IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 12, Issue 7, July 2024

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.379

9940 572 462 | 6381 907 438 | ijircce@gmail.com | www.ijircce.com

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 7, July 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1207053 |

Improved Bidirectional GAN- based Approach for Network Intrusion Detection using One-Class C

Nydile D[1], C S Swetha[2]

MCA Student, Department of Computer Application, Bangalore Institute of Technology, Bangalore, India[1]

Assistant Professor, Department of Computer Application, Bangalore Institute of Technology, Bangalore, India[2]

ABSTRACT: Generative Adversarial Networks (GANs) are typically utilized to produce false images that look real. This process demands that the discriminator and the images generator be closely synchronized. However, Regarding using GANs for network intrusion detection, this strict synchronization might not be the best approach. This is because information used in network intrusion detection is simpler in structure with fewer dimensions, specific values, and smaller sizes compared to image data used in GANs.

To solve this problem, the researchers propose a new type of GAN, called Bidirectional GAN (Bi-GAN), which is better suited for the detection of network intrusions. According to their method, the discriminator and the generator are taught independently until they meet a specific condition related to a cross-entropy loss function. This strategy improves the performance of both components, even when dealing with imbalanced data.

Additionally, their model introduces a one-class classifier with the encoder that has been trained and discriminator. This classifier identifies unusual network activity through binary classification, avoiding the need for complex calculations. The outcomes of the experiment demonstrate that this method is very effective for detecting network intrusions, outperforming other similar techniques on the NSL-KDD and CIC-DDoS2019 datasets.

I. INTRODUCTION

Network intrusion detection examines incoming and outgoing information for indications of malicious behavior in order to detect unwanted access to a network. It's crucial for blocking or stopping cyberattacks**.**

Traditional ML techniques perform best when the data is labeled correctly. However, as data grows, labeling becomes expensive or impractical. Unsupervised methods don't need labeled data and can recognize patterns from normal traffic to detect anomalies. However, they struggle when there's an imbalance in the data.

Generative models like Autoencoders and GANs create synthetic data to improve detection accuracy. Autoencoders compress data into a smaller form and then recreate it, identifying anomalies by comparing the original and recreated data. GANs use two networks, a generator and a discriminator, to produce accurate data. The discriminator attempts to distinguish between actual and phony data, while the generator attempts to produce bogus data that seems real. fake data. This adversarial training helps both networks improve.

In order to identify network intrusions, existing GAN methods have two issues:
The generator and discriminator are trained together, but for network data, this isn't necessary due to simpler and reduced input features compared to images.

Current techniques make use of complex calculations to identify anomalies, which isn't needed for simpler network data.

The researchers suggest a Bidirection GAN (Bi-GAN) for network intrusion detection in order to address these problems. Through more rigorous training of the generator and encoder than the discriminator, their system improves performance without needless training. Additionally, they detect anomalies using a less complex loss function.

Their contributions include:
Relaxing the synchronous training requirement for the generator and discriminator, allowing more iterations for the generator, improving overall performance.Using the trained encoder-discriminator to create a one-class classifier that can identify anomalies without complex calculations.Achieving over 99% F1-score on the CIC-DDoS2019 dataset and over 92% F1-score on the NSL-KDD dataset, demonstrating high effectiveness in detecting network anomalies.

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 7, July 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1207053 |

II. BACKGROUND

- Generic GAN

In a general GAN approach, two neural networks, videlicet the creator and the discriminator, independently, contest with each other in a game approach generally in the form of a zero- sum game where one agent's gain is another agent's loss. The generative network(i.e., the creator) generates new data samples from a low dimensional distribution while the discriminational network(i.e., the discriminator) evaluates them. In another word, the creator learns to collude from arbitrary noise to a data distribution of the real data while the discriminator distinguishes the new datasets generated by the creator(i.e., regarded as fake data) from the true data distribution. Figure 1 illustrates the armature of the GAN.

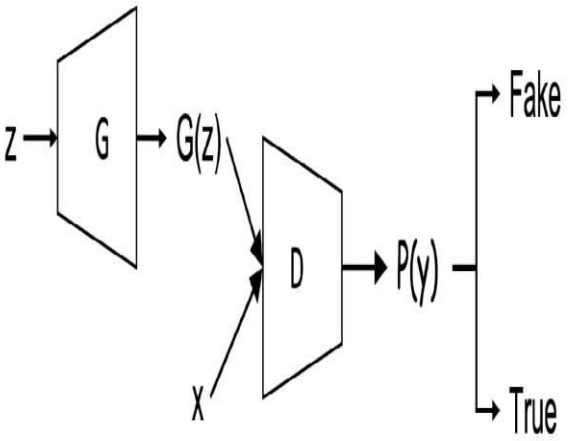


Figure 1. Structure of GAN. The creator G collude the input z(i.e., arbitrary noise) in idle space to produce a high dimensional G(z)(i.e., fake samples). The discriminator D is anticipated to separate x(i.e., real samples) from G(z). Algorithm 1 demonstrates the procedure involved in the training phase of a general GAN model. The creator takes a batch of vectors z(e.g., aimlessly drawn from a Gaussian distribution) and maps to G(z) which has the same confines as the real samples x. The discriminator receives two sources of input(i.e., fake samples and real samples) and tries to distinguish them. The loss between the observation and the vaticination at the discriminator is calculated and latterly used to modernize both the creator and the discriminator until the training iscomplete.It must be noted that there's no independent loss function for the induce the standard GAN as it's streamlined laterally with the objective function linked to the discriminator. Equation(1) depicts the ideal of V(G, D) for measuring the residual and optimizing both the creator and the discriminator.

$$\text{Min G max D } V(G, D) = E_{x\sim pX}[\log D(x)] + E_{z\sim pZ}[\log(1 - D(G(z)))]$$

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 7, July 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1207053 |

Algorithm 1: Training in Generic GAN

```
for number of training iterations do
    for k steps do
        Sample {z^(1), z^(2), ..., z^(r)} ~ p(z);
        e z ( ..
        Sample {x^(1), x^(2), ..., x^(r)} ∈ p(x);
        e x ( ..
        x̂ = G(z);

        Update D(x, x̂) by maximizing 1
        ate Equation (
    e
    n
    Sample {z^(1), z^(2), ..., z^(r)} ~ p(z);
    e z ( ..
    Update G(z) by minimizing without )
    ate Equation ( (updating D
e
n
```

III. BIDIRECTIONAL GAN

Bidirectional GAN or BiGAN is a variant of GAN by adding an encoder to the original GAN model. With the added encoder, the BiGAN is able to learn the inverse mapping from the real data to the idle space(19, 20) to more support the creator producing further semantically rich synthetic datasets. The encoder then plays an importable part for the BiGAN model by furnishing the learning the idle representation from the real data(19). Figure 2 illustrates the armature of BiGAN, and Algorithm 2 depicts the training involved in the BiGAN approach.

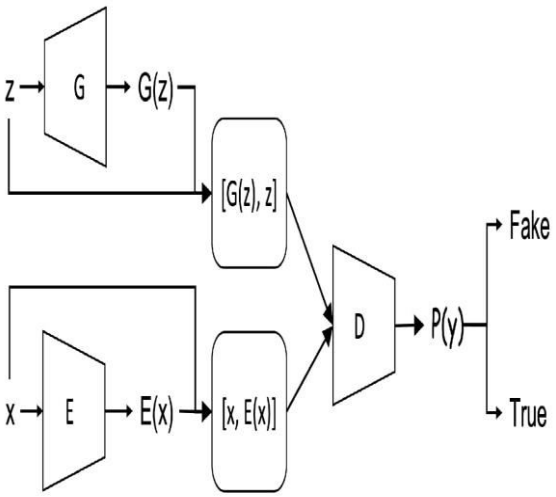


Figure 2. Structure of BiGAN. Note that(z and E(x)) and(G(z) and x) have the same confines. The concatenated dyads(G(z), z) and(x, E(x)) are the two input sources of the discriminator D.The Generator G and the encoder E are optimized with the loss generated by the discriminatorD.Like the standard GAN, the training is comprised of two way. The first step involves training the discriminator(D) to maximize the objective function described in Equation(2)

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 7, July 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1207053 |

without streamlining the creator(G) or encoder(E). The alternate step involves training both the creator and encoder to minimize the same objective function linked to the discriminator(D). Min G,E max D L(D, E, G) = Ex∼pX [Ez∼pE(·|x)[log D(x, z)]]+ Ez∼pZ [Ex∼pG(·|z)[log(1 − D(x, z))]]

Algorithm 2: Training in BiGAN

for *number of training iterations* do
  for *k steps* do
    Sample $z\,(z^1, z^2 \ldots . z^n) \sim p(Z)$;
    Sample $x\,(x^1, x^2 \ldots . x^n) \in p(X)$;
    $f(z) = G(z)$; /* $f(z).shape = x.shape$ */
    $\hat{f}(x) = E(x)$; /* $\hat{f}(x).shape = z.shape$ */
    Concatenate $([f(z), z])$;
    Concatenate $([x, \hat{f}(x)])$;
    Update $D([f(z), z])$ and $D([x, \hat{f}(x)]$ by maximizing Equation (2);
  end
  Sample $z\,(z^1, z^2 \ldots . z^n) \sim p(Z)$;
  Sample $x\,(x^1, x^2 \ldots . x^n) \in p(X)$;
  $f(z) = G(z)$;
  $\hat{f}(x) = E(x)$;
  Concatenate $([f(z), z])$;
  Concatenate $([x, \hat{f}(x)])$;
  Update $G(z)$ and $E(x)$ simultaneously by minimizing Equation (2) (without updating D).
end

Training Loss Function

Algorithm 3: Training Phase of our proposed method

for *number of training iterations* do
  D.trainable = True;
  Sample $z\,(z^1, z^2, \ldots, z^n) \sim p(Z)$;
  Sample $x\,(x^1, x^2, \ldots, x^n) \in p(X)$;
  $f(z) = G(z)$; /* $f(z).shape = x.shape$ */
  $\hat{f}(x) = E(x)$; /* $\hat{f}(x).shape = z.shape$ */
  Concatenate $([f(z), z])$;
  Concatenate $([x, \hat{f}(x)])$;
  Update $D([f(z), z])$ and $D([x, \hat{f}(x)]$ by maximizing Equation (2);
  D.trainable = False;
  for *k steps* do
    Sample $z\,(z^1, z^2 \ldots . z^n) \sim p(Z)$;
    Sample $x\,(x^1, x^2 \ldots . x^n) \in p(X)$;
    $f(z) = G(z)$;
    $\hat{f}(x) = E(x)$;
    Concatenate $([f(z), z])$;
    $O(\hat{y}|y) \leftarrow D([f(z), z])$;
    Update G by minimizing $-\log(D(f(z), z))$;
    Concatenate $([x, \hat{f}(x)])$;
    $O(\hat{y}|y) \leftarrow D([x, \hat{f}(x)])$;
    Update E by minimizing $-\log(D(x, f(x)))$;
  end
end

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 7, July 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1207053 |

IV. TRAINING PHASE

In numerous cases, Kullback – Leibler(KL) divergence or Jensen – Shannon(JS) divergence are used to measure the distance between the common distribution of P(x, E(x)) and P(G(z)z) in numerous image- rested BiGAN variants which produces the optimum, E = G −1 and the divergence of 0. This doesn't work for textbook- rested BiGAN as the divergence of the two common distributions of P(x, E(x)) and P(G(z), z) can not be reckoned directly but can only be approached laterally using the discriminator. With this understanding, we use thecross- entropy to compare the divergence. Equation(3) depicts thecross- entropy of two distribution P(x) and Q(x) and where P(x) is the factual target(0 or 1) and the Q(x) is the common distribution of P(x, E(x)) or P(G(z), z).

$$H(P, Q) = Ex \sim P(\log Q(x)) = -\sum P(x) \log(Q(x))$$

where P(x) is the factual target(0 or 1) and the Q(x) is the prognosticate. Now we can unify the updating of the encoder, creator, and discriminator to minimize the result of this loss.
Computers 2022, 11, 85 8 of 18
In the first stage of training, the discriminator is streamlined in replication if the input comes from the encoder, the alternate part of the objective function(becomes 0. Maximizing the first part of the objective function equals to minimize thecross- entropy values between P(x = 1) and Q(x) as seen in the following Equation:

$$H(1, Q) = -\log D(x, E(x))$$

still, the first part of the objective function(2) becomes 0, If the input comes from the creator. The alternate part of the objective function is original to minimizing thecross- entropy values between P(x = 0) and Q(z), as seen in the following Equation

$$H(0, Q) = -\log D(G(z), z) \quad (5)$$

When training the creator and the encoder, the parameters of the discriminator is fixed. In fact, the two modules are trained independently the encoder is trained in the encoder- discriminator common structure while the creator is trained in the creator- discriminator common network. Since the markers of input are shifted, the target marker will set to be 0 when the input is a real sample x ∼ p(x). streamlining the encoder is to minimize thecross- entropy between P(x = 0) and Q(x)

$$H(0, Q) = -\log D(x, E(x)) \quad (6)$$

On the other hand, streamlining the creator is to minimize thecross- entropy between
P(x = 1) and Q(x) $H(1, Q) = -\log D(G(z), z)$

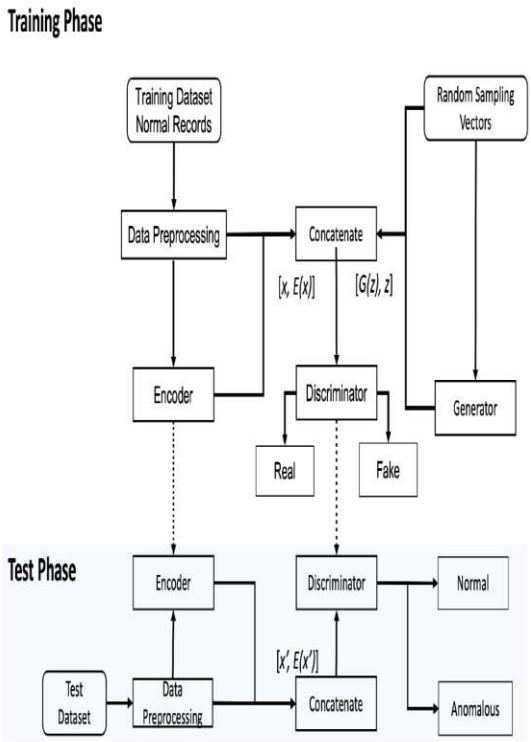


Figure 3: Flowchart of our proposed approach.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 7, July 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1207053 |

Training Phase:

Normal Data Processing:

We take a normal traffic sample, call it x.
This sample goes through the encoder, which transforms it into a different form called E(x).
We combine the original sample x with its transformed version E(x) to create a new input [x, E(x)].
The discriminator receives this combined input, which is given the label 0 to indicate that it is real data.
Generating Fake Data:

The generator creates fake samples. It starts with a small random vector z.
It turns this vector into a fake sample, called G(z).
We combine this fake sample G(z) with its original small vector z to create another input [G(z), z].
This combined input is labeled as fake data (with the label 1) and fed to the discriminator.
Discriminator's Role:

The discriminator's job is to guess whether an input is real or fake by giving a probability score O(yˆ|y).
If the score is close to 0, it means the discriminator thinks the input is real.
If the score is closer to 1, it means the discriminator thinks the input is fake.
The discriminator learns from these guesses and improves its ability to progressively separate authentic from phony samples by utilizing a loss function to adapt.

V. DATA AND PREPARATION SETS

The NSL-KDD dataset and the CIC-DDoS2019 dataset were the two distinct datasets that we used for our analyses. These datasets are commonly used because there aren't many public datasets available for developing new network intrusion detection models, despite the fact that they don't exactly mimic real-world networks. These datasets are useful reference points for contrasting various approaches.

We employed two components for the NSL-KDD dataset: KDDTrain+ for model training and KDDTest+ for model testing. There are 125,973 records in the KDDTrain+ dataset overall, comprising 67,343 normal samples and 58,630 aberrant samples. There are 22,544 records in the KDDTest+ dataset, consisting of 12,833 aberrant samples and 9,711 normal samples. Table provides a summary of the NSL-KDD dataset's details.

Table. Records of two NSL-KDD datasets: KDDTrain+ and KDDTest+.

NSL-KDD	Total	Normal	Others
KDDTrain+	125,973	67,343	58,630
KDDTest+	22,544	9711	12,833

Each traffic record in the NSL-KDD dataset contains a total of 41 features, including 38 numeric (e.g., "int64" or "float64") and 3 symbolic values (e.g., "object").

Thirteen distinct kinds of DDoS assaults are present in the CIC-DDoS2019 dataset. With almost 50 million attack samples and relatively few BENIGN samples, the dataset is extremely skewed. We created a training set for our experiment by extracting all 56,425 distinct benign samples. To assess our method, we created a test set by randomly selecting 5% of the assault samples and a subset of the benign samples. The number of sample sizes utilized for testing and training is displayed in Table 3.

Information Preprocessing

For the dataset NSL-KDD, we process the data as follows:
One-Hot Encoding: We convert 84 distinct traits with 3 symbolic features using one-hot encoding. This increases the total number of features to 122.
Outlier Removal: After encoding, we remove outliers, specifically the top 5% of values in any feature. This ensures all features are treated equally and reduces bias.
Normalization: We use Min-Max scaling to normalize the data, scaling all values to be between 0 and 1.

- For producing the model's input data:

Vectors entered: We select vectors at random from a normal distribution that has the same dimensions as the latent space, for example, ten.Generator Output: The output size of the generator is adjusted to correspond with the total number of features, for example, 122 features.The encoder and discriminator work together to translate the data from

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 7, July 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1207053 |

122 characteristics to the latent space, which consists of, for example, 10 dimensions. The discriminator produces a single value (for binary classification) from the generator's combined input and output, which in this case totals 132 characteristics.

The preprocessing procedures for the CIC-DDOS2019 dataset are the same:

- Feature Reduction: We remove uninformative features and all-zero features. We also encode the "protocol" feature, which has 3 values, into three separate features.
- Outlier Removal: We remove the top 5% of values for each feature and select 29,731 benign samples for training.
- Normalization: Finally, The data is normalized to fall between 0 and 1.

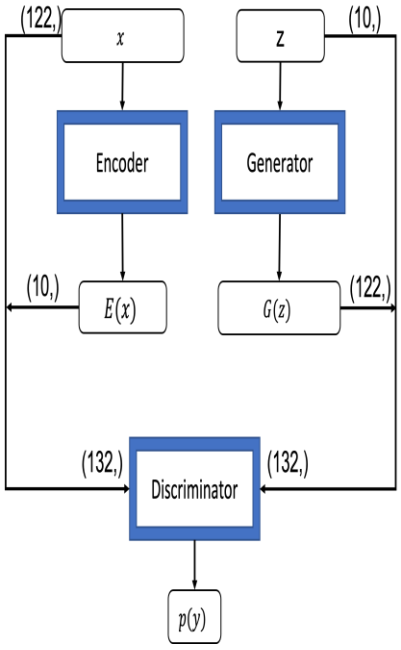


Figure 4: Data flow of BiGAN. (10), 122 are the output dimensions of the encoder; (10), 122 are the output dimensions of the generator; Discriminator acts as a binary classifier by concatenating the input and output of an encoder or generator to generate the input.

Performance Metrics

Accuracy: Measures how often the model predicts correctly overall. It's calculated as the proportion of samples that were accurately predicted (both anomalies and normal instances) to the total number of samples.

Accuracy=(TP+TN)/(TP+TN+FP+FN)

TP: True Positives (correctly predicted anomalies)
TN: True Negatives (correctly predicted normal instances)
FP: False Positives (normal instances misclassified as anomalies)
FN: False Negatives (anomalies misclassified as normal)

Precision: Measures the precision of the positive predictions. It's the proportion of accurately forecast anomalies to all predicted anomalies.

Precision = TP/TP+FP

Recall (True Positive Rate): Measures the proportion of accurately anticipated anomalies to all anomalies of actual anomalies.

Recall(TPR) = TP/TP+FN

High recall indicates that the model is good at detecting anomalies.

F1 Score: Combines accuracy and memory into a single metric. It's the harmonic mean of precision and recall, providing a balance between them.

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 7, July 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1207053 |

Area Under the ROC Curve (AUC-ROC): Indicates the model's performance to distinguish between classes (anomaly vs. normal) at various thresholds. A higher AUC-ROC value indicates better performance.

- ROC curve plots True Positive Rate (Recall) against Fals Positive Rate (the ratio of false positives to all actual negatives) across different thresholds.

AUC-ROC calculates the region beneath this curve, the a value closer to 1 indicates better model performance.

VI. RESULTS

We present our analysis of the observations we made throughout our trials..

PCA Definition and Training Loss:

A model's ability to learn during training is shown by its training loss. You are utilizing a Generative Adversarial Network in your research.

(GAN), where the both a generator and a discriminant have opposing objectives:

Generator (Gloss): Aims to generate data that looks like the real dataset.

Discriminator (Dloss): Tries to distinguish between precise and genuine data.

Encoder (Eloss): Inverse maps generated data back to latent space.

Trend Over Iterations: Initially, the losses for the encoder and generator are unstable but stabilize after around 200 iterations. The discriminator's loss stabilizes much earlier, around 50 iterations. This stability shows that the model is converging to a consistent state where both generator and discriminator are improving in their tasks.

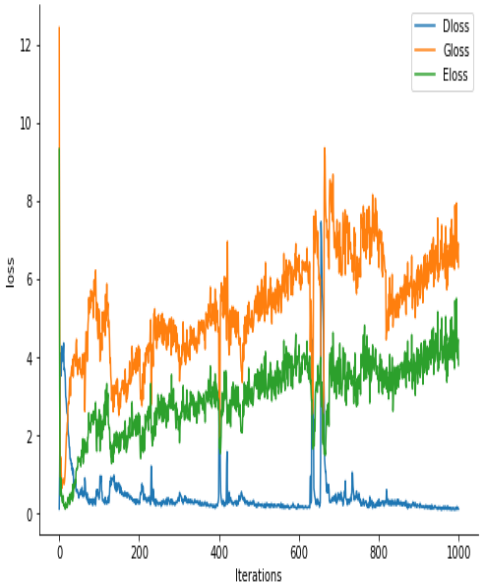


Figure 5. Iterations versus training losses. The training loss trend of the discriminator, generator, and encoder are represented by the symbols Eloss, Gloss, and Dloss, respectively.

PCA Visualization

Purpose: Principal Component Analysis (PCA) is used to visualize the distribution of data in a lower-dimensional space, showing patterns and clusters within the data.

Interpreting Figures:

KDDTrain+ Dataset: Shows two clear clusters—one for normal data and another for abnormal data. Normal data points cluster tightly together, while abnormal data points are more spread out.

KDDTest+ Dataset: Exhibits less distinct clusters, with overlapping points between normal and abnormal data. Normal data features are tightly grouped, but abnormal data features are more widely spread.

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 7, July 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1207053 |

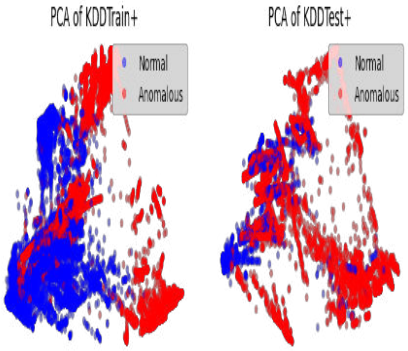


Figure 6. The PCA visualization of data distribution in (**a**) KDDTrain+ and (**b**) KDDTest+ dataset.

- Encoder and Generator Results:
- NSL-KDD Dataset: After training, real samples (from the dataset) cluster tightly, while generated samples (from the generator) surround them but are more scattered. This suggests the generator can mimic normal data distribution but generates anomalies that are more varied.
- CIC-DDoS2019 Dataset: Similar pattern observed where generated samples (fake) are spread out more than real samples (real), indicating challenges in accurately generating data that matches the real dataset's distribution.

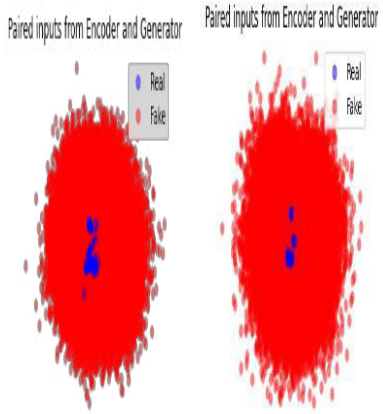


Figure 7 shows the concatenated outputs of the generator and encoder after PCA visualization on (a) NSL-KDD and (b) CIC-DDoS2019 training.

Testing

Performance Metrics: The model's performance is evaluated using four metrics**:**
Precision: Measures the accuracy of positive predictions.
Recall: Measures how well the model detects positive instances.
F1-Score: Harmonic mean of precision and recall, providing a balanced measure.
Accuracy: Overall accuracy of the model in dividing cases into normal and abnormal categories.

Results:
NSL-KDD Dataset: Achieves a F1-score of over 92%, indicating strong performance in both precision and recall.
CIC-DDoS2019 Dataset: Achieves a F1-score of over 99%, showing exceptionally high accuracy in detecting anomalies.

Training Runtime
Training Iterations: The model undergoes 1000 iterations during training.
Generator vs. Discriminator Iterations:
Compared to the discriminator, the generator iterates five times more. This disparity results from the roles of generator (creating synthetic data) and discriminator (distinguishing real from fake) have different training requirements and don't need to synchronize their iterations.

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | A Monthly Peer Reviewed & Referred Journal |

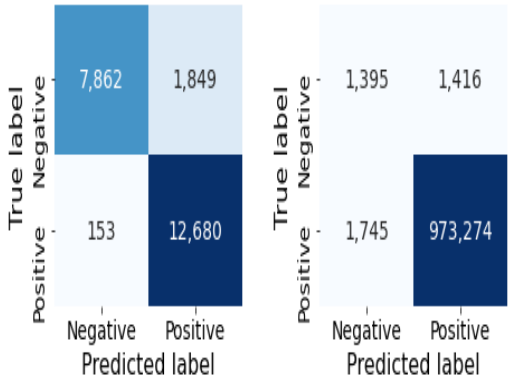
|| Volume 12, Issue 7, July 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1207053 |

Training Runtime:
Average Time: Represents the time taken for 1000 iterations of training, measured in milliseconds. This metric indicates how efficiently the model processes and learns from the data over multiple training cycles.

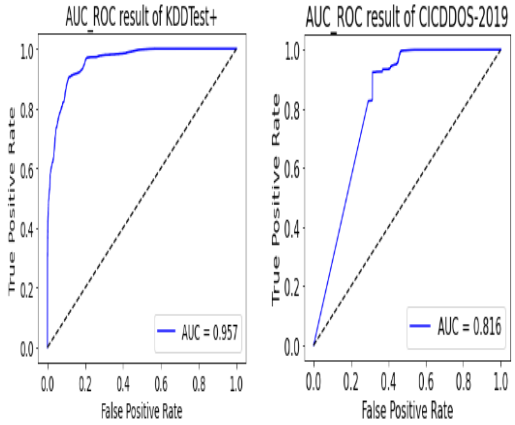
Figure 8 displays the experimental results based on the confusion matrix to provide a more thorough examination of the performance. Approximately 2,000 records (or less than 9% of the total records) in the NSL-KDD dataset displayed in (a) were incorrectly classified as either FP (1849) or FN (153), out of the 22,544 records utilized for the testing. The remaining 20,542 records were correctly classified in accordance with their label. In the CIC-DDoS2019 dataset displayed in (b), out of 989,780 testing samples, 974,669 records (more than 99% of the total records) were correctly identified, while 3161 samples were incorrectly classified.

The AUC_ROC curve, Figure 9 shows, makes it evident how true positive rate and false positive rate are traded off when evaluating the performance of our suggested model from a different perspective. Our suggested model is very successful in correctly classifying network intrusions, as evidenced by the AUC score of 0.953 for the NSL-KDD dataset. Conversely, the model's AUC score on the CICDDoS2019 dataset is a comparatively low 0.816. This results from the low number of samples being trained, which raises the false positive rate in the categorization of BENIGN samples.
.



a) b)

Figure 8. Confusion matrix result of (**a**) NSL-KDD and (**b**) CIC-DDoS2019



(a) (b)

Figure 9. AUC_ROC curve of our proposed model on (**a**) NSL-KDD and (**b**) CIC-DDoS2019.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 7, July 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1207053 |

VII. CONCLUSIONS

- Proposed Bidirectional GAN Model

Purpose: The study introduces a new Bidirectional Generative Adversarial Network (GAN) designed specifically for detecting network intrusion attacks. Unlike typical GANs used in image processing, our model allows the generator and discriminator to train independently without strict synchronization.

- Advantages:

Reduced Training Overheads: By decoupling the training of generator and discriminator, our model achieves more efficient training without compromising on performance.
One-Class Classifier: Includes a simpler one-class binary classifier based on the trained encoder and discriminator, which helps in identifying anomalous network traffic directly without complex threshold calculations.

- Effectiveness:

Performance: Evaluated on two datasets (NSL-KDD and CIC-DDoS2019), High accuracy is achieved by the model using F1-scores exceeding 92% and 99%, respectively. This demonstrates its effectiveness in generating synthetic network traffic data and detecting anomalies.

- Future Directions:

Data Augmentation: To address issues like high false positives in existing network intrusion datasets, future work involves expanding the model's ability to produce additional artificial data samples that closely resemble actual data.
Techniques:

Similarity Functions: Consider using methods like Pearson Correlation or flocking methods to evaluate how closely synthetic data matches real data across various aspects of their distribution.

REFERENCES

1.Jang-Jaccard, J.; Nepal, S. A survey of emerging threats in cybersecurity. J. Comput. Syst. Sci. 2014, 80, 973–993. [CrossRef]
2.Ahmad, Z.; Khan, A.S.; Shiang, C.W.; Abdullah, J.; Ahmad, F. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. Trans. Emerg. Telecommun. Technol. 2021, 32, e4150. [CrossRef]
3.Zhu, J.; Jang-Jaccard, J.; Liu, T.; Zhou, J. Joint Spectral Clustering based on Optimal Graph and Feature Selection. Neural Process. Lett. 2021, 53, 257–273. [CrossRef]
4.Kingma, D.P.; Welling, M. Auto-encoding variational bayes. arXiv 2013, arXiv:1312.6114.
5.Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. Adv. Neural Inf. Process. Syst. 2014, 27, 1–9.
6.Schlegl, T.; Seeböck, P.; Waldstein, S.M.; Schmidt-Erfurth, U.; Langs, G. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In Information Processing in Medical Imaging; Niethammer, M., Styner, M., Aylward, S., Zhu, H., Oguz, I., Yap, P.T., Shen, D., Eds.; Springer International Publishing: Cham, Switzerland, 2017; pp. 146–157.
7.Schlegl, T.; Seeböck, P.; Waldstein, S.M.; Langs, G.; Schmidt-Erfurth, U. f-AnoGAN: Fast unsupervised anomaly detection with generative adversarial networks. Med. Image Anal. 2019, 54, 30–44. [CrossRef] [PubMed]
8.Akcay, S.; Atapour-Abarghouei, A.; Breckon, T.P. GANomaly: Semi-supervised anomaly detection via adversarial training. In Computer Vision—ACCV 2018; Jawahar, C.V., Li, H., Mori, G., Schindler, K., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 622–637.
9.Chen, H.; Jiang, L. Efficient GAN-based method for cyber-intrusion detection. arXiv 2019, arXiv:1904.02426.
10.Kaplan, M.O.; Alptekin, S.E. An improved BiGAN based approach for anomaly detection. Procedia Comput. Sci. 2020, 176, 185–194. [CrossRef]
11.Javaid, A.; Niyaz, Q.; Sun, W.; Alam, M. A deep learning approach for network intrusion detection system. Eai Endorsed Trans. Secur. Saf. 2016, 3, e2.
12.An, J.; Cho, S. Variational autoencoder based anomaly detection using reconstruction probability. Spec. Lect. IE 2015, 2, 1–18.
13.Chang, Y.; Tu, Z.; Xie, W.; Yuan, J. Clustering driven deep autