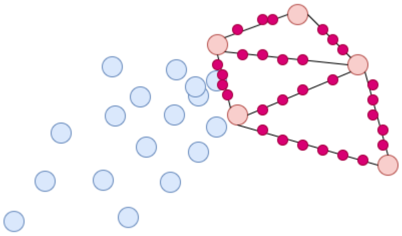


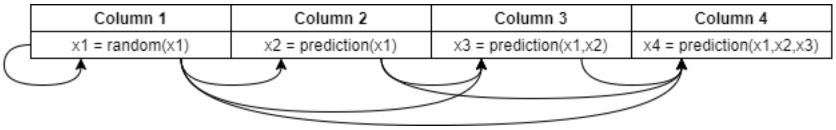
Figure 2.4: SMOTE interpolation between samples.

SMOTE is simple to implement and the only heuristic required is clustering the samples in k nearest neighbours. The downside of SMOTE is that it can create an overlap with other similar classes leading to a fuzzy decision boundary which can cause noise in the synthetic data. Such issue is mentioned in a research to improve detection of Android malwares in an imbalanced dataset and was reworked by Y. Xu et al. (2017) by increasing the number of minority samples near the decision boundaries.

2.5.2 CART

Classification and Regression Trees (CART) were proposed by (Breiman et al., 1984), and used by (Reiter, 2003) to generate synthetic data to address privacy issues on sensitive values of microdata, to allow its publication. Sabay et al. (2018) have used CART to create a surrogate dataset to deal with class imbalance and privacy issues in Breast Cancer and Nursery datasets and successfully trained a classifier model.

Synthetic data generation using CART consists in predicting the next feature based on the previous features, so the last feature will be conditional on all the previous features, thus preserving correlations, the first feature is a special case and is randomly sampled with replacement from the original dataset (Nowok et al., 2016), as shown in figure 2.5.



Column 1	Column 2	Column 3	Column 4
x1 = random(x1)	x2 = prediction(x1)	x3 = prediction(x1,x2)	x4 = prediction(x1,x2,x3)

Figure 2.5: Synthetic data generation based on CART.

CART models are capable of working with non-gaussian distributions and capturing non-linear relationships like network intrusion data, but can be hard to be interpret the generated model (Caiola & Reiter, 2010), which is not a problem on this research

2.6 Deep Learning

2.6.1 GAN

A Generative Adversarial Network (GAN) is a machine learning architecture consisting of two neural networks, the generator and the discriminator who compete against each other in a min-max game. The generator wants to maximize and the discriminator to minimize the loss of the discriminator. These two neural networks are trained simultaneously, from random noise the generator will learn how to produce fake samples that look like real data. These fake samples from the generator will feed to the discriminator along with samples from a dataset of real data, the discriminator job then is to distinguish/predict if a sample is real data or is a fake sample produced from the generator. Again, the idea is that the generator should be able to produce fake data that is very similar to the real data distribution so that the discriminator is not able to distinguish between genuine and fake data, as shown in 2.7.

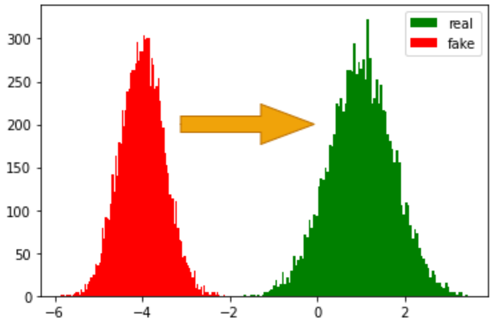


Figure 2.6: GAN training objective.

The training process for the generator consists in feeding the generator input layer with random data of fixed size, for example an array 128 of random numbers between -1 and 1, this values will then feed forward to the neural network layers until it reaches the output layer, the output layer size should be of the same size of the original data, for example the number of columns in a dataset, by this time the fake sample is ready to be evaluated by the discriminator.

The discriminator will be trained by analysing samples from the generator and also from real data, these samples will feed to the discriminator input layer that should have the same size as the original data, the samples are then feed forward until it reaches the discriminator output layer of size one, which will return a value of 1 for real and 0 for fake, for example.

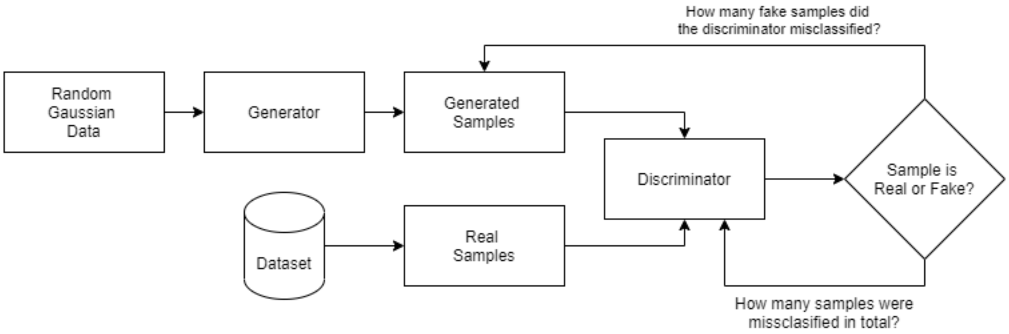


Figure 2.7: GAN training flow

The discriminator loss value will be calculated based on the misses and hits from his predictions of real and fake samples, while the generator loss is based only on the discriminator predictions of the fake samples, the generator doesn't care about real samples.

Each model having its separate loss value allows to calculate the derivative of the error gradient and backpropagate adjustments to the layers of each model. GAN training will occur until the generator is able to produce synthetic data that assembles real data, for example in figure 2.6 one wants to the red portion (synthetic samples) being the most similar as the green mass (real samples).

Lee and Park (2019b, 2019a) have implemented a GAN to generate data to solve imbalanced data issues on CICIDS2017, the predecessor of CICIDS2018, with positive results for some classes of attacks. To simulate a malware that leverages from artificial intelligence Shu et al. (2020) used a GAN with a variational autoencoder (VA) as the generator, the VA is type of model that maps raw data to a hyperspace, this GAN produced simulated malicious samples capable of bypass a neural network classifier model acting as a NIDS.

GAN Issues

The original GAN architecture is known be difficult to train and to be applied to generate tabular data. GAN is a trending topic and researchers are working around the issues and exploring GAN capabilities, researches mention some common problems such as mode collapse, vanishing gradients and non-convergence and propose workarounds for such issues (Walia et al., 2020; Arjovsky et al., 2017; Karlsson & Sjöberg, 2020; L. Xu et al., 2019).

Mode collapse is when the generator has learned to produce only a single type or very similar outputs, and the discriminator is always fooled by this fake sample causing the learning process to get stuck, the discriminator is not able to minimize its loss function. For example in a GAN to generate tabular data with categorical columns, that was the detect in the experiments conducted for this research, the GAN learned the imbalance present in the original dataset and was producing only a single traffic type.

Vanishing gradients happen when the discriminator is not able to provide useful feedback for the generator, the generator loss value which is produced by the discriminator is so small that the generator won't improve. GANs work well with images because of their range between 0 and 255. Applying min-max transformation to a continuous value present in a tabular dataset will be likely to cause the gradients vanishing (L. Xu et al., 2019).

As the generator objective is to produce fake samples similar to real, at certain point the discriminator feedback will be of less importance, it is also hard to tell when a GAN model is producing fake samples of good quality, there's no definitive evaluation metric (Borji, 2019). If the GAN is trained for too long the generator may start to produce useless samples of poor quality, that's called non-convergence.

2.6.2 CTGAN

Conditional Tabular Generative Adversarial Network (CTGAN) was proposed by (L. Xu et al., 2019) to address issues arising when using a GAN to generate tabular data, such as coexistence of continuous and categorical data in the same dataset; Non-Gaussian distributions from continuous data that can't go through min-max transformations; Multi-modal distributions, vanilla GAN is not able to capture continuous values multi-modes.

CTGAN estimate the continuous columns based on the number of modes present on the distribution, which is given by fitting a Gaussian mixture model into the columns, a value present in the third mode would be represented in a one-hot encoding vector α = [0,0,1] vector, for example. This vector will be concatenated with a scalar β representing the value inside that mode, that's how a continuous column is presented to the CTGAN generator.

The CTGAN takes in consideration that values from a categorical column may be imbalance, and the generator may never learn the minority classes. Each categorical column value will be presented mutually excluded from any other categorical value. Consider one row with columns D1 and D2 as "marital status" (single, married, **divorced**) and "is employed" (yes, **no**) to one hot encoded vector D1=[0,0,1] and D2=[0,1]. Between the two columns "is employed" is randomly picked, the two vectors D1 and D2 are concatenated into a mask M = [0,0,0,0,1] this mask will be mapped from an original sample and presented to the discriminator. The generator is free to populate the mask with any values it wants, but will get penalized thus forcing it to eventually learn to pick only one categorical value of only a single column value, that also forces the generator to learn the continuous values associated with the respective categorical value.

