Generative adversarial network usage toward Building Models to Detect Intrusions and Malicious Activity on the Network

1st Hosam Mahmoud[a], 2nd Kareem Waly[b], 3rd Mohamed Elesawy[c], 4th Sondos Ali[d]

[a] 1st School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, Canada, hmahm074@uottawa.ca

[b] 2nd School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, Canada, kwaly008@uottawa.ca

[c] 3rd School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, Canada, meles043@uottawa.ca

[d] 4th School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, Canada, sali207@uottawa.ca

Abstract

There are several uses for Artificial Intelligence(AI), especially in the area of Cybersecurity, Nowadays Cybersecurity is important due to the increasing internet usage, apps, and websites more security solutions are needed to keep the system aware of malicious activities, Intrusion Detection Systems (IDS) is an indispensable part in the security framework in communication systems and networked computing, it's a software app that examines what occurs to the network or has occurred during the performance of a task and seeks to identify or locate signs that a certain region of a computer has been improperly utilized, any illegal activity or violation is often alarmed to the administrator, traditional methods suffer from low detection rate, high false alarm rate, difficulty in dealing efficiently with the huge amount of data due to time changing network environments, and according to the development of deep learning technologies, deep learning performs more accurately than machine learning algorithms when learning large amounts of data, especially in case of learning from imbalanced data. In this Paper, In this paper, the KDDCUP-99 dataset is chosen as the dataset of interest because it includes a wide variety of intrusions, and we implemented three GANs architectures to generate new normal samples, and then make the discriminator classify the generated data whether real or fake, the results demonstrate that GANs architecture can achieve superior performance compared with traditional ML methods where it scored 96% macro average F1-score with a multiclass classification only on 10 percent of data.

Keywords

Artificial Intelligence (AI); Network security; Generative Adversarial Network (GAN); Intrusion Detection System (IDS); Machine Learning (ML); Deep Learning (DL); Deep Convolutional Generative Adversarial Networks (DCGAN); KDDCUP-99.

1. Introduction

With the emergence of new Internet technologies such as data sharing, online payment, and the spread of the Internet of Things (IoT), network security has become more complex, and the need to secure personal data has increased. As different systems are constantly facing the risk of trying to steal their data and trace and track their critical information from cyber attackers using many methods. One of which is spreading malware in the network to extract its data, Therefore, solutions like Intrusion Detection Systems (IDSs) have become essential for spotting and defending against network attacks brought on by malicious network traffic. the IDSs are used on the basis of two fundamental approaches as shown in **fig.1.**, first, the identification of anomalous activities as they typically occur when normal or malicious behavior turns, and second, the misuse detection by looking for unauthorized "signatures" of those identified malicious assaults and classification vulnerabilities. The main objective of IDS is to distinguish between Normal and malicious network records.

Since machine learning is a double-edged weapon that can both protect against vulnerabilities or create them, and after generative adversarial networks (GANs) have proved it's successful in generating data originating from different distributions and types, so we tried to solve the imbalance between normal and malicious classes using different GANs architectures.

In this work, we use multiclass classification and a binary classification technique with DCGAN and CTGAN networks for the models to distinguish between normal and anomalous data by generate a fake normal data using the generator and after that make the discriminator classify the generated data whether real or fake.

In the related literature, authors always work on the images to get the most out of the GANs network by converting the tabular data to images first and then working on it as the case of the images but in our work, we decide to work on data as a tabular with only 10 percent of data to show how the results will change compared with the others work.

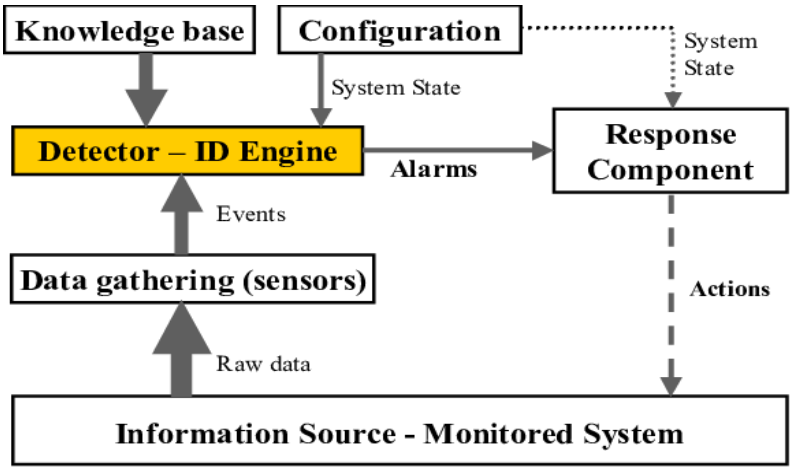


Figure 1. The basic architecture of the Intrusion Detection System [5]

2. State-of-the-art review

2.1. Our Baseline Paper [1] (Chen & Jiang, 2019)

The problem that the authors of that research paper [1] tried to solve is to find a new way to detect cyber-intrusions that can endanger the security of enterprise systems and devices. The problem is that traditional ways like SVM and Decision trees (statistical learning models) don't give good results with imbalanced data, while anomaly detection algorithms like Local Outlier Factor, Robust Covariance, and Isolation Forest is handling imbalanced data better but these algorithms will face some difficulties when dealing with high dimensional data. The authors solved this problem by developing a model that is derived from Bi-GAN architecture, with some modifications. The experiments that the authors of this research paper did Are:

- The authors have trained different models on the KDD-99 dataset which is used for the testing of cyber-intrusions detectors.
- The authors tuned and evaluated their model, and the authors also did some comparisons between the models.

The main conclusion of this paper is that the authors have developed a model that outperforms any of the previous GAN-based models on cyber-intrusion detection problems, and also their model is faster than the previous GAN-based models on this problem.

This paper [1] aims to develop a model that is able to detect cyber-intrusions and give a better performance than the previous models in terms of speed and accuracy, The authors have achieved their goal by developing

a model that is derived from Bi-GAN, and this model not only transforms the latent samples into real in the generator but also simultaneously transforms the real into their latent status via an encoder, and also, the authors have added intermediate layers in the discriminator to optimize the feature extraction. and this helps a lot in increasing the speed of their model. This model helped them achieve the best performance from the previously proposed models in terms of accuracy and speed. Before proposing their model, the authors mentioned a review of different previous works and what is the problems for each method. The authors have mentioned their goal for future work, and they plan to look into how timing affects intrusion detection.

The authors in paper [1] used a model which is derived from Bi-GAN that uses the encoder to reflect the real samples into their latent state. Then they used different approaches which helped in enhancing the GAN's training and they found that the minimax objective is the most often utilized. Then, they did Anomaly Assessment to detect the anomalies and evaluate the level of the produced output. Anomaly score has been defined in several ways, but they are all essentially the same. The authors suggested two criteria to select the abnormal sample based on its anomaly score: The first method is appropriate for online detection since all that is required is a threshold that may be determined by experience; there is no need to understand the ratio of normal samples to abnormal samples. The second method, which is based on the ratio of normal samples to aberrant samples, is typically used to test datasets and assess the model's performance.

The author’s experiment is based on the KDD-99 dataset which is used to test cyber-intrusion detectors. Each sample in this dataset contains 41 features. The authors did a preprocessing on the dataset, then they applied many models, including typical anomaly detection techniques like Isolation Forest, BiGANFM and OC-SVM to detect the abnormal samples. They did a comparison between these traditional methods and their model and found that their model achieved results higher than the other method.

Figure 2. Performance of paper model on KDD-99 with 20%, 10%, 5%, 1% contaminate rate

Model	Precision	Recall	F1
Isolation Forest	0.4415	0.3260	0.3750
OC-SVM	0.7457	0.8523	0.7954
DSEBM-r	0.8521	0.6472	0.7328
DSEBM-e	0.8619	0.6446	0.7399
AnoGANFM	0.8786	0.8297	0.8865
BiGANFM	0.6578	0.7253	0.6899
This paper Model	0.9324	0.9473	0.9398

Table 1. Performance on the KDD-99 Dataset

They have overcome the problem of working with imbalanced data by not using traditional statistical learning models. The other problem that the authors have overcome is working with high dimensional data by not using anomaly detection algorithms like Local Outlier Factor, Robust Covariance, and Isolation Forest, instead they used Generative Adversarial Network (GAN) to overcome this problem. The third problem is the speed of the model and to overcome this problem they used an encoder that inverse the mapping from real space to latent space. They have discussed and explained these problems clearly and efficiently. They have conducted different experiments, tried different models, and made good comparisons between these models in terms of accuracy and speed, and they provided the results of these comparisons on tables and graphs.

We think that the authors of paper [1] have done pretty good work with a perfect number of mentioned details.

2.2. Paper Two [2] (Yang et al., 2019)

The problem authors of that research paper [2] tried to solve the weaknesses in the traditional machine learning model by finding a model that can deal with high-dimensional data samples and efficiently detect intrusion in the network. The authors solved this problem by developing a model that used NIDS which provides protection from Internet-based attacks, and long short-term memory (LSTM) to automatically learn the features of network intrusion behaviors, recurrent unit based to enable intrusion detection in real time The main conclusion of this paper is that the authors have developed six popular learning models (including SVM, Random-Tree, Random Forest, NB Tree, Naive, Bayes, and J48) and all of these models can only

achieve maximum accuracy 94% on KDD-99 and 83% on NSL-KDD wherever the SRU-DCGAN get 99.73% on KDD-99 and 99.62% on NSL-KDD

The authors aim to generate a network intrusion detection model based on simple recurrent unit based (SRU) and DCGAN which can detect cyber-intrusions and give a better performance compared to any Machine learning model.

The authors in paper [2] used an SRU-DCGAN model and six other methods to detect the intrusions in the network an SRU-based multichannel network intrusion detection model is applied. It beats the LSTM algorithm in terms of classification speedup and automatically extracts the features through repeated multi-level learning. Additionally, this approach increases the effectiveness of categorization and the precision of detection of network danger behaviors. The generation of cyber threat samples using a generative adversarial model is suggested. New training samples are produced using this technique. It addresses the issue of inadequate and imbalanced samples that standard intrusion perception methods frequently confront. Additionally, it lowers the number of false alarms and increases system detection rates. To guarantee that the data can be processed by our model with great efficiency, a preprocessing approach of mapping the network data using Mahalanobis Distance is recommended. This method provides high-quality data input for the proposed deep learning model.

They used 3 datasets in the experiment: the KDD'99 dataset, the NSL-KDD dataset Firstly, in the experiments of the SRU-DCGAN model on the KDD-99, The Hyper-parameter settings were hidden units = [30, 70, 30, 30], steps = 400000, batch-size = 20000, epoch = 500, Learning Rate $\eta = 0.001$ as shown in **fig.3** and the SRU-DCGAN model was the highest accuracy model compared to all other methods.

Secondly, in the experiments of the SRU-DCGAN model on the NSL-KDD, as shown in **fig.4** and the SRU-DCGAN model was the highest accuracy model compared to all other methods. We only illustrate a sample from their experiments, they really tried a lot of experiments to be sure that their model doing great on different datasets.

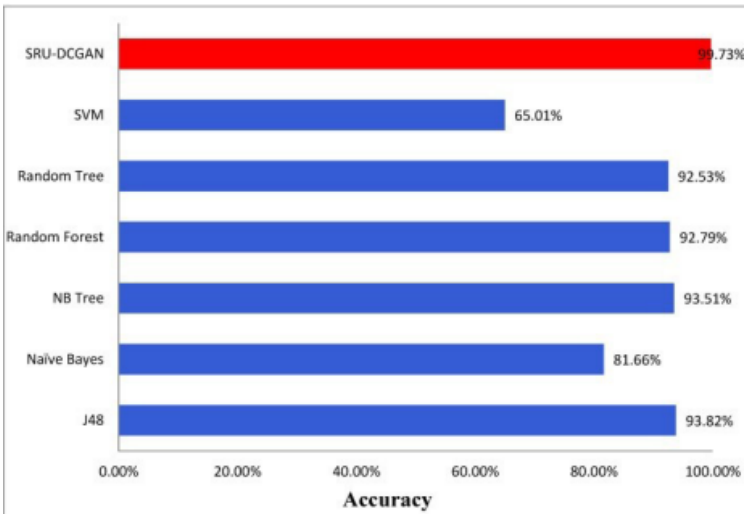


Figure 3. Comparing experiment with SRU-DCGAN model and the classical algorithm on KDD-99.

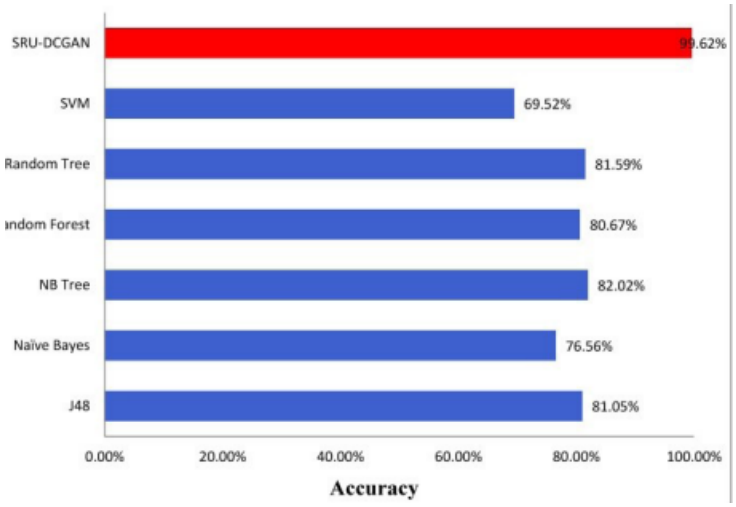


Figure 4. Comparing experiment with SRU-DCGAN model and the classical algorithm based on NSL-KDD dataset.

The whole paper was well-written and very clear, anyone without any domain knowledge can understand it, not only the specialist, they use graphs and tables to illustrate their results

It was difficult for us to find a weak point in this paper, they have already covered all aspects of this project and have tried many models on many datasets, and they plot their results in comparison forms.

2.3. Paper Three [3] (Salem & Taheri, 2018)

The problem authors of that research paper [3] tried to understand and solve the problem of imbalance in the distribution, especially the imbalance between normal and anomaly data instances using the GANs networks to learn how to transform normal to anomaly data and use it to generate anomalies from the normal data.

The authors solved this problem by converting the data from the tabular to images to have the ability to use a Cycle-GAN network which helped them to learn the transformations between normal and anomalous host-based data.

The main conclusion of this paper is that the authors used Cycle-GAN to transform images of normal data into images of anomalous data. Following that, the created data is merged with the original dataset to train algorithm (Multilayer Perceptron algorithm) to recognize anomalies.

The authors of this paper aim to create anomalies and utilize GANs to learn more about the distribution of those abnormalities, it is possible to enhance the performance of host-based intrusion detection systems by appending generated anomalies data to the original dataset.

The authors in paper [3] converted the host-based intrusion data into images to take advantage of Cycle-GAN, they used Cycle-GAN network to train two distributions one for normal data and another for anomalous data to the algorithms know how to transform normal to malicious data for the training they partitioned data to training and test where the test set ratio is 30% of the whole data and the rest as the training set. They append the generated data to the original source data, which help the training set solve the imbalanced problem between the normal and anomalous classes of data. The new balanced training set used to train Multilayer Perceptron (MLP) algorithms, also they trained two more MLP networks on the imbalanced training set and balanced training set generated using SMOTE to compare their results to each other. To guarantee that the Cycle-GAN network perfectly fit this case the structure and hyperparameters are not changed, only the learning rate a bit decreased to aid the classification process.

The dataset used in this experiment: the ADFA-LD dataset, Firstly, in the experiments of the Cycle-GAN network with TensorFlow on the ADFA-LD after converted it into images, and changing the learning rate η, the model trained for 330,000 epochs and the loss functions F, G, D_x, and D_y recorded every 10,000 epochs as shown in **fig.5**.

From the **fig.5** we can conclude that in training, the G and Dx functions exhibit greater volatility than the F and Dy functions. When evaluating the realism of the generated images based on the smaller number of anomaly images, the G function changes the input to the anomaly distribution and Dx, which is more volatile. The challenge authors faced during the GAN training was the volatility in the loss functions, and they overcome this challenge using Cycle-GAN which saved record every 10,000 epochs to be evaluated later, saved models are exported after that to be help in generate synthetic malicious data.

Models at first have no knowledge of distributions, it generates random value of each pixels, but after 10,000 epochs it learnt how to preserve the white portion of the image and is producing images that are nearly grayscale, after 150,000 epochs the generated data become more sharper border above the white segmentations

Finally, after each individual graph's data has been generated, the generated data is merged with the original data to train the MLP and the AUC achieved 0.71 as shown in **fig.6** at 210,000 epochs, the power of detected unseen malicious scored 80.49%, and the comparison of the classification results shown in **Table.2**.

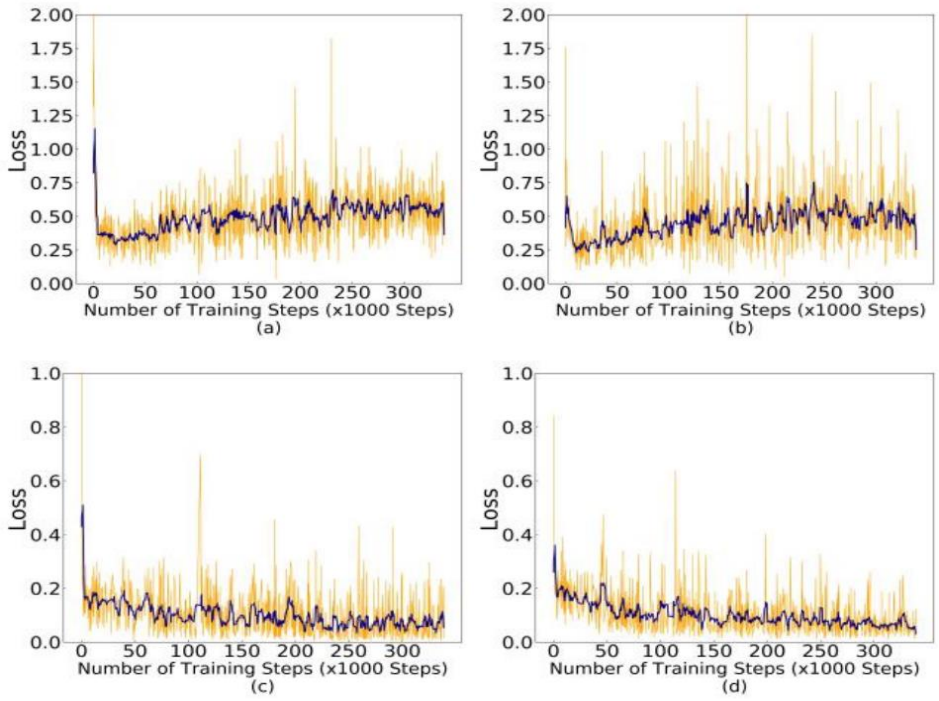


Figure 5. loss functions a) F, b) G, c) Dx, d) Dy

Approach	TP	TN	FP	FN	Recall	F1	AUC
Imbalanced Data	14	415	31	68	17.07	22.05	55.06
SMOTE	67	250	196	15	81.71	38.84	68.89
Cycle-GAN	66	277	169	16	80.49	41.64	71.30

Table 2. comparison of the classification results

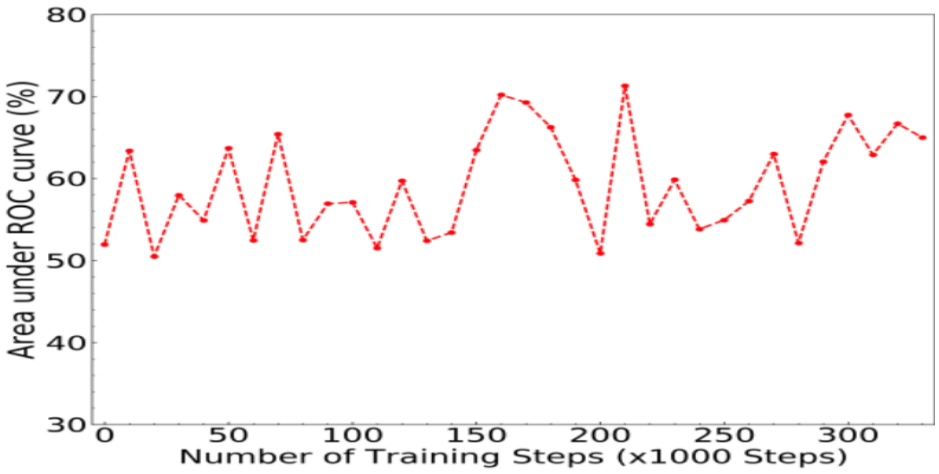


Figure 6. AUC record for each model

Anyone without any domain knowledge can understand it, not only the specialist, but they also use graphs and tables to illustrate their results, They Used the AUC metric for the evaluations and that's seems to be perfect for that case

They didn't mention any future work, and for the methodology section, they only use one model (MLP) without declaring why they chose that particular model, the paper as a whole need to be more captivating to hold reader's attention.

2.4. Paper Four [4] (Kaja et al., 2019)

The problem that the authors addressed that every organization has a daily war against the harmful attacks, although the new technologies bring major advancements, but also new challenges. So, security breaches may be disastrous to a corporation and sometimes result in Un survivable scenarios.

The authors solved this problem by building an intrusion detection system which is consists from two-stages, the authors solution detects attacks using K-Means in the first stage, and then in the second stage the authors used different supervised learning models to classify such attacks and reduce the number of false positives.

The main conclusion of this paper is that the authors have used KDD dataset with 4 steps preprocessing, after that the authors have built their anomaly-based intrusion detection system which consist from two-stages, the first one is to train an unsupervised model to detect the attacks records form the normal ones, and the second stage is for training a supervised learning algorithm to be able to classifies such attacks into their types. They

have tried different machine learning algorithms for the stage two like: Random Forest with 99.97% accuracy, J48 Decision tree with 99.95% accuracy, AdaBoost with 97.86% accuracy and Naive Bayes with 92.75% accuracy.

The Authors goal is to build an efficient intrusion detection system with minimal false positive rate as possible as they can, and with high accuracy compared to the state-of-the-art previous works and with less Computational time.

The authors have built their intrusion detection system by firstly, choosing the dataset which is KDD dataset and applying 4 steps preprocessing which is:

1- Remove the low variance features because such features don't provide significant information.

2- Remove closely correlated features which helping in removing the bias from the model.

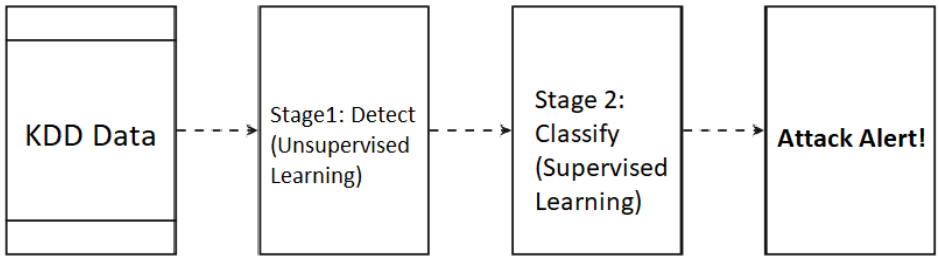
Feature A	Feature B	Correlation between A and B
destination_srv_serror_rate:	destination_serror_rate:	0.95
destination_srv_rerror_rate	rerror:	0.93
dst_srv_rerror_rate	server_rerror_rate:	0.95
rerror:	service:	0.91
nr_compromised:	nr_root	0.99
serror	server_serror	0.97
server_rerror	rerror	0.97

Table 3: correlated features.

3- Minimizing feature similarity and maximizing dimensionality reduction by using Least Square Regression Error (LSQE).

4- Analyzing features relationships by using by using Maximal Information Compression Index (MICI).

After the preprocessing of the data, the authors have built their system in two stages, the first stage is to train an unsupervised model to detect the normal connections from the attack ones, and for the second stage they tried four different supervised machine leaning models to classify the attacks to their types and to reduce the false positive rate as possible as they can.



For the first stage the authors have used k-Means model to cluster the data into two categories “attack” connections or “normal” connections. because the authors can in this stage afford false positive attacks but they don’t want to miss any true positives, the algorithm is intentionally designed to show bias towards "attack" connections.

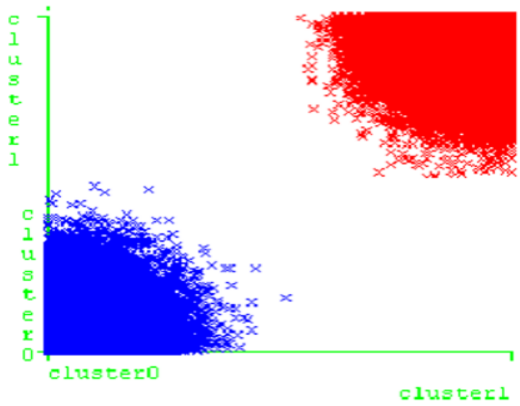


Figure 7. Visual of the k-means clusters.

For the second stages they have tried different four models:

- A decision tree-based algorithm called **J48** is used to classify data. this algorithm, which is based on the C4.5 decision tree, constructs decision trees using information entropy.
- In order to combine decision trees, **Random Forest** employs an ensemble learning technique known as bagging, a machine learning ensemble meta-algorithm designed to increase accuracy, decrease variance, and reduce over-fitting.
- **Adaptive Boosting (AdaBoost)** is a machine learning ensemble algorithm that builds its ensemble in such a way that prediction errors are reduced at each layer. The subsequent models concentrate on correcting flaws in the earlier models. These are comparable to common boosting algorithms. Adaptive Boosting adds a series of short decision trees until the performance is not further enhanced.
- Another algorithm chosen for the second stage of the intrusion detection system is **Naive Bayes**. Based on the Bayes theorem, this algorithm uses probabilistic classification. This algorithm's goal is to determine the probability that each class will contain a given feature and to return the class that is most likely to contain that feature.

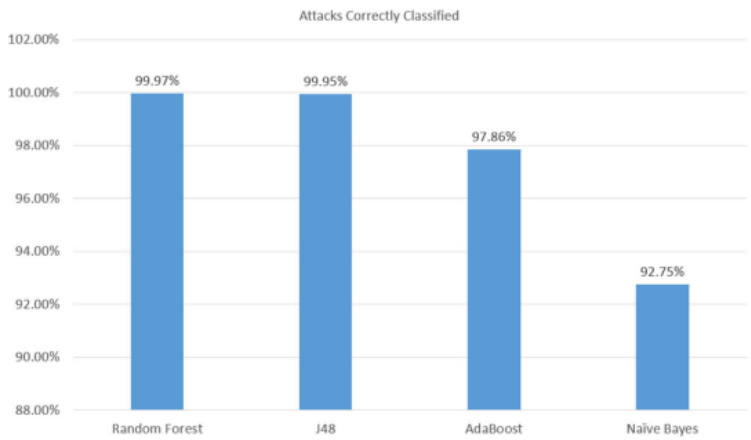


Figure 8. Summarized Results.

Algorithm	Performance	Computational time
Random Forest	**99.97%**	4.29e-5
J48	99.95%	**4.02e-6**
Adaptive Boosting	97.86%	3.71e-6
Naïve Bayes	92.75%	1.48e-5

Figure 9. performances and computational times.

As we see from the above figures that the **Random Forest** gave the best classification performance but the **J48 Decision Tree** gave the best computational and training time.

The Authors made a good comparison between the four different models, providing different visualization to make the comparison clear. The authors explain each model in a clear way and provides a visualization for each model to illustrate the models results. The authors are very organized and illustrate their topics in a simple and clear way.

3. Proposed Method

3.1. Binary Class Classification

First, we focus on the binary classifying problem. The labels for normal and anomalous data points will be "0" and "1" respectively, we tried to build different ML models that can distinguish between two classes and after that we want to go deep into GANs architecture before we see their effect on our result. The GAN consists of two

networks the generator produces fake samples based on the normal class and the discriminator receives samples from both generator and the dataset and it discriminates between original data and fake data produced by the generator, for that phase we implement two GANs models CTGAN and DCGAN, the Conditional Tabular GAN it generate a tabular data depending on a certain conditions that we give for our model, and the Deep Convolutional GAN 's more deep compared to the standard GAN, it also works on images, but we customized it to work on tabular data. The effectiveness of the generator's trick on the discriminator is measured by its loss. It makes sense that the discriminator would label the fake data as a real if the generator is functioning properly.

3.2. Multiclass Classification

First, we focus on the multiclass classifying problem. We transformed the 23 labels into only 5 labels using [6] where "0" for normal: normal traffic without malware, "1" for DOS: denial-of-service, e.g., syn flood; "2" for R2L: unauthorized access from a remote machine, e.g. guessing password; "3" for U2R: unauthorized access to local superuser (root) privileges, e.g., various ``buffer overflow" attacks; and "4" for probing: surveillance and other probing, e.g., port scanning. In this phase we tried to build different ML models that can distinguish between 5 classes and after that we want to go deep into CTGAN architecture before we see their effect on our result. The CTGAN as mentioned before but here we implemented it manually.

4. Dataset

DARPA used network traffic that had been captured in 1999 to create the KDDCUP-99 dataset. It is being preprocessed into 41 features " 9 discrete and 32 continuous ". The KDDCUP-99 data collection includes four groupings of features: Basic features are listed from 1 to 9, followed by Content features from 10 to 22, Time-based traffic features from 23 to 31, and Host-based traffic features from 32 to 41, it consists of 4,898,430 records with the four basic types of attacks: DoS, R2L, U2R, and Probe. The feature importance bar plot shown in **fig.9.,** the correlations between features shown in **fig.10.,** and the histogram for each shown in **fig.11.**

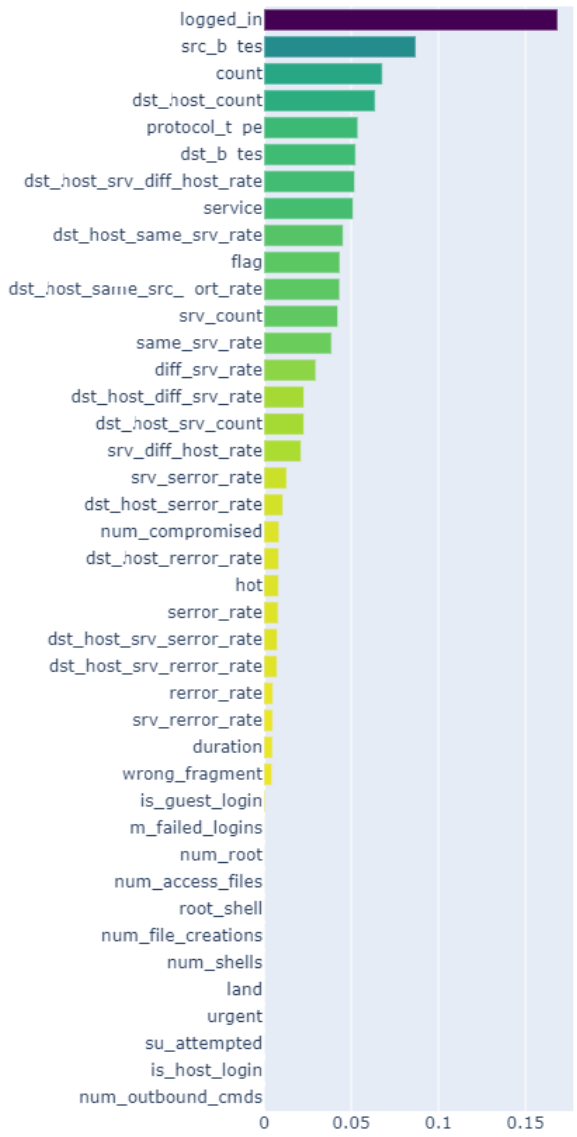


Figure 9. Feature importance Bar Plot of the Attacks classification in KDDCup-99 data set.

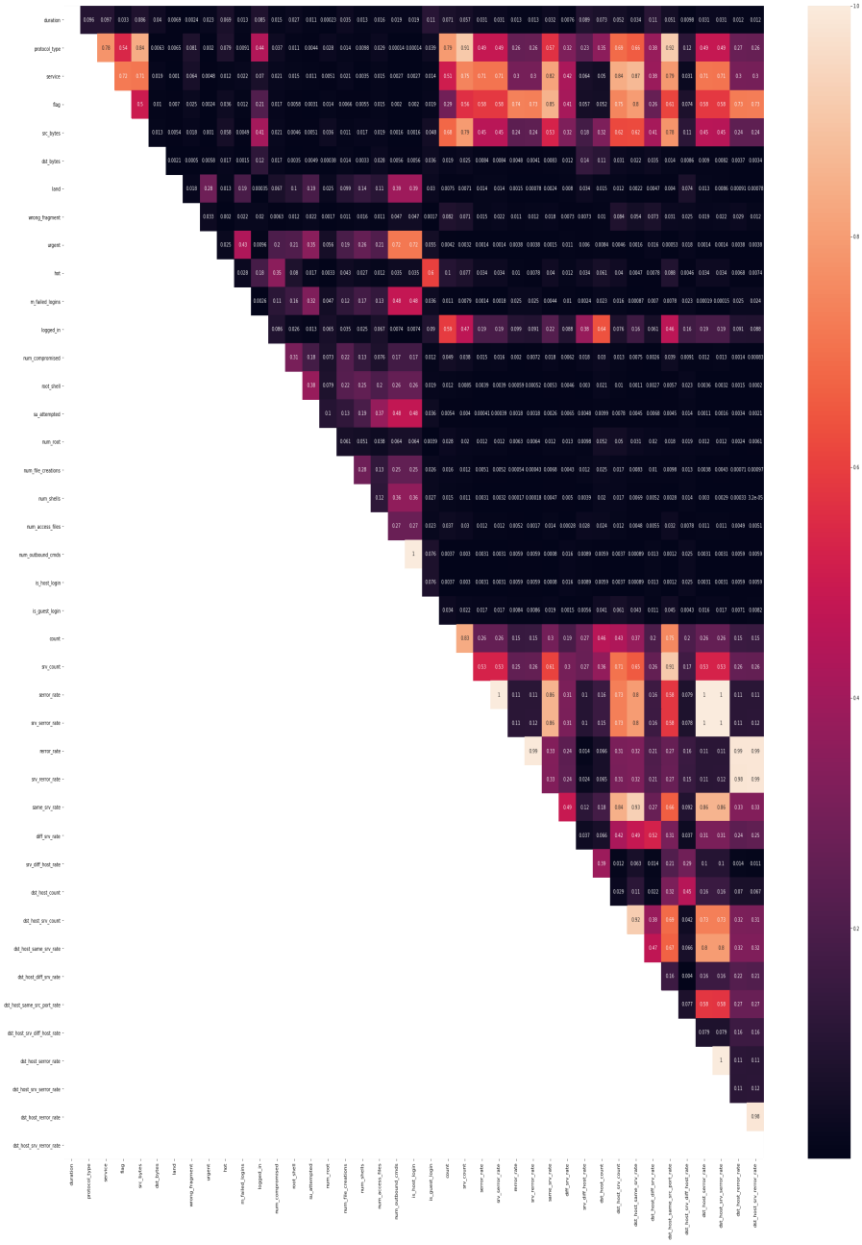


Figure 10. correlations between features.

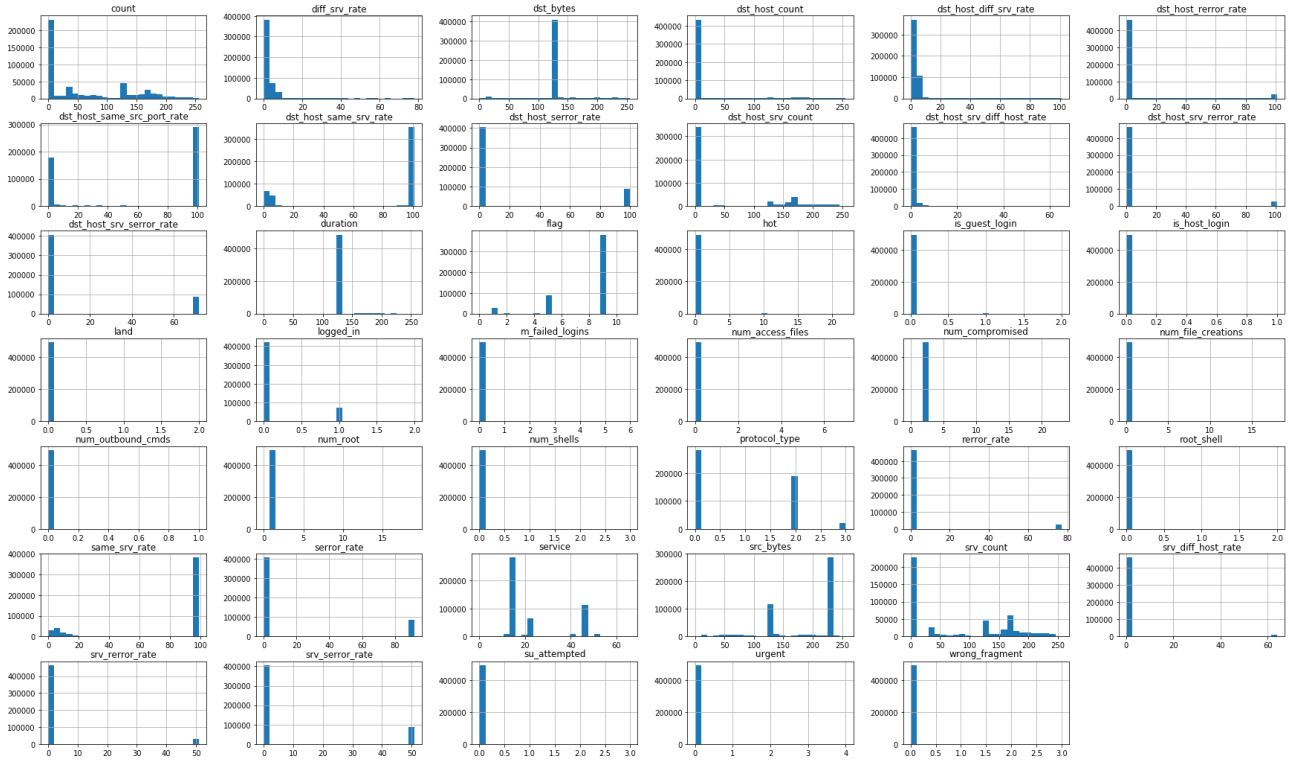


Figure 11. histogram for each feature.

5. Methodology

5.1. Data Pre-Processing:

First, we renamed the column names using the Names that were provided to us From the Dataset websites and then we checked for null values then we split the Data into Two parts categorical and numerical values.

For the categorical values, we applied Label Encoder on them So that we can transform them into numerical values so that the machine learning models will be able to process the data.

For the numerical values, we applied the Minmax scaler to get rid of anomalies and skeins in the data.

5.2. Binary Class Classification

First, we applied Data preprocessing on the data then we transformed the 23 labels into 2 labels Normal and Attacks as shown in PART 3. Then we applied Five different models Namely: DecisionTree, RandomForest,

AdaBoost, Bagging, and ExtraTrees Classifiers without Feature Selection then we chose the champion model which was RandomForest to further apply Feature Selection using feature important and select the columns that Pass 0.03 Values that we determined to be optimal values using trial and error also improved the F1 score for the champion model.

5.3. Multiclass Classification:

First, we applied Data preprocessing on the data then we transformed the 23 labels into 5 Different labels "0" for normal, "1" for DOS, "2" for R2L, "3" for U2R, and "4" for probing as shown in PART 3. Then we applied Five different models Namely: DecisionTree, RandomForest, AdaBoost, Bagging, and ExtraTrees Classifiers without Feature Selection then we chose the same column name as the binary classification, but it made the model less accurate which is logical since the multiclass problems needs more column to differentiate between the classes.

5.4. GAN Architecture

5.4.1. Binary Class Classification

In this phase we have implemented a conditional GAN to generate fake normal data to test our best binary classifier machine learning model (Random Forest) if it will be able to successfully classify these fake normal data as an attack or not.

5.4.2. Multiclass Classification

In the multiclass part, we implemented a CTGAN using Pytorch Framework and then we take the Training data and Trained the CTGAN on the condition of learning only the normal Data. And considered the Fake Data as malicious data since they are not real and injected the GAN data into the original test data and applied it to the Champion model and then we noticed that the accuracy had dropped a lot after injecting the GAN Data finally we combined both the champion model with the Discriminator of the GAN where it removed all the Fake Value we Found that it also removed Some of the malicious Data that the Champion model Couldn’t Detect.

6. Evaluations

In this section we will discuss the evaluation metrics which was F1- score because we have imbalanced data between their classes and a low number for one of the two inputs makes the F1-score significantly more sensitive. It's therefore excellent if you wish to balance the two also, we are more interested in the FP, also the F1-Score gives an indication of the overfitting so it’s the suitable metric for our case.

7. Baseline Methods

Our baseline method was the model the authors of paper [1] implemented because they developed a model that outperforms any of the previous GAN-based models on cyber-intrusion detection problems, they applied their experiment based on the KDD-99 dataset too, and they applied many models, including typical anomaly detection techniques like Isolation Forest, BiGANFM and OC-SVM to detect the abnormal samples. They did a comparison between these traditional methods and their model and found that their model achieved results higher than the other method as shown in **Table 1**.

8. Results

In this section we will discuss

8.1. Binary Class Classifications

For the binary classification methods we building 5 different models (DecisionTreeClassifier, RandomForestClassifier, AdaBoostClassifier, BaggingClassifier, ExtraTreesClassifier) trained on original training data after splitting data into training, validation, and testing 70:20:10, respectively and the champion models was the RandomForestClassifier gives an accuracy 99.975% and an F1 score equal 100% as shown in **fig.13** after that we used this model for the feature selection and that were the most important feature ("protocol_type", "service","flag", "src_bytes", "dst_bytes", "logged_in", "count", "srv_count", "same_srv_rate", "dst_host_count", "dst_host_same_srv_rate", "dst_host_same_src_port_rate", "dst_host_srv_diff_host_rate") we trained the ML model again on the selected feature (13 features only of 41 features) the Random Forest Classifier gives an accuracy 99.965% and an F1 score equal 100%, as shown in **fig.18** it provides a superior level of accuracy

and I can see that the number of false positives is the lowest (only 38) when compared to the other models, indicating that it does not have malicious flaws. And then we start training the two GANs architecture, to train the Generator, we first feed it with the normal data and then make the discriminator classify the generated data whether real or fake and after the data is generated, we retrained our champion model (RandomForestClassifier) using the original data combined with the generated data. Generator uses binary cross-entropy to determine the generator loss and is constructed of 5 Dense Layers with LeakyReLU activation function layers and 2 BatchNormalization layers. A two-class classification model's performance is judged by binary cross-entropy loss, which produces a probability value between 0 and 1. In an ideal model, there would be no loss as shown in **fig.24** when the number of (1) approximately zeros and the number of (0) equal 97277.

8.1.1. Results of Training ML Models without Feature Selection

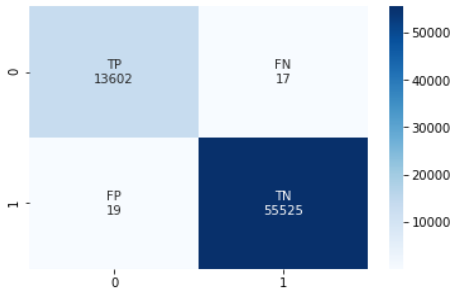


Figure 12. DecisionTreeClassifier

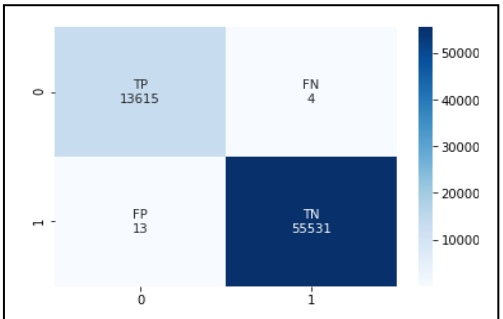


Figure 13. RandomForestClassifier

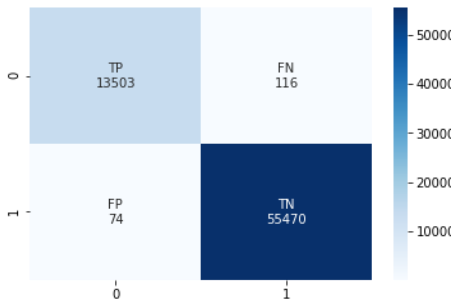


Figure 14. AdaBoostClassifier

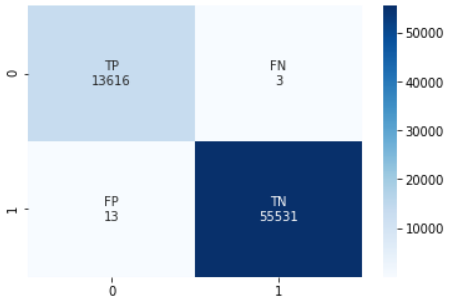


Figure 16. ExtraTreesClassifier

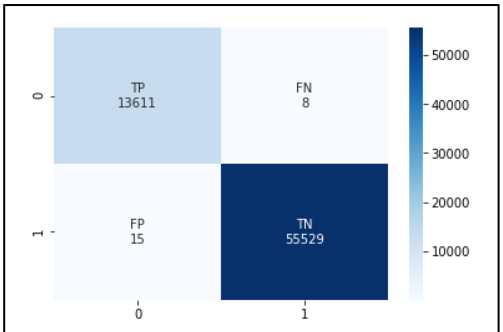


Figure 15. Bagging Classifier

8.1.2. Results of Training ML Models after Feature Selection

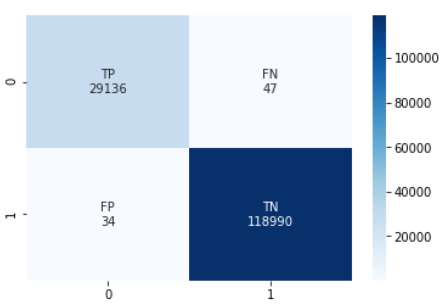


Figure 27. DecisionTreeClassifier

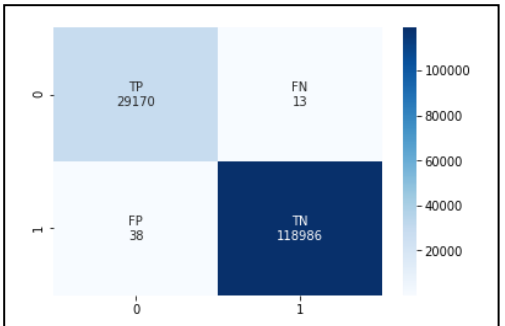


Figure 18. RandomForestClassifier

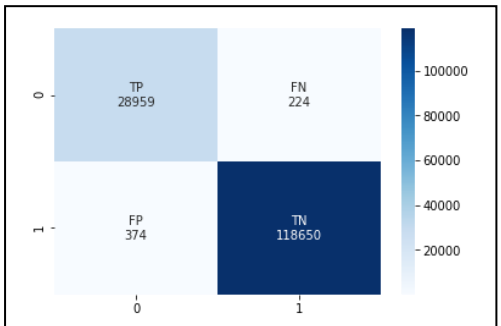


Figure 19. AdaBoostClassifier

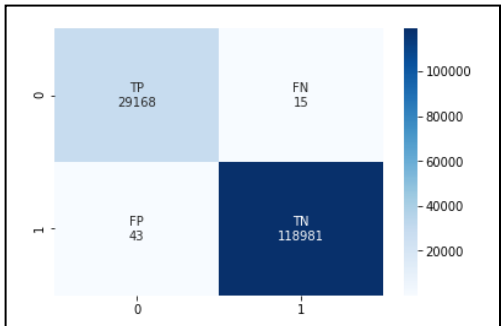


Figure 21. ExtraTreesClassifier

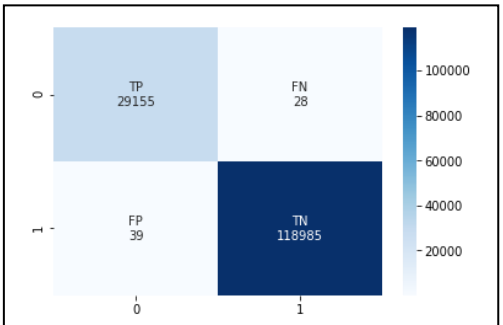


Figure 20. Bagging Classifier

8.1.3. The Evolution of training DCGAN and CTGAN

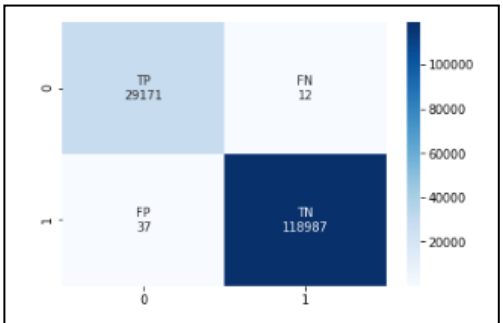


Figure 12. RandomForestClassifier for DCGAN

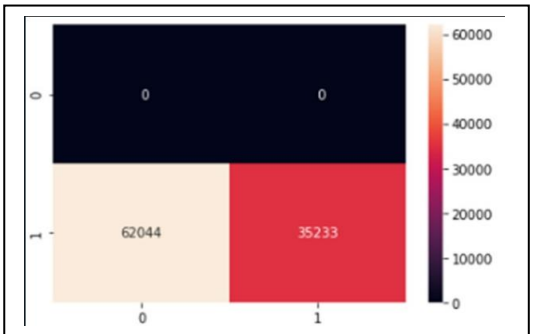


Figure 23. RandomForestClassifier for CTGAN

```
prob =-0.2 and  : accuracy=1.0
The pred_value.count(0)=97277 : and the pred_value.count(1)=0

prob =-0.1 and  : accuracy=0.9992906853624186
The pred_value.count(0)=97208 : and the pred_value.count(1)=69

prob =-0.05 and  : accuracy=0.9374261130585853
The pred_value.count(0)=91190 : and the pred_value.count(1)=6087

prob =0 and  : accuracy=0.5163399364700803
The pred_value.count(0)=50228 : and the pred_value.count(1)=47049

prob =0.005 and  : accuracy=0.5029965973457241
The pred_value.count(0)=48930 : and the pred_value.count(1)=48347

prob =6e-06 and  : accuracy=0.5163193766255127
The pred_value.count(0)=50226 : and the pred_value.count(1)=47051
```

Figure 24. Accuracy for DCGAN

8.2. Multiclass Classifications

For the Multiclass Classification methods we built 5 different models (DecisionTreeClassifier, RandomForestClassifier, AdaBoostClassifier, BaggingClassifier, ExtraTreesClassifier) trained on original training data after splitting data into training, and testing 80:20, respectively and the champion models were the RandomForestClassifier gives an accuracy 99.97 % and an F1 score equal 96% as shown in fig.26 after that we used this model for the feature selection and that was the most important feature That we Got from the Binary Class classification we trained the ML model again on the selected feature (13 features only of 41 features) and these Features Had the highest performance where the Random Forest Classifier gives an accuracy 99.9 % and an F1 score equal 90 %, as shown in fig.31 and this showed that model needed all the columns so that it could have the ability to distinguish between the 5 class So the Champion model was Random Forest Classifier with all the columns. And then we Trained the CTGAN on the normal Data and Made the Generator create 100000 Data Sample and then We labeled them as One since they are not normal Data and we inject these 100000 Data sample with the Test data and as shown in fig.35 we see that the accuracy Drops signafly to reach 63% F1 score then after applying the Discriminator That discards the Fake Data as an Extended layer over the Champion model it moves back the accuracy to reach 96% as shown in fig.35 also we can notice that both the TP and TN accuracy had improved From the Confusion matrix.

8.2.1. Results of Training ML Models without Feature Selection

Figure 25. DecisionTreeClassifier

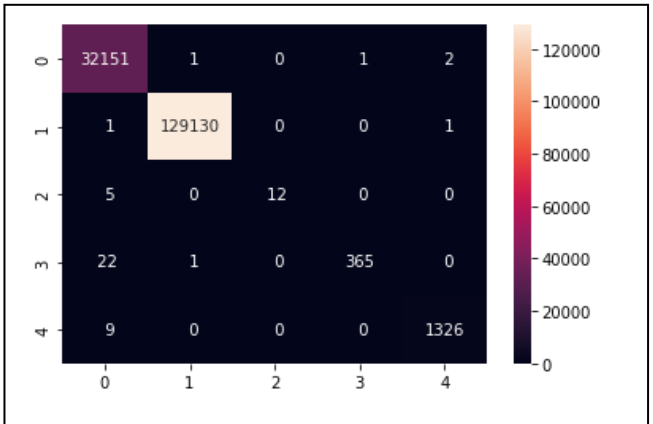


Figure 26. RandomForestClassifier

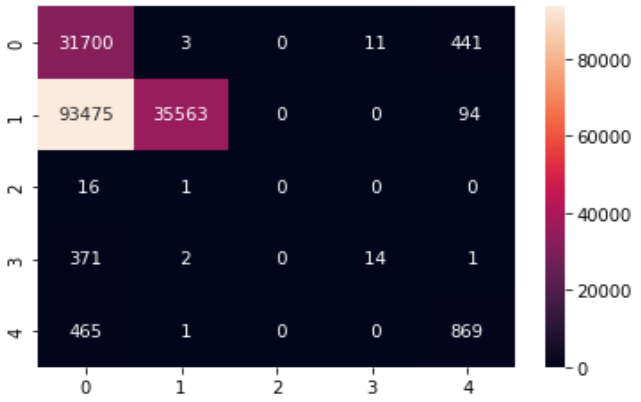


Figure 27. AdaboostClassifier

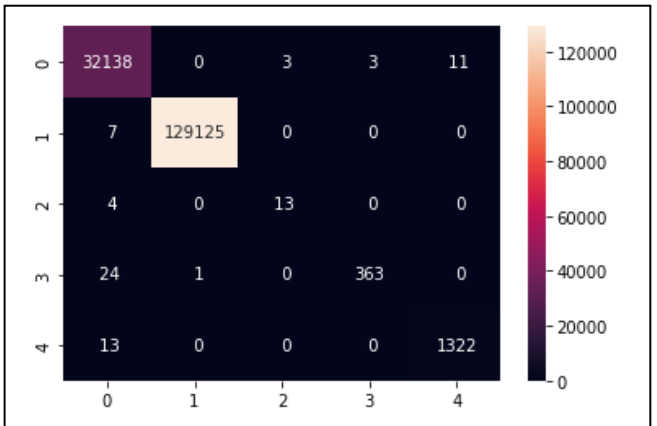


Figure 28. Bagging Classifier

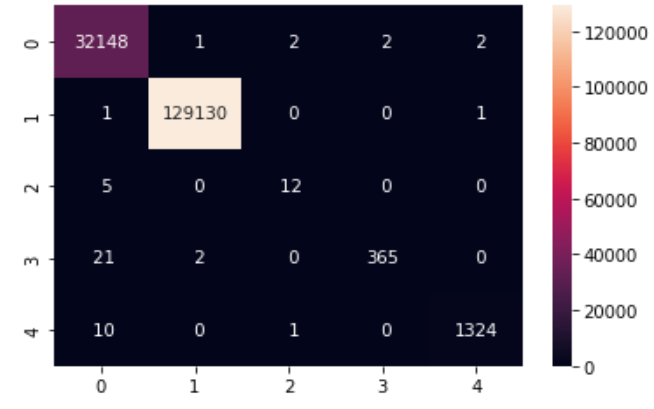


Figure 29. ExtraTreesClassifier

8.2.2. Results of Training ML Models after Feature Selection

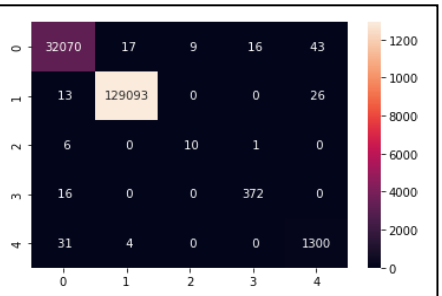


Figure 30. DecisionTreeClassifier

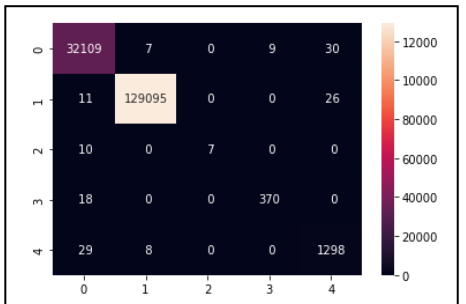


Figure 31. RandomForestClassifier

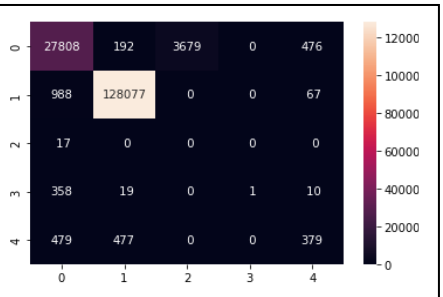


Figure 32. AdaboostClassifier

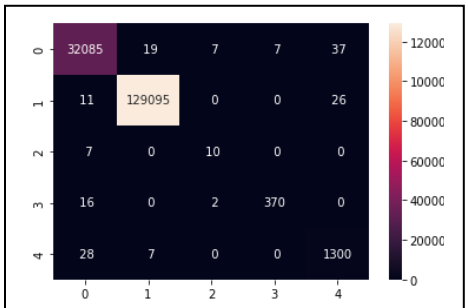


Figure 33. Bagging Classifier

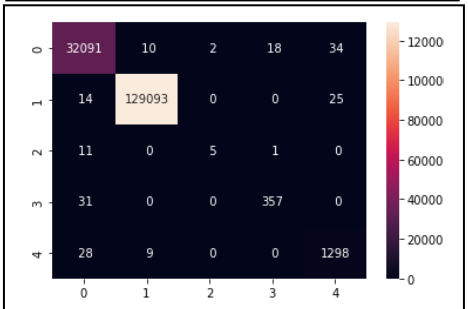


Figure 34. ExtraTreesClassifier

	ACC without feature Selection	F1-Score without feature Selection	ACC with feature Selection	F1-Score with feature Selection
DT	**99.95%**	**92%**	**99.888%**	**89%**
RF	**99.973%**	**96%**	**99.909%**	**90%**
Adaboost	**41.80%**	**31%**	**95.852%**	**45%**
Bagging	**99.959%**	**95%**	**99.897%**	**90%**
ExtraTrees	**99.97%**	**94%**	**99.887%**	**86%**

Table 4. the ML models performance

8.2.3. The Evolution of training CTGAN

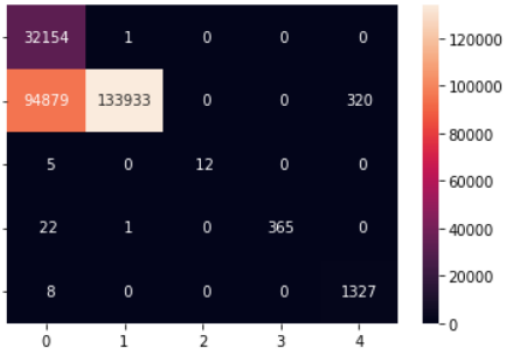


Figure 35. Before Adding the Discriminator to RandomForestClassifier

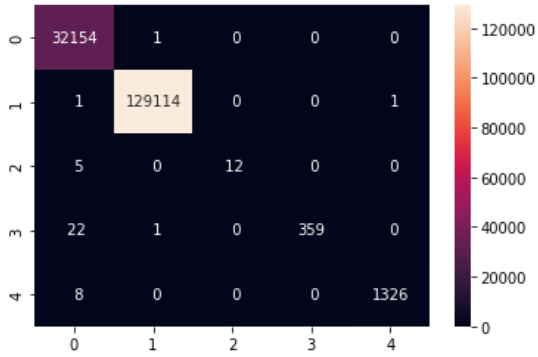


Figure 36. After Adding the Discriminator to RandomForestClassifier

9. Critical Discussion

The results of training different GANs network performance are high in term of accuracy and F1-score and that indicate our models can detect anomalies in the data perfectly, also when we try to check a loss using binary cross-entropy in the DCGAN model the loss equal zero and that was the ideal case as shown in **fig.36**

```
cross_entropy_binary = tf.keras.losses.BinaryCrossentropy()
loss = cross_entropy_binary([1., 1., 1., 1.], [1., 1., 1., 1.])
print('Loss: ', loss.numpy())
```

```
Loss:  -0.0
```

Figure 36. loss function

10. Results

Relevance

GANs network can be counted as a two sided weapon, since they have the ability to make the model learn new pattern from the data or we can make it produce fake data that can harm our systems, the things we have learn from this project were how we can produce different models using same data, also as we know GANs was widely used in computer vision but our case was tabular and we succeeded in converting the GAN model to be able to deal with our case.

11. Conclusion

In this work, the authors have visualized and analyzed our dataset to get more insights about the problem, after that we have implemented different binary and multiple classifiers to see which way is better to approach this problem, the binary classifiers focus only if each data record is an attack or not, while the multiple classifiers classify the normal attacks and the different types of the attacks. after that we built different GAN architectures to generate fake normal data to be able to test our both binary and multiple classifiers champion models, and we have found that the multiple classification RandomForestClassifier model was successfully able to predict the fake normal data as different types of attacks data (and mostly the attack of class 1) that we generated using our Conditional GAN architecture that we have implemented from scratch, and also we have added some noise data (some data from each attacks classes) and also we have added a normal data to be able to test the performance of our champion multiple classifiers on each different classes of the five classes. And fortunately, it performed well with the accuracy of 100% and an f1 score of 96%.

12. Future Work

1. We will apply our mechanism but on the whole data labels 23 instead of using only 5 labels.
2. We will try different data engineering to see the effect of it on our results.
3. We will implement it on different dataset.
4. We will convert our tabular data into images and apply the DCGAN and Cycle-GAN to achieve the power of that models on the images.
5. More sophisticated GAN architecture and/or other GAN types.

13. REFERENCES

[1] Chen, H., & Jiang, L. (2019). Efficient GAN-based method for cyber-intrusion detection. https://arxiv.org/pdf/1904.02426.pdf

[2] Yang, J., Li, T., Liang, G., He, W., & Zhao, Y. (2019). A Simple Recurrent Unit Model Based Intrusion Detection System With DCGAN. IEEE Access, 7, 83286–83296. https://doi.org/10.1109/access.2019.2922692

[3] Salem, M., & Taheri, S. (2018, November 19). Anomaly Generation Using Generative Adversarial Networks in Host-Based Intrusion Detection. Dr Shayan Sean Taheri. https://www.researchgate.net/publication/329607892_Anomaly_Generation_using_Generative_Adversarial_Networks_in_Host-Based_Intrusion_Detection

[4] Kaja, N., Shaout, A., & Ma, D. (2019). An intelligent intrusion detection system. Applied Intelligence, 49(9), 3235–3247. https://doi.org/10.1007/s10489-019-01436-1

[5] Meena, G., & Choudhary, R. R. (2017, July 1). A review paper on IDS classification using KDD 99 and NSL KDD dataset in WEKA. IEEE Xplore. https://doi.org/10.1109/COMPTELIX.2017.8004032