The continuous and categorical column representations are then concatenated and feed to the discriminator as shown in figure 2.8

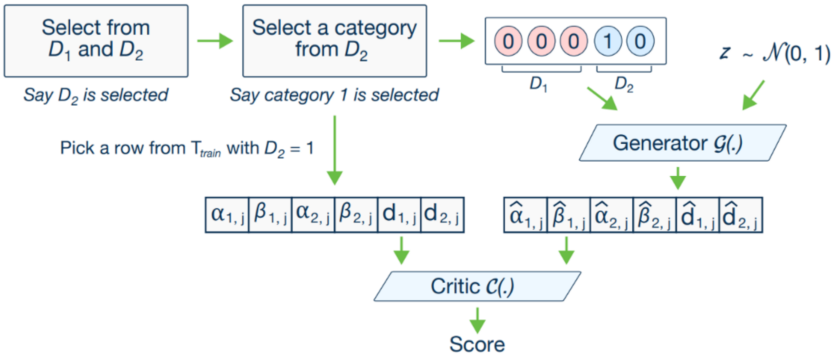

Select from D_1 and D_2
Say D_2 is selected
Select a category from D_2
Say category 1 is selected
0 0 0 1 0
D_1
D_2
$z \sim \mathcal{N}(0, 1)$
Pick a row from T_{train} with $D_2 = 1$
Generator $\mathcal{G}(.)$
$\alpha_{1,j}$ $\beta_{1,j}$ $\alpha_{2,j}$ $\beta_{2,j}$ $d_{1,j}$ $d_{2,j}$
$\hat{\alpha}_{1,j}$ $\hat{\beta}_{1,j}$ $\hat{\alpha}_{2,j}$ $\hat{\beta}_{2,j}$ $\hat{d}_{1,j}$ $\hat{d}_{2,j}$
Critic $\mathcal{C}(.)$
Score

Figure 2.8: CTGAN Conditional Generator. (L. Xu et al., 2019)

CTGAN also leverages from other advancements in GAN research, it uses the Wasserstein distance as loss on the discriminator, or critic as it's known on the Wasserstein GAN (WGAN) (Arjovsky et al., 2017), this new loss function is the distance between real data and generated data distributions, similar to figure 2.6, in contrast to the cross entropy used by the vanilla GAN, the Wasserstein distance is a meaningful metric to tell the distance. The generator objective is to minimize distance while the discriminator looks to maximize the distance, once the loss function gets to a value of zero the discriminator is no longer able to distinguish between real and fake. In WGAN the critic output is linear and ranges between -1 and 1 respectively for fake and real, in contrast with vanilla GAN that uses sigmoid in the discriminator output layer thus limited between 0 and 1.

To avoid vanishing gradients, and also the inverse, exploding gradients, the authors clipped the discriminator weights in the range between -0.01 and 0.01, which the authors in (Gulrajani et al., 2017) found to be problematic and proposed to enforce the same constraint in form of penalty on the gradient of the discriminator. This constraint is known as Lipschitz constrain, explanation of such mathematical property is outside of the proposed scope for this research and can be referred in the original paper.

The CTGAN also incorporates the packing element (Lin et al., 2017), which consist in presenting N more samples from the same class during the discriminator training, for example it will present 10 real samples or 10 fake samples to be analysed by the discriminator at once, which is an efficient method to avoid mode collapse thus forcing the GAN to diversify on the generate samples.

Tang et al. (2020) have compared classifiers trained with CTGAN augmented data to predict oil reservoirs quality which didn't improved the classification task. GAN based models have been used in the financial domain to generate synthetic data. Karlsson and Sjöberg (2020) experimented with WGAN-GP as a baseline against CTGAN, that is designed to deal with tabular data, WGAN-GP (Gulrajani et al., 2017) had better results on continuous simulated data and CTGAN on adult census, a more categorical dataset. (Walia et al., 2020) have compared SMOTE as a baseline against WCGAN-GP in the efficiency on machine learning tasks and on privacy metrics with favourable results to WCGAN-GP on cardiovascular and credit card fraud datasets, but not on adult census which have more categorical data than the previous datasets.

At the time of writing no other research on network intrusion detection made use of CTGAN before.

2.7 Evaluation Methods

An evaluation framework is necessary to ensure the generated data is a reliable representation of its origin. At a first stage the synthetic data is evaluated in how similar it is to the real data, that's done using histograms to visually judge the distribution shapes while pairwise correlation allows both visual and numerical inspection on the columns dependencies. The most important metric in this experiment is the machine learning efficiency which tells the synthetic data suitability for a classification model inference.

2.7.1 Synthetic data quality

Synthetic data evaluation can be done in a visual manner by plotting both real and synthetic data to a histogram. However, the histograms, do not allow to check if the synthetic data is following the same correlations as the real data does, that motivated to use the pairwise correlation evaluation.

Histogram

Histograms will provide visual aid to check if the synthetic data is able to follow the same ranges of original data distributions, a histogram will be generated for each real and synthetic column pair. The synthetic data is represented in red, real data in green, the grey parts is when both overlaps, meaning that the synthetic data generator is able to capture that set. It is expected that the generated data would capture the mean and modes and that can be evaluated using a histogram.

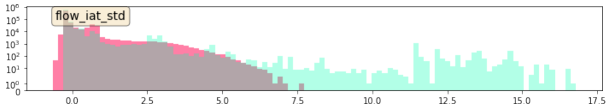


Figure 2.9: Example of histogram comparison between real and fake data.

Figure 2.9 is a small example of a histogram for one column, a full picture is available in appendix A.1, the figure shows that the synthetic data has captured the mode and some part of the real data distribution, although with the synthetic a bit outside of the original data range.

Pairwise correlation

To tackle the histogram limitations, similarly to (Karlsson & Sjöberg, 2020) who used a statistical measure to test synthetic data fidelity by using a correlation matrix that provides both visual and numerical ways to evaluate. This evaluation is mainly necessary to check if the deep learning methods are generating good quality data, since GAN and derivatives may struggle to capture feature inter-dependencies.

The Pearson ”r” correlation coefficient tells how much two columns are correlated, its value ranges between -1 and 1. A positive correlations value means that while one variable X increases other variable Y will increase at similar proportion, for example the number of times a clothes dryer machine appliance is used within a month and the electric bill at the end of the moth. A negative value means that while a variable X increases, other variable Y will decrease, for example the number of times one did physical exercise will decrease chance of heart disease. A value of zero means there’s no correlation. A matrix of correlations shows the correlation value for each pair of variables, as shown in figure 2.10 a small example, the ones used later in this work contain 68x68 squares and are available in the appendix A.3.

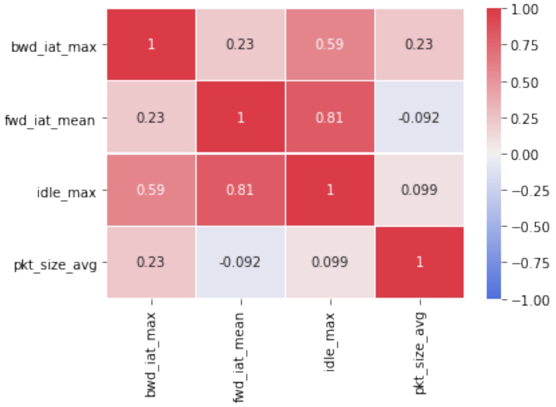


Figure 2.10: Example of correlation matrix

Each square of synthetic data is subtracted by its real data equivalent square, which is then transformed to an absolute value to generate a new matrix, the mean of this new matrix will return the correlation value for each feature, by calculating the mean again, a single value is obtained and is used to facilitate the comparison between different synthetic datasets. A value of zero for this coefficient means the synthetic data does follow exactly the same correlation degrees as the real data does. Good synthetic data should have the pairwise correlation value as close to zero as possible, and avoid going any higher.

2.7.2 Machine learning efficiency

The objective of this research is to evaluate different synthetic data generation methods to solve the imbalanced learn problem on a classification model, machine learning efficiency is the key measure to assess the different methods experimented here, it tell if the data augmentation methods are suitable for machine learning tasks.

There's no definitive evaluation metric for a machine learning classifier, the metric must be chosen taking in consideration the problem it is trying to solve, the accuracy metric for example can be misleading when dealing with an imbalanced dataset (Saito & Rehmsmeier, 2015; He & Garcia, 2009; Parkinson de Castro, 2020), while precision metric can be used when false positives are a bigger concern than false negatives, and recall metric should be used when false negatives are more prejudicial than false positives. For example, it is preferable to say someone has cancer and forwarding this person to do more tests which then can confirm s/he's healthy, than saying that there's no disease and the person wouldn't seek any treatment.

Precision and Recall

In the context of network intrusion detection a model precision relates to total number of records that are correctly detected as attacks among all the traffic, while recall tells if the samples classified as attacks were in reality just normal traffic, to assure maximum detection and low noise the models should respectively have high precision and high recall.

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

Considering attack the positive class and benign traffic the negative class, a true positive (TP) and a true negative (TN) respectively mean that an attack and a benign traffic flow were correctly classified, a false positive (FP) would mean that a benign traffic was misclassified as attack and a false negative (FN) that an attack was misclassified as benign. The impact in a real-world scenario is that a false positive could break a legitimate business application and the false negative that an attacker successfully by-passed detection mechanisms. Is up to each business to decide on the trade-off between FP (Precision) and FN (Recall) to match with their network security requirements.

F1-Score

Introduced by (Van Rijsbergen, 1979) F1-Score is a more robust metric for evaluating an imbalanced classification task, it consists in the harmonic mean of precision and recall, the F1-Score allows quicker decision process by looking at a single number per class.

$$\text{F1} = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN}$$

Misclassification per class

There are 12 different types of attack in the CICIDS2018, some are harder to detect than others, by analysing the misclassification per class is possible to analyse how the different data augmentation methods perform on different attack types.

Chapter 3

Experiment design and methodology

The purpose of this chapter is to describe the experiment framework developed towards the research objective, question, sub-questions. Here's is discussed the steps taken to build the test framework, issues found and workarounds, to assure reproducibility of the study.

3.1 Dataset

The CICIDS2018 a dataset of network flows, which is a summarized sequence of packages from one host to one or more hosts in a computer network, the CICIDS2018 was generated in the network topology shown in Figure 2.1. The dataset contains 82 columns, fully listed in appendix A.1, and 16,232,943 samples representing network flows of 14 types of attacks plus benign traffic, composing approximately 83% of benign and 17% of malicious traffic flows. The CICIDS2018 was designed to be used on network anomaly detection studies, like a classification model built to distinguish between benign and malicious traffic to generate security alerts to contain attacks in a hypothetical scenario. Most of the CICIDS2018 data is benign, reflecting the fact that in a corporate network environment the vast majority of traffic is benign. As shown in table 3.1 the target classes — attacks — are unbalanced in favour of benign samples, this can negatively impact classification models.

Traffic Type	Total	Percentage
Benign	12,817,082	83.07%
DDOS attack-HOIC	686,012	4.23%
DDoS attacks-LOIC-HTTP	576,191	3.55%
DoS attacks-Hulk	461,912	2.85%
Bot	286,191	1.76%
FTP-BruteForce	193,360	1.19%
SSH-Bruteforce	187,589	1.16%
Infilteration	161,934	1.00%
DoS attacks-SlowHTTPTest	139,890	0.86%
DoS attacks-GoldenEye	41,508	0.26%
DoS attacks-Slowloris	10,990	0.07%
DDOS attack-LOIC-UDP	1,730	0.01%
Brute Force -Web	611	0.00%
Brute Force -XSS	230	0.00%
SQL Injection	87	0.00%

Table 3.1: CICIDS2018 Class Labels

3.1.1 Data Preprocessing

According to Leevy and Khoshgoftaar (2020) there's no detailed literature on how CICIDS2018 can be cleaned and prepared for machine learning and the class imbalance solved. It was noticed that some files had repeated header information on the body, which may indicate that the parser to generate such datasets was executed more than once or multiple files were appended to a single one. Another finding is that one of the files "Thuesday-20-02-2018" (sic) has more columns than the others, such as Network Flow ID, Source IP, Source Port, Destination IP. Such columns could have been used for designing other experiments that take into consideration where traffic is coming from and going to, in a real-world application these columns can be used to fine tune a NIDS and reduce number of false positives.

The column timestamp is ignored since the problem is not analysed from a time-series perspective, along with dst_port since its pair src_port is present only to a single file on the dataset, other columns with no variance are also removed, resulting in a 68 column train and test sets. The dataset literally presented ”Infinite” values which were are replaced by “NA”. The target variable indicates if traffic is benign or which attack type it represents, the target variable is then expanded to two columns, the first a boolean flag indicating if it’s an attack or benign traffic and the second, a numeric code for the traffic type.

Data is then split into 80% for training and 20% for test using stratified sampling, NA values are then replaced by the respective column median. Stratified sampling allows each split to have a proportional percentage of target classes.

It is important to scale and normalize the data since some machine learning algorithms can be sensitive working with data that is not normal, such as neural networks, which implies in the GAN and CTGAN. Standard scaler is chosen since L. Xu et al. (2019) used a MinMax scaler to train a GAN but found that the test models didn’t capture the distribution modes. Standard scaler normalization is fitted on the training set and transformed both training and test sets.

Although most of the CICIDS2018 columns are numeric/continuous, it was noticed on some columns that their top 5 most common values comprised a threshold of 99% of the dataset, and that’s how categorical columns are defined for CART in this research, namely columns: ack flag cnt, cwe flag count, down up ratio, ece flag cnt, fin flag cnt, fwd psh flags, fwd urg flags, psh flag cnt, rst flag cnt, syn flag cnt, urg flag cnt. A similar method will be applied to define categorical columns to the CTGAN, which is explained later.

3.2 Data Augmentation

3.2.1 ROS

Random oversampling augmented attacks to 100,000 samples for each attack type by copying the samples with replacement until the desired number. ROS is very simple to implement and cheap in computational requirements since no complex calculations are required, which scales well to big datasets, in contrast to other robust machine learning approaches. Although it may cause the classification model to overfit thus leading to bias towards few samples from the training data, which may prejudice the model generalization to unseen data.

3.2.2 RUS

Random undersampling limited each attack type and benign traffic to a maximum of 50,000 samples, which is half of what was asked for ROS, SMOTE, CART, GAN and CTGAN to generate. Like ROS, it is also very simple to implement with the same benefits for both methods. The downside is that valuable samples may be randomly removed, which may negatively impact on the model predictions, RUS per se doesn't allow to delete only samples that won't prejudice a machine learning model, for example.

3.2.3 SMOTE

This method one could call a smart approach to oversampling when compared to ROS and RUS. SMOTE requires a simple training of k-nearest neighbours algorithm, after that it is possible to specify the desired number of samples to generate for each class, which was set to 100,000. The downside of SMOTE is that it can create samples that overlap with other classes and that may prejudice the decision boundary of the classifier.

3.2.4 CART

With CART it was possible to generate 100,000 samples for each attack type, but not for SQL Injection and DDoS attacks-LOIC-HTTP attacks due to highly correlation in columns, causing the algorithm to halt and sometimes crashing the application. Fortunately, that didn't interfere with final experimentation results as seen later in table 4.4

Benign samples were excluded from CART training due to performance issues. CART allows to define which columns to treat as categorical, these were selected based on if their five most common values do comprise 99% of the distribution as explained in 3.1.1.

3.2.5 GAN

A GAN was trained for each type of attack, batch size was dependant on the number of training samples. The batch size was set to 32, 64, 128 for classes with less than 500, 2,000, 10,000 samples respectively, and batch size of 512 for when higher than 10,000. All the GANs were trained for 10,000 epochs, which took a total of three hours.

The GAN architecture and parameters are detailed in table 3.2, the architecture used on the experiments was defined based on tests using 4 to 6 columns randomly selected from the dataset and also from a random data distribution of 4 columns. The same tests showed that a generator input layer of size 128 produced better samples and showed more stability than 32 and 256 neurons.

Parameter	**Generator**	**Discriminator**
Input Layer -Number of neurons	128	68 (Number of columns in training dataset)
Hidden Layer 1	128 Leaky ReLu (Dropout, alpha: 0.2)	512 Leaky ReLu (Dropout, alpha: 0.2)
Hidden Layer 2	256 Leaky ReLu (Dropout, alpha: 0.2)	256 Leaky ReLu (Dropout, alpha: 0.2)
Hidden Layer 3	512 Leaky ReLu (Dropout, alpha: 0.2)	128 Leaky ReLu (Dropout, alpha: 0.2)
Output Layer - Number of neurons	68 (Number of columns in training dataset) - Linear	1 - Linear
Loss function: Binary Cross Entropy. Optimizer: Adam rate at 0.0002, betas b1 = 0.9, b2 = 0.999		

Table 3.2: GAN Architecture

It is worth mentioning the number of neurons on the output layer of the generator and the input layer of the discriminator are the same, as per table 3.2, these must be the same number of the columns from the data one is looking to generate.

3.2.6 CTGAN

Like the GAN, individual CTGAN models were trained per attack category, the number of epochs and batch size was dependant on the number of samples available in the training dataset.

Similar to CART, CTGAN also allows to specify the categorical columns, these were defined after a number of tests using different thresholds for considering a column categorical or continuous. If the top 5 most common values did comprise — a threshold of — 80% of the column, then consider the column as categorical, for example. The issue is that depending on the threshold set, if the number of samples for a class was too high the application would run out of memory and crash.

The default CTGAN architecture is two layers of 256 neurons for the generator and discriminator, but after tests it was observed that the respective crescent and decrescent pattern for generator and discriminator did improved the results, as shown in table 3.3, on the flip side this larger architecture had increased the training time to 72 hours.

	Generator	Discriminator
Input Layer -Number of neurons	128	68 * 10 (Packs)
Hidden Layer 1	128 ReLu (BatchNorm 1D)	128 Leaky ReLu (Dropout, alpha: 0.5)
Hidden Layer 2	256 ReLu (BatchNorm 1D)	256 Leaky ReLu (Dropout, alpha: 0.5)
Hidden Layer 3	512 ReLu (BatchNorm 1D)	512 Leaky ReLu (Dropout, alpha: 0.5)
Output Layer - Number of neurons	68 - Linear	1 - Linear
Gradient Penalty	Adam - Decay (L2) 0.000006	Lipschitz Constraint
Loss function: Wasserstein Distance. Optimizer: Adam rate at 0.0002, betas b1 = 0.9, b2 = 0.999		

Table 3.3: CTGAN Architecture

Similar to the GAN, the same principle of the number of neurons in the output and input layers of the generator and discriminator applies to the CTGAN. However, as explained in section 2.6.2 the CTGAN uses the idea of packs, which means that in this case is that 10 samples will be presented to the discriminator in each epoch, which is one workaround against the known issue of mode collapse present in the vanilla GAN.

3.3 Experimental Design

Figure 3.1 shows the experiment flow, the train data will feed the generator methods, the generators will be trained where applicable, once trained the generators will produce synthetic data to be evaluated regarding the generation method ability to capture the original data distribution and correlations. To then augment the original dataset to solve class imbalance and train a random forest, leading to a more robust test which is the machine learning efficiency, this last test will tell which data-augmentation method is doing a better job in solving the class imbalance problem.

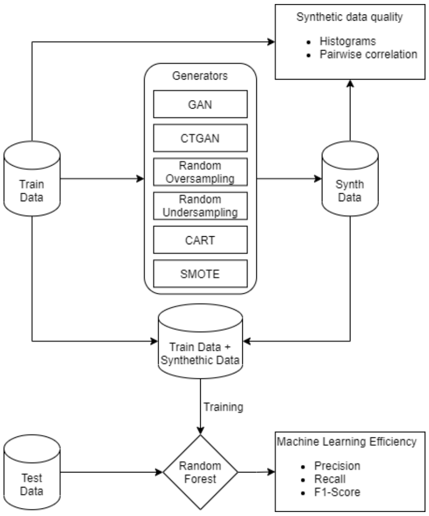


Figure 3.1: Experiment Flow

3.3.1 Synthetic data quality

By analysing a histogram that plots the generated data against its real counterpart, it is possible tell if the generator method was able to capture the data ranges and different modes from the original data it was trained on. In the case the GAN and

CTGAN one should look if the synthetic data is stuck to a single mode, which may indicate mode collapse, that in practice means the generator only learned to output a single or a limited set of samples.

The pairwise correlation can be checked either in the plots or in coefficient values, it tells how good the generation method has captured the data correlations. The goal is to get the synthetic data correlation plot as similar as possible to the real data plot, while the coefficient value should be as close to zero as possible.

3.3.2 Machine learning efficiency

Machine learning efficiency metric is obtained after training a random forest to classify the network flows as benign or malicious regardless of which attack type the sample represents, assuming that the classifier would be able to capture new attacks that doesn't exist today regardless of the attack type. The data generator output is summed with its own training data, this bigger and balanced dataset will serve as training data for the RF, as pictured in Figure 3.1

Misclassification per class

There's only few samples for some attack types, table 3.1 shows only 87 samples for SQL Injection, for example. To get a better understanding and a fair comparison of which attack categories the RF is able to detect a detailed misclassification report is also provided.

Research Question

Machine learning efficiency is the key metric to answer the research question, as classification algorithms are negatively affected by the imbalanced learning problem. This metric allows to evaluate *To what extent adding synthetic generated samples to augment the minority classes of CICIDS2018 dataset to train a random forest classifier model is capable of improving the model precision, recall and F1-Score*

3.4 Computational Environment

CART training and data generation was performed in R Studio with an Intel i7 processor and 16gb of RAM. All the other experiments were computed using Python 3.6 in a Google Colab notebook hosted with 25gb of Ram, 4x Intel Xeon CPU at 2.30GHz and randomly allocated GPUs between Nvidia K80s, T4s, P4s and P100s [1]. Where applicable the random seed was set to the number 42. Table 3.4 lists the external software used on the experiments.

Software/Package	Version	Purpose
Tensorflow	2.4.1	Deep learning
Pytorch	1.4	Deep learning
CTGAN	0.3.1	Deep learning
Sklearn	0.23	Machine Learning
Imblearn	0.4.3	Machine Learning
Numpy	1.19.5	Vector and matrix operations
Pandas	1.1.4	Dataset operations
Matplotlib	3.2.2	Visual Graphics
Seaborn	0.11.1	Visual Graphics
ml-ids	NA	CSV Processing, Visual Graphics
Synthpop	1.6.0	Machine Learning

Table 3.4: List of external libraries used

Due to computational lab environment memory limitations only the first 100,000 lines on nine out of the ten CICIDS2018 files will be read to allow CTGAN to be trained and fairly compared to other methods. This caused to leave Infilteration (sic), SSH and FTP Bruteforce attacks out of the experiment, still 900,000 samples to be used 80% for training and 20% for test.

[1] https://research.google.com/colaboratory/faq.html

Chapter 4

Results, evaluation and discussion

A classifier trained on a dataset with imbalanced data can lead to biased predictions towards the majority class, data-augmentation techniques are one approach to deal with such problem. The experiment evaluates the use of naive, machine learning and deep learning approaches to deal with class imbalance when training a random forest. This section presents results and comparisons of the implementation of such data-augmentation techniques.

4.1 Synthetic data generation

In an early experiment it was noticed that the data generations methods CART, GAN and CTGAN learned the class imbalance when trained with all the data, which may indicate that these methods are sensible to class imbalance. To solve that problem, each method trained one model per attack class, resulting in 11 different model for GAN and CTGAN methods. For CART it was 9 models as it couldn't learn SQL Injection and DDoS attacks-LOIC-HTTP as explained in section3.2.4.

The two most basic techniques were ROS with RUS, the first consists in duplicating existing samples until the class balance is achieved, the second reduces the number of the majority class samples, to 50,000 in the case of this experiment. SMOTE was other method applied, by generating data samples in between the lines of 5 k-nearest neighbour clusters of a minority class to upsample the minority samples to 100,000 instances each attack class.

It took 15 minutes to train nine CART models individually per attack class except SQL Injection and DDoS attacks-LOIC-HTTP, these two classes have columns with high correlation and caused the algorithm to halt, leaving these classes out of the experiment didn't interfere, as showed later in this section.

Eleven GAN models were trained for each attack class for 10,000 epochs which took around three hours in total, the GAN was called to generate 100,000 samples per class. Another eleven models were produced using a CTGAN, which took a total of 72 hours of training, in a bespoke approach per attack class, defining batch size, epochs and which columns to treat as categorical. The CTGAN was called to generate 100,000 samples per class.

By this time seven different datasets: Original, ROS, RUS, SMOTE, CART, GAN and CTGAN were ready to be evaluated in terms of how well it did capture the original data correlation and probability distributions. Table 4.1 shows how many samples were available for the machine learning efficiency test.

	Data Augmentation Method						
Traffic Type	Original	CART	ROS	RUS	SMOTE	GAN	CTGAN
Benign	375157	375157	375157	50000	375157	375157	375157
Bot	64146	100000	60000	50000	100000	100000	100000
Brute Force -Web	489	100000	60000	489	100000	100000	100000
Brute Force -XSS	184	100000	60000	184	100000	100000	100000
DDOS attack-HOIC	76851	100000	60000	50000	100000	100000	100000
DDOS attack-LOIC-UDP	1384	100000	60000	1384	100000	100000	100000
DDoS attacks-LOIC-HTTP	79934	79934	60000	50000	100000	100000	100000
DoS attacks-GoldenEye	33206	100000	60000	33206	100000	100000	100000
DoS attacks-Hulk	6640	100000	60000	6640	100000	100000	100000
DoS attacks-SlowHTTPTest	73147	100000	60000	50000	100000	100000	100000
DoS attacks-Slowloris	8792	100000	60000	8792	100000	100000	100000
SQL Injection	70	70	60000	70	100000	100000	100000
Total	720,000	1,355,161	1,035,157	300,765	1,475,157	1,475,157	1,475,157

Table 4.1: Number of traffic sample types per augmentation method

4.2 Synthetic data quality

Evaluation of the synthetic is one of the steps towards the research question, these evaluations were designed with the deep learning methods in mind, the GAN is known to be difficult to train and to be applied to tabular data. CTGAN is a GAN variant that is designed to address known issues from the GAN, the synthetic data quality evaluation will provide support to detect and workaround such issues.

The histograms illustrated if the generated data managed to capture real data distribution column by column. The original data which was used to train the generator model is shown in green, the generator output in red. Where the generator managed to capture the distribution is shown in grey, which is an overlap between real and synthetic data distributions. Figure 4.1 shows only five columns, while a full comparison of the 68 features for each method is available in the appendix A.

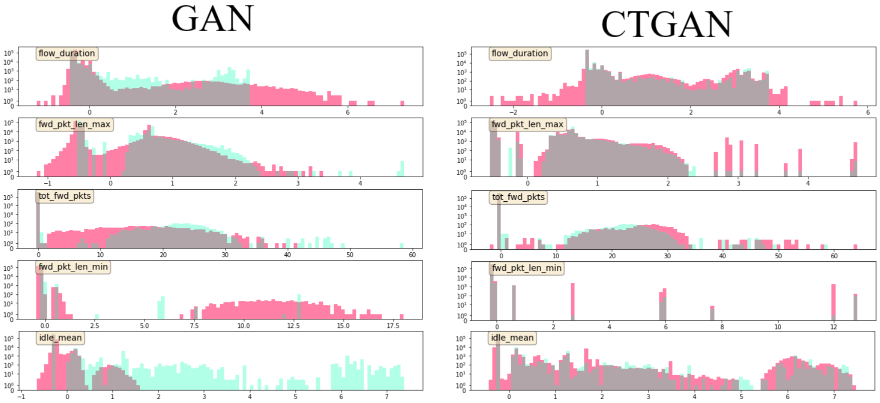


Figure 4.1: CTGAN comparison histogram

By analysing the histograms in figure 4.1, it is observed in flow_duration column that both GAN and CTGAN went beyond the ranges of the original data distribution. In column fwd_pkt_len_max the GAN also went beyond the original data range. While CTGAN was able to capture very specific data points between 2.7 and 5 in X axis. In column tot_fwd_pkts the GAN created data points were there's no real data, between 3 and 6. In column fwd_pkt_len_min is the GAN struggled on that distribution that looks to be categorical data, while the CTGAN did a good job covering the real data points. Mode collapse, when the generator learns to produce a limited set of outputs, is visible in idle_mean and occurred for GAN, meaning the generator models is able to produce limited values for this column. While CTGAN managed to capture the different modes of idle_mean.

As described in section 2.7.1 the pairwise correlation has the objective to assess if the data generation method captured the relationships between variables from which it was trained on. A value of zero means that the dataset is exactly the same, so the low the number the better. As shown in table 4.2 and figure 4.2 the CTGAN captured the correlations better than GAN. The goal in figure 4.2 is to have the generator method correlation matrix as close as possible to the original data, full size images are available in appendix 2.6.

ROS	RUS	SMOTE	CART	GAN	CTGAN
0.09540	0.0462	0.0958	0.1051	0.2283	0.0593

Table 4.2: Pairwise Correlation

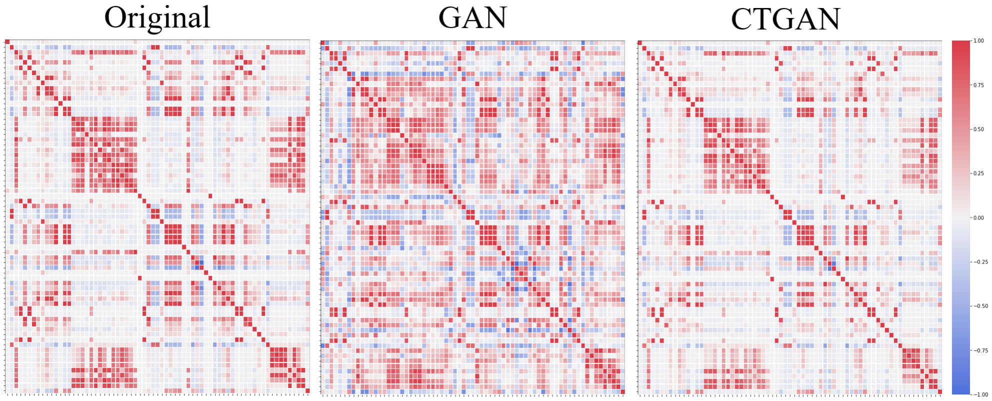


Figure 4.2: Pairwise correlation figures the original data and synthetic data from GAN, CTGAN.

4.3 Machine learning efficiency

With the synthetic data quality inspected, the experiment continued to the most important tests. Seven IDS models were trained on different datasets, of which six used data augmentation techniques as described in section 3.2, while the remaining model used the original dataset. The models were evaluated on the same test set in terms of precision, recall and F1-Score. Three possible scenarios are presented as interpretations to the results, to fulfil one objectives towards the security of a hypothetical network environment, where the objective can be one of the following:

a) have the safest environment

b) the most functional environment

c) a balance between security and functionality

	Precision	Recall	F1
CART	99.3964%	99.9049%	99.6500%
SMOTE	99.0686%	99.9362%	99.5005%
ROS	99.0914%	**99.9385%**	99.5132%
RUS	99.6585%	99.8620%	99.7601%
GAN	99.7403%	99.8063%	99.7733%
CTGAN	**99.9014%**	99.9084%	**99.9049%**
Original	99.7231%	99.8353%	99.7792%

Table 4.3: Machine learning efficiency

Security is priority

If one is focusing to have the most secure environment, then the recall metric should be maximized. Low recall would imply in attacks being classified as benign traffic. The results available in table 4.3, show that ROS and SMOTE respectively have the highest recall among all the methods, thus letting a minimal number of attacks passing through the network. In general, every method had good performance on recall. ROS and SMOTE recall was followed by CTGAN, CART, RUS, Original and GAN methods.

Functionality is priority

Although not the most performant for recall, CTGAN and GAN scored respectively the best precision. In a real-world scenario, a high precision means they are the less obstructive methods, with the lowest false positives. CTGAN and GAN precision were followed by Original, RUS, CART, ROS and SMOTE. If aiming for a less obstructive environment, the precision metric should be maximized, but with the possible downside of being less secure. Low precision implies in legitimate traffic being classified as attack while a low recall will not detect attacks correctly, leading to higher number of false negatives.

Balance between security and functionality

The F1-Score is the harmonic mean between precision and recall. If one is looking for balance between security and functionality of the network, one should maximize the F1-Score. It is evident that CTGAN showed the best trade-off between precision and recall, followed by Original, GAN, RUS, CART, ROS and SMOTE.

Misclassifications

Some traffic types could be more easily detected than others as shown in table 4.4. From the bottom of the table, the traffic types DDOS attack-LOIC-UDP, DoS attacks-SlowHTTPTest, DDoS attacks-LOIC-HTTP, DoS attacks-GoldenEye were 100% detected by all the different NIDS models. The GAN was the only one that didn't improve detection of DoS attacks-Hulk and increased Brute Force-XSS and SQL Injection misclassification.

Interestingly CTGAN was the only data-augmentation technique that improved DDOS attack-HOIC detection, while other methods lead to worse — higher misclassifications — on DDOS attack-HOIC. Similar to DoS attacks-Slowloris CTGAN was the only one that showed improvement.

The only method capable of improving bot detection was ROS, although further investigation is required to check if ROS didn't cause the model to overfit. The other

Number of Samples Misclassified

	Original	ROS	RUS	SMOTE	CART	GAN	CTGAN
Benign	295	790	500	810	523	224	90
Brute Force -Web	69	10	27	8	28	72	35
DoS attacks-Slowloris	24	24	24	25	24	25	18
Bot	9	4	11	9	11	30	13
DDOS attack-HOIC	2	13	14	12	14	15	2
SQL Injection	8	2	4	1	3	10	2
Brute Force -XSS	5	0	0	0	2	13	3
DoS attacks-Hulk	2	0	0	0	0	2	0
DoS attacks-GoldenEye	0	0	0	0	0	0	0
DDoS attacks-LOIC-HTTP	0	0	0	0	0	0	0
DoS attacks-SlowHTTPTest	0	0	0	0	0	0	0
DDOS attack-LOIC-UDP	0	0	0	0	0	0	0
Total Attacks	119	**53**	80	55	82	*167*	73
Total Benign	295	790	500	*810*	523	224	**90**
Total	414	843	580	*865*	605	391	**163**

Table 4.4: Number of Misclassified Samples

methods lead to worse or no difference in results for bot. SMOTE and ROS showed competitive results for Brute Force-Web.

The deep learning methods were the only ones that reduced misclassification on benign samples, all other methods lead to increased benign traffic misclassification, which relates to findings discussed in previous subsection.

Chapter 5

Conclusion

5.1 Research Overview

The objective of this research was to compare the effect of using data augmentation techniques to generate samples to augment the minority classes of the CICIDS2018. The impact was measured by using the augmented dataset to train a random forest classifier model, which was then in terms of precision, recall and F1-Score.

The experiments were performed on CICIDS2018, a big dataset and one of the newest network intrusion detection benchmarking datasets available for public use. The experiment was composed of three main parts: augmenting an imbalanced dataset using different methods; evaluating the quality of the synthetic data in terms of similarity to the real data counterpart; and finally comparing how much influence each data balancing method had on a binary classification task executed by a random forest, measured in terms of precision, recall and F1-Score.

5.2 Problem Definition

The internet increases opportunities for businesses. On the other hand, being connected to the internet also increases the risks related to information security, which may be associated to a regulatory requirement or protection of trade secrets, for example. A network intrusion detection system is capable of mitigating the risks coming from being connected to the internet by detecting or even better, containing intruders. A NIDS can be implemented as a classification model to detect for anomalous patterns by listening to the traffic flow of an enterprise computer network. In such network many business applications would be running and exchanging information with nodes inside and outside the company network boundaries, like a messaging application or even an employee navigating the internet. It is expected that minority of the network traffic would be malicious, being very small in proportion to the legitimate traffic.

Many classification problems have to deal with class imbalance which can have a negative impact on a classifier performance, one example is the data of a network intrusion detection system. The CICIDS2018 is a NIDS benchmark dataset that comprises of attacks and a network infrastructure that reflect the actual cyberspace, in counterpart of datasets from twenty years ago that are still used by researchers.

As expected, major part of CICIDS2018 is comprised of benign traffic leading to imbalanced learn issues that can be solved by augmenting the data using classic approaches like ROS, RUS, SMOTE, CART and also deep learning approaches such as GAN and its derivative, CTGAN. GANs are a sophisticated neural network architecture capable of generating synthetic/fake data that can't be distinguished between real and fake, usually applied to images GANs are known to be difficult to train and to be applied to tabular data. CTGAN is designed to improve the vanilla GAN to generate synthetic tabular data, and can be used to overcome the imbalanced learn problem in CICIDS2018.

5.3 Design/Experimentation, Evaluation & Results

The evaluation of synthetic data quality in section 4.2 allowed to answer the **sub-question 1:** To what extent are the synthetic generated samples are capable of capturing the original data distribution and correlations? The histograms allowed to understand the deep learning method's behaviour and to workaround with experiments towards answering the research main question. It was possible to check that the vanilla GAN did suffer from mode collapse, since some columns got stuck to a single mode. Thus, limiting the quality of the samples generated by the GAN and consequently impacting the classifier efficiency.

Pairwise correlation measures the capability of the synthetic generation method to capture the real data distribution it was trained on. RUS had the best pairwise correlation which is expected since it is a subset of the original data. RUS pairwise correlation was followed by CTGAN, ROS, SMOTE being similar to CART, and GAN with the lowest results. Showing that the capability of capturing feature interdependencies were important to some extend but it's not necessarily enough to assure machine learning efficiency, given the good pairwise correlation results for RUS and CART.

From the results in table 4.3, the machine learning efficiency metric allowed to answer **sub-question 2:** Will the data augmentation methods influence on the misclassification of each attack type? CTGAN was capable of improving the detection of one attack type, DoS attacks-Slowloris that no other methods were capable of doing so.

Machine learning efficiency metric also answers the **Research question:** To what extent adding synthetic generated samples to augment the minority classes of CICIDS2018 dataset to train a random forest classifier model is capable of improving the model precision, recall and F1-Score?

It was provided three interpretations to the results, depending on one strategy and risk appetite to cyber security, ultimately is up to a business to decide what risks can be accept and match with their requirements.

A more restrictive scenario could leverage from the methods that scored higher on recall, such as ROS and SMOTE, the top 2 that scored similarly (ROS: 99.9385%, SMOTE: 99.3262%). While a high score in precision, achieved by the CTGAN (99.9014%), indicate that the method is the best at keeping a business running smoothly, since it trains a less intrusive NIDS (RF) that is better at detecting benign traffic, the downside may be less security. The balance between security and functionality can be achieved by leveraging the methods that scored the best F1-Score, which was the CTGAN (99.9049%).

Hence there's information to contribute towards answering the research question, it is possible to use data-augmentation methods to improve a random forest classification task on the CICIDS2018 dataset.

The research question remains partially answered since this experiment didn't provide statistical tests for the machine learning efficiency metric. Other limitation is that due to computational resources restrictions only 5% of the dataset was used. Considerable efforts was needed to use CTGAN to achieve such results, a bespoke CTGAN model was trained for each attack category, demanding experimentation and consumption of expensive computational resources that are required for such method.

Also using the CICIDS2018 Pawlicki et al. (2020) had better results with an algorithm-level approach to deal with class imbalance. The results here are similar to (Lee & Park, 2019a, 2019b) who successfully improved a random forest by augmented data using deep learning methods on the CICIDS2017 (Sharafaldin et al., 2019), the previous version of the dataset presented in this work.

5.4 Contributions and impact

This research tested different data augmentation methods to deal with imbalanced learn in an up-to-date network intrusion detection benchmark dataset. The augmentation methods include CTGAN (L. Xu et al., 2019) a state-of-the-art improvement of GAN to generate synthetic tabular data. The experiments demonstrated the capability of the CTGAN model to generate synthetic data for the CICIDS2018.

Through evaluation, it was found that synthetic data generated from a CTGAN model, to deal with class imbalance did improve the classification task performance in terms of precision, F1-Score, and still keep a high recall score. While much simpler methods such as ROS and SMOTE performed better in recall than state-of-the-art approaches.

This work has contributed towards dealing with the class imbalance problem present on CICIDS2018, which was highlighted by (Leevy & Khoshgoftaar, 2020). Also extending the research on deep learning algorithms for synthetic data generation for the specific dataset.

5.5 Future Work & recommendations

CTGAN took a long time to train, with sometime spent on fitting Gaussian mixture models to capture the modes of continuous columns, other methods to capture distribution modes could be explored. Hyperparameter tuning for the GAN and CTGAN models was not fully explored and could produce substantial research. Both GAN and CTGAN take a long time to train and also require more computational resources than the other methods, while SMOTE still an interesting method to augment data due to its simplicity to implement and fast training and sample generation. GAN and variants may be an interesting approach if one is looking to address privacy issues within the data, which SMOTE doesn't so well (Walia et al., 2020). GAN and derivatives work well with images (Karras et al., 2019), which have a limited data range (0-255) and may be difficult to generalise with broader data ranges.

Due to constraints of time and resources it was not possible to use enough data and computational power to carry out the repeated measures necessary for tests of statistical significance, future work should look to work on results through statistical tests thus providing a more robust case. Other classification models could be implemented as a NIDS and class weights could be specified as an algorithm-level approach to deal with the imbalanced learning problem, as mentioned in the literature review. Due to the CICIDS2018 dataset size on around 10gb, a big data approach to the experiments is an interesting path. CTGAN looks promising in generating synthetic tabular data and in addressing the mode collapse issue.

References

Arjovsky, M., Chintala, S., & Bottou, L. (2017, 06–11 Aug). Wasserstein generative adversarial networks. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th international conference on machine learning* (Vol. 70, pp. 214–223). International Convention Centre, Sydney, Australia: PMLR. Retrieved from `http://proceedings.mlr.press/v70/arjovsky17a.html`

Biswas, S. (2018). Intrusion detection using machine learning: A comparison study. *International Journal of Pure and Applied Mathematics (IJPAM)*, *118*(19), 101–114.

Borji, A. (2019). Pros and cons of gan evaluation measures. *Computer Vision and Image Understanding*, *179*, 41-65. Retrieved from `https://www.sciencedirect.com/science/article/pii/S1077314218304272` doi: https://doi.org/10.1016/j.cviu.2018.10.009

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32. Retrieved from `https://doi.org/10.1023/A:1010933404324` doi: 10.1023/A:1010933404324

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and regression trees.* doi: 10.1201/9781315139470

Buczak, A. L., & Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys Tutorials*, *18*(2), 1153-1176. doi: 10.1109/COMST.2015.2494502

Buda, M., Maki, A., & Mazurowski, M. A. (2018). A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, *106*, 249–259.

Caiola, G., & Reiter, J. P. (2010, April). Random forests for generating partially synthetic, categorical data. *Transactions on Data Privacy*, *3*(1), 27–42.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR)*, *41*(3), 15. doi: 10.1145/1541880.1541882

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, *16*, 321–357.

Chen, C., Liaw, A., & Breiman, L. (2004). *Using random forest to learn imbalanced data* (Technical Report No. 666). Department of Statistics: University of California, Berkeley. Retrieved from `https://statistics.berkeley.edu/tech-reports/666`

Draper-Gil., G., Lashkari., A. H., Mamun., M. S. I., & Ghorbani., A. A. (2016). Characterization of encrypted and vpn traffic using time-related features. In *Proceedings of the 2nd international conference on information systems security and privacy - volume 1: Icissp,* (p. 407-414). SciTePress. doi: 10.5220/0005740704070414

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., … Bengio, Y. (2014). Generative adversarial nets. In *Proceedings of the 27th international conference on neural information processing systems - volume 2* (pp. 2672–2680). Cambridge, MA, USA: MIT Press. Retrieved from `http://dl.acm.org/citation.cfm?id=2969033.2969125`

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. C. (2017). Improved training of wasserstein gans. In I. Guyon et al. (Eds.), *Advances in neural information processing systems* (Vol. 30). Curran Associates, Inc. Retrieved from `https://proceedings.neurips.cc/paper/2017/file/892c3b1c6dccd52936e27cbd0ff683d6-Paper.pdf`

He, H., & Garcia, E. A. (2009, Sep.). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, *21*(9), 1263-1284. doi: 10.1109/TKDE.2008.239

Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: A systematic study. *Intell. Data Analytics.*, *6*, 429-449.

Karlsson, A., & Sjöberg, T. (2020). *Synthesis of tabular financial data using generative adversarial networks* (Unpublished master's thesis). KTH, Mathematical Statistics.

Karras, T., Laine, S., & Aila, T. (2019, June). A style-based generator architecture for generative adversarial networks. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition (cvpr).*

Khalilia, M., Chakraborty, S., & Popescu, M. (2011). Predicting disease risks from highly imbalanced data using random forest. *BMC medical informatics and decision making*, *11*(1), 1–13.

Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, *2*(1), 1–22.

Lashkari., A. H., Gil., G. D., Mamun., M. S. I., & Ghorbani., A. A. (2017). Characterization of tor traffic using time based features. In *Proceedings of the 3rd international conference on information systems security and privacy - volume 1: Icissp,* (p. 253-262). SciTePress. doi: 10.5220/0006105602530262

Lee, J., & Park, K. (2019a). Ae-cgan model based high performance network intrusion detection system. *Applied Sciences*, *9*(20). Retrieved from `https://www.mdpi.com/2076-3417/9/20/4221` doi: 10.3390/app9204221

Lee, J., & Park, K. (2019b). Gan-based imbalanced data intrusion detection system. *Personal and Ubiquitous Computing*, 1–8. doi: 10.1007/s00779-019-01332-y

Leevy, J. L., & Khoshgoftaar, T. M. (2020). A survey and analysis of intrusion detection models based on cse-cic-ids2018 big data. *Journal of Big Data*, *7*(1), 1–19.

Lin, Z., Khetan, A., Fanti, G., & Oh, S. (2017, 12). Pacgan: The power of two samples in generative adversarial networks. *IEEE Journal on Selected Areas in Information Theory*, *PP*. doi: 10.1109/JSAIT.2020.2983071

Mishra, S. (2017). Handling imbalanced data: Smote vs. random undersampling. *International Research Journal of Engineering and Technology (IRJET)*, *4*(8).

Nowok, B., Raab, G. M., & Dibben, C. (2016). synthpop: Bespoke creation of synthetic data in r. *Journal of Statistical Software, Articles*, *74*(11), 1–26. Retrieved from `https://www.jstatsoft.org/v074/i11` doi: 10.18637/jss.v074.i11

Park, K., Song, Y., & Cheong, Y. (2018, March). Classification of attack types for intrusion detection systems using a machine learning algorithm. In *2018 ieee fourth international conference on big data computing service and applications (bigdataservice)* (p. 282-286). doi: 10.1109/BigDataService.2018.00050

Parkinson de Castro, E. (2020). *An examination of the smote and other smote-based techniques that use synthetic data to oversample the minority class in the context of credit-card fraud classification* (Master's thesis, Technological University Dublin). doi: https://doi.org/10.21427/wj33-n221

Pawlicki, M., Choraś, M., Kozik, R., & Hołubowicz, W. (2020, 06). On the impact of network data balancing in cybersecurity applications. In (p. 196-210). doi: 10.1007/978-3-030-50423-6_15

Prati, R. C., Batista, G. E., & Monard, M. C. (2009). Data mining with imbalanced class distributions: concepts and methods. In *Iicai* (pp. 359–376).

Reiter, J. P. (2003). Using cart to generate partially synthetic, public use microdata. *Journal of Official Statistics*, 441–462.

Rigaki, M., & Garcia, S. (2018). Bringing a gan to a knife-fight: Adapting malware communication to avoid detection. In *2018 ieee security and privacy workshops (spw)* (p. 70-75). doi: 10.1109/SPW.2018.00019

Sabay, A., Harris, L., Bejugama, V., & Jaceldo-Siegl, K. (2018). Overcoming small data limitations in heart disease prediction by using surrogate data. *SMU Data Science Review*, *1*(3), 12.

Saito, T., & Rehmsmeier, M. (2015, 03). The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PLOS ONE*, *10*(3), 1-21. Retrieved from `https://doi.org/10.1371/journal.pone.0118432` doi: 10.1371/journal.pone.0118432

Seiffert, C., Khoshgoftaar, T. M., Hulse, J. V., & Napolitano, A. (2008, Dec). A comparative study of data sampling and cost sensitive learning. In *2008 ieee international conference on data mining workshops* (p. 46-52). doi: 10.1109/ICDMW.2008.119

Sharafaldin, I., Gharib, A., Habibi Lashkari, A., & Ghorbani, A. (2017, 01). Towards a reliable intrusion detection benchmark dataset. *Software Networking*, *2017*, 177-200. doi: 10.13052/jsn2445-9739.2017.009

Sharafaldin., I., Lashkari., A. H., & Ghorbani., A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the 4th international conference on information systems security and privacy - volume 1: Icissp,* (p. 108-116). SciTePress. doi: 10.5220/0006639801080116

Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2019). A detailed analysis of the cicids2017 data set. In P. Mori, S. Furnell, & O. Camp (Eds.), *Information systems security and privacy* (pp. 172–188). Cham: Springer International Publishing.

Shiravi, A., Shiravi, H., Tavallaee, M., & Ghorbani, A. A. (2012). Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*, *31*(3), 357-374. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0167404811001672` doi: https://doi.org/10.1016/j.cose.2011.12.012

Shu, D., Leslie, N. O., Kamhoua, C. A., & Tucker, C. S. (2020). Generative adversarial attacks against intrusion detection systems using active learning. In

Proceedings of the 2nd acm workshop on wireless security and machine learning (p. 1–6). New York, NY, USA: Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/3395352.3402618` doi: 10.1145/3395352.3402618

Tang, J., Fan, B., Xu, G., Xiao, L., Tian, S., Luo, S., ... others (2020). A new tool for searching sweet spots by using gradient boosting decision trees and generative adversarial networks. In *International petroleum technology conference.*

Tesfahun, A., & Bhaskari, D. L. (2013, Nov). Intrusion detection using random forests classifier with smote and feature reduction. In *2013 international conference on cloud ubiquitous computing emerging technologies* (p. 127-132). doi: 10.1109/CUBE.2013.31

Van Rijsbergen, C. (1979). Information retrieval: theory and practice. In *Proceedings of the joint ibm/university of newcastle upon tyne seminar on data base systems* (pp. 1–14). doi: https://doi.org/10.1002/asi.4630300621

Walia, M. S., Bredan, T., & Susan, M. (2020). Synthesising tabular datasets using wasserstein conditional gans with gradient penalty. Technological University Dublin. doi: 10.21427/e6wa-sz92

Weiss, G. M., McCarthy, K., & Zabar, B. (2007). Cost-sensitive learning vs. sampling: Which is best for handling unbalanced classes with unequal error costs? In *Proceedings of the 2007 international conference on data mining, DMIN 2007, june 25-28, 2007, las vegas, nevada, USA* (pp. 35–41). CSREA Press.

Xu, L., Skoularidou, M., Cuesta-Infante, A., & Veeramachaneni, K. (2019). Modeling tabular data using conditional gan. In *Advances in neural information processing systems* (Vol. 32). Curran Associates, Inc. Retrieved from `https://proceedings.neurips.cc/paper/2019/file/254ed7d2de3b23ab10936522dd547b78-Paper.pdf`

Xu, Y., Wu, C., Zheng, K., Niu, X., & Yang, Y. (2017). Fuzzy–synthetic minority oversampling technique: Oversampling based on fuzzy set theory for android malware detection in imbalanced datasets. *International Journal of Distributed Sensor Networks*, *13*(4), 1550147717703116. Retrieved from https://doi.org/10.1177/1550147717703116 doi: 10.1177/1550147717703116

Appendix A

Additional Content

Feature Name	Description
ack_cnt	Number of packets with ACK
atv_avg	Mean time a flow was active before becoming idle
atv_max	Maximum time a flow was active before becoming idle
atv_min	Minimum time a flow was active before becoming idle
atv_std	Standard deviation time a flow was active before becoming idle
bw_blk_rate_avg	Average number of bulk rate in the backward direction
bw_byt_blk_avg	Average number of bytes bulk rate in the backward direction
bw_hdr_len	Total bytes used for headers in the forward direction
bw_iat_avg	Mean time between two packets sent in the backward direction
bw_iat_max	Maximum time between two packets sent in the backward direction
bw_iat_min	Minimum time between two packets sent in the backward direction
bw_iat_std	Standard deviation time between two packets sent in the backward direction
bw_iat_tot	Total time between two packets sent in the backward direction
bw_pkt_blk_avg	Average number of packets bulk rate in the backward direction
Bw_pkt_l_avg	Mean size of packet in backward direction
Bw_pkt_l_max	Maximum size of packet in backward direction
Bw_pkt_l_min	Minimum size of packet in backward direction
Bw_pkt_l_std	Standard deviation size of packet in backward direction
bw_pkt_s	Number of backward packets per second
bw_psh_flag	Number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP)
bw_seg_avg	Average size observed in the backward direction
bw_urg_flag	Number of times the URG flag was set in packets travelling in the backward direction (0 for UDP)
bw_win_byt	# of bytes sent in initial window in the backward direction
cwe_cnt	Number of packets with CWE
down_up_ratio	Download and upload ratio
ece_cnt	Number of packets with ECE
fin_cnt	Number of packets with FIN
fl_byt_s	flow byte rate that is number of packets transferred per second
fl_dur	Flow duration
fl_iat_avg	Average time between two flows
fl_iat_max	Maximum time between two flows
fl_iat_min	Minimum time between two flows
fl_iat_std	Standard deviation time two flows
fl_pkt_s	flow packets rate that is number of packets transferred per second
Fw_act_pkt	# of packets with at least 1 byte of TCP data payload in the forward direction
fw_blk_rate_avg	Average number of bulk rate in the forward direction
fw_byt_blk_avg	Average number of bytes bulk rate in the forward direction
fw_hdr_len	Total bytes used for headers in the forward direction
fw_iat_avg	Mean time between two packets sent in the forward direction
fw_iat_max	Maximum time between two packets sent in the forward direction

Feature Name	Description
fw_iat_min	Minimum time between two packets sent in the forward direction
fw_iat_std	Standard deviation time between two packets sent in the forward direction
fw_iat_tot	Total time between two packets sent in the forward direction
fw_pkt_blk_avg	Average number of packets bulk rate in the forward direction
fw_pkt_l_avg	Average size of packet in forward direction
fw_pkt_l_max	Maximum size of packet in forward direction
fw_pkt_l_min	Minimum size of packet in forward direction
fw_pkt_l_std	Standard deviation size of packet in forward direction
fw_pkt_s	Number of forward packets per second
fw_psh_flag	Number of times the PSH flag was set in packets travelling in the forward direction (0 for UDP)
fw_seg_avg	Average size observed in the forward direction
fw_seg_min	Minimum segment size observed in the forward direction
fw_urg_flag	Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)
fw_win_byt	Number of bytes sent in initial window in the forward direction
idl_avg	Mean time a flow was idle before becoming active
idl_max	Maximum time a flow was idle before becoming active
idl_min	Minimum time a flow was idle before becoming active
idl_std	Standard deviation time a flow was idle before becoming active
pkt_len_avg	Mean length of a flow
pkt_len_max	Maximum length of a flow
pkt_len_min	Minimum length of a flow
pkt_len_std	Standard deviation length of a flow
pkt_len_va	Minimum inter-arrival time of packet
pkt_size_avg	Average size of packet
pst_cnt	Number of packets with PUSH
rst_cnt	Number of packets with RST
subfl_bw_byt	The average number of bytes in a sub flow in the backward direction
subfl_bw_pkt	The average number of packets in a sub flow in the backward direction
subfl_fw_byt	The average number of bytes in a sub flow in the forward direction
subfl_fw_pk	The average number of packets in a sub flow in the forward direction
syn_cnt	Number of packets with SYN
tot_bw_pk	Total packets in the backward direction
tot_fw_pk	Total packets in the forward direction
tot_l_fw_pkt	Total size of packet in forward direction
urg_cnt	Number of packets with URG

Table A.1: CICIDS2018 Columns (Sharafaldin. et al., 2018)

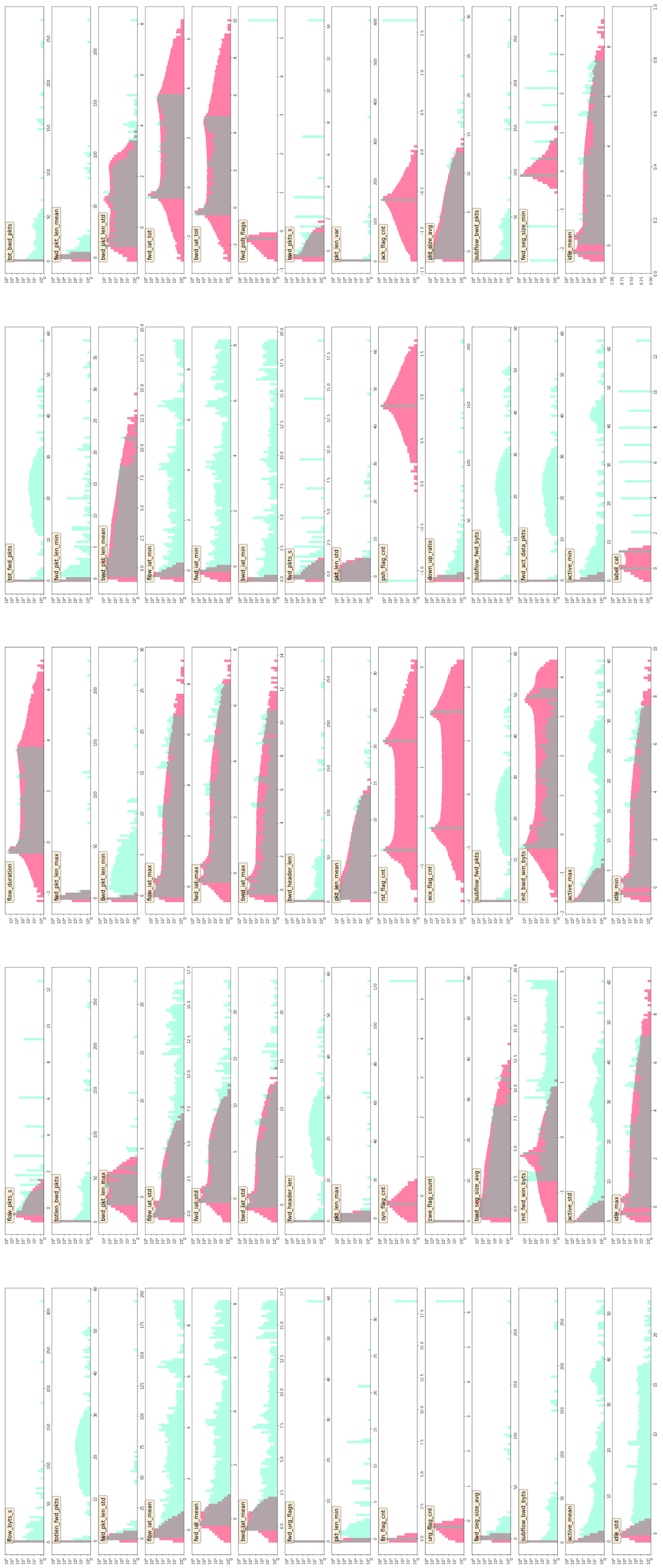


Figure A.1: GAN Histograms

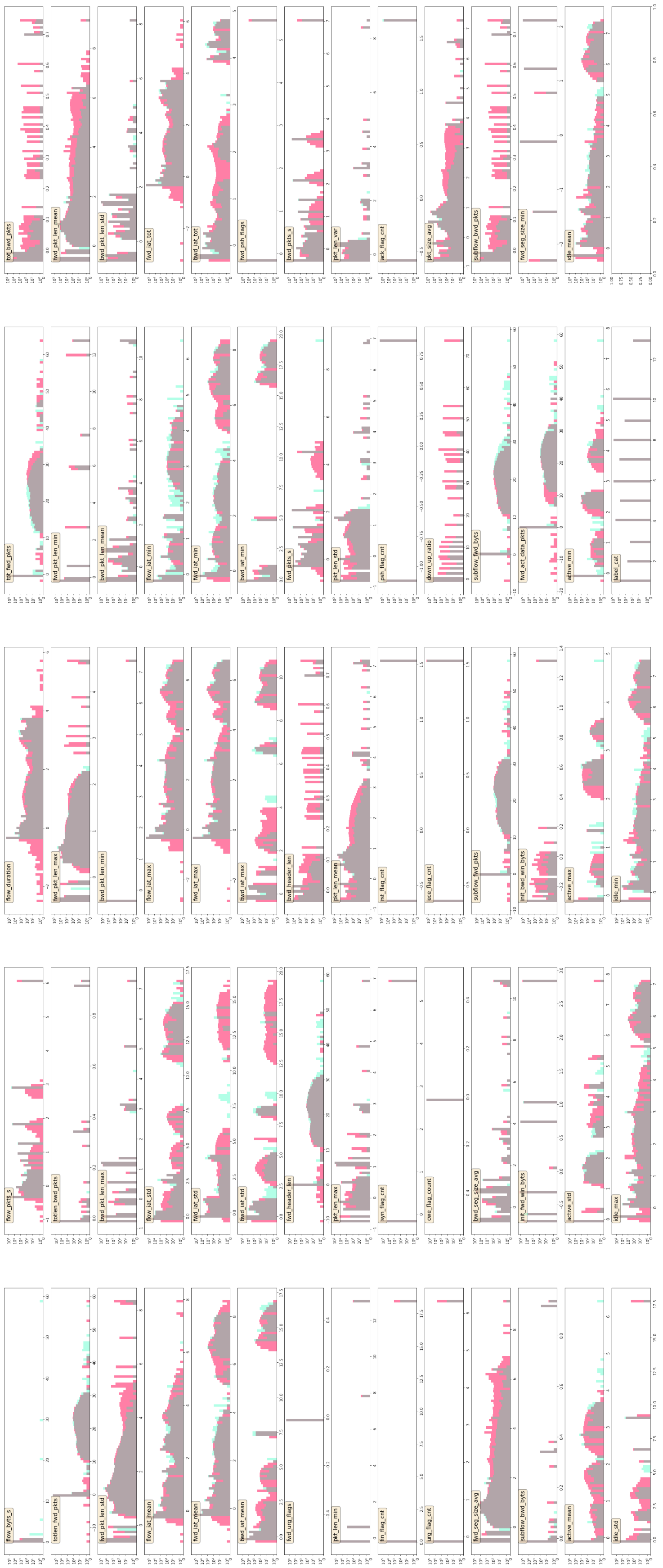


Figure A.2: CTGAN Histograms

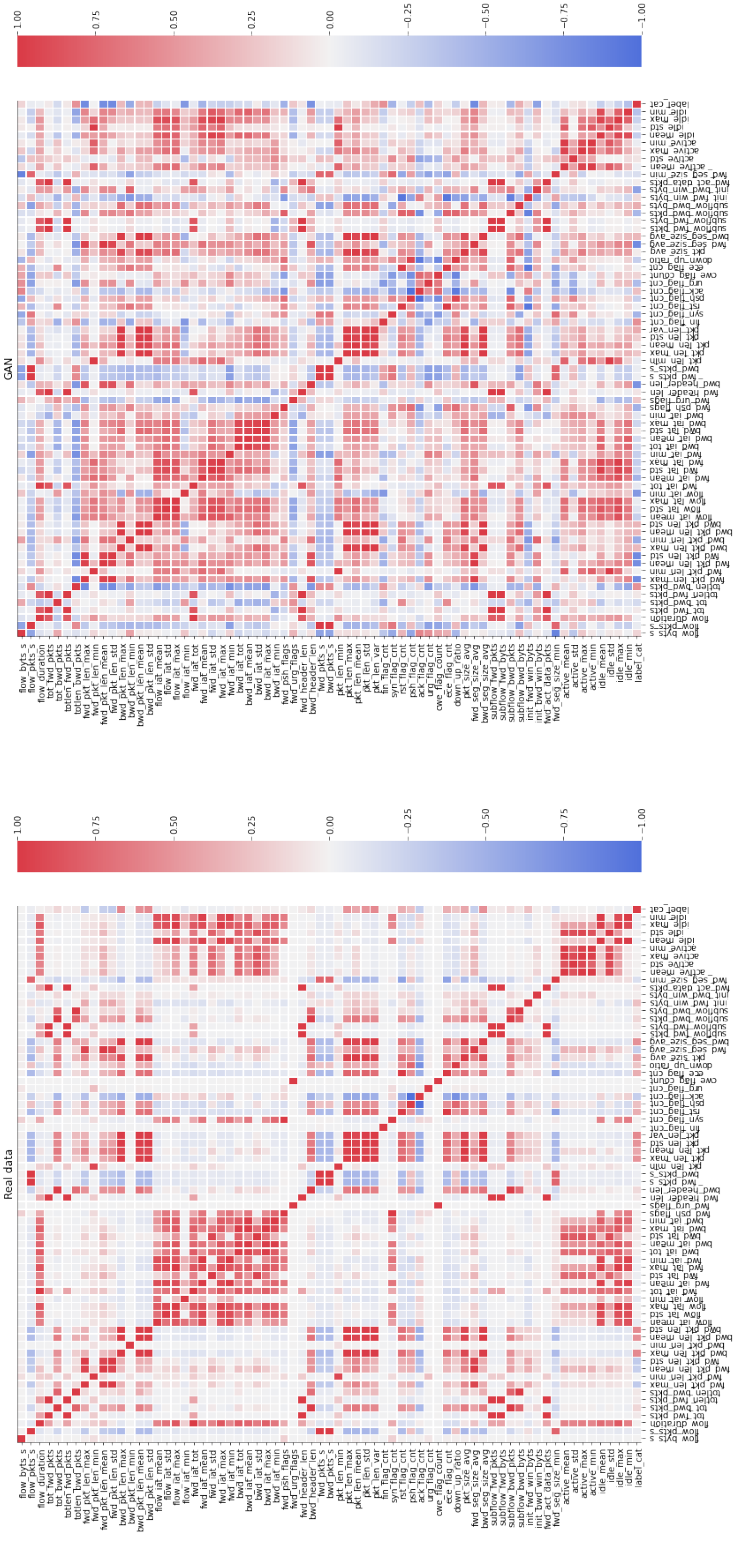


Figure A.3: GAN pairwise correlation matrix comparison with real data

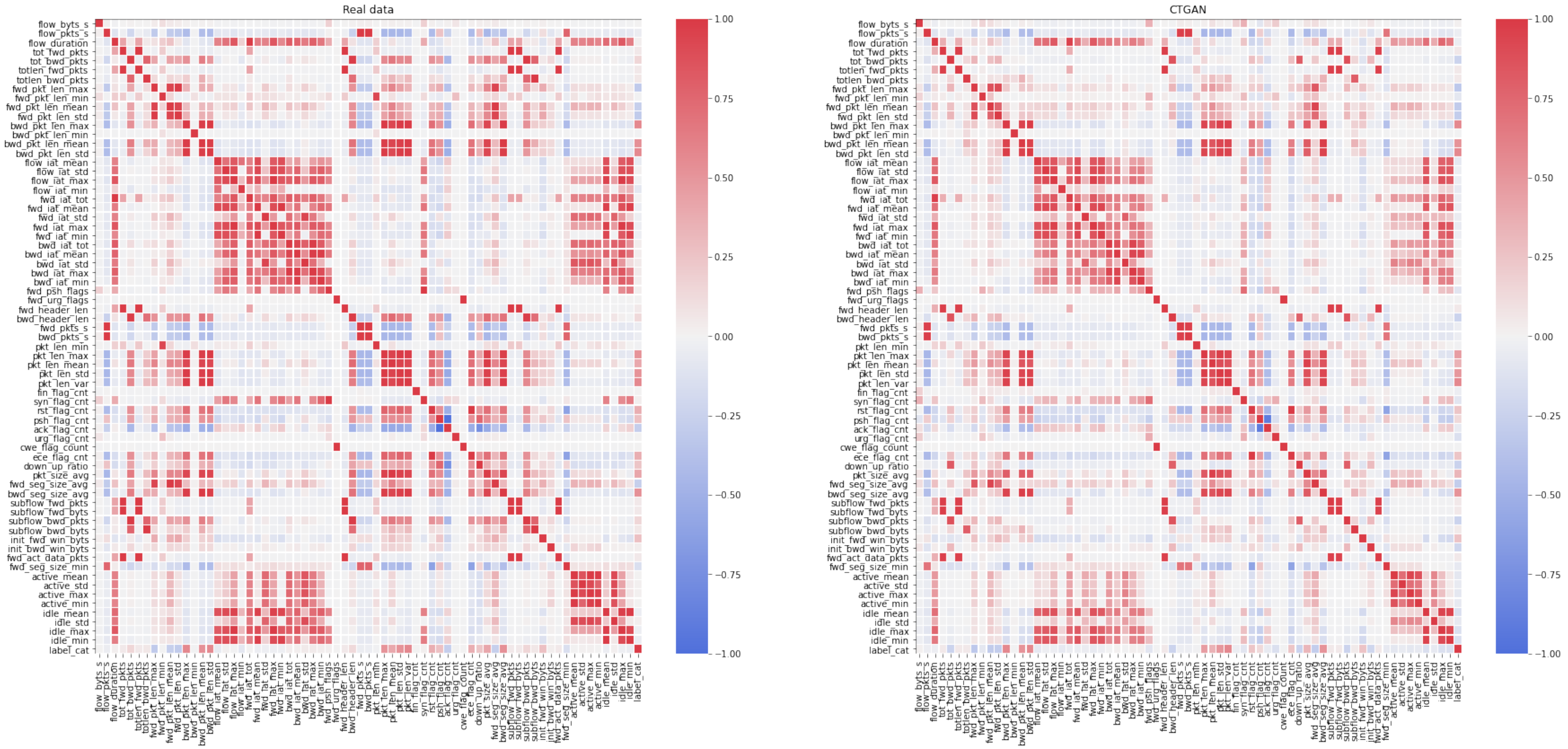


Figure A.4: CTGAN pairwise correlation matrix comparison with real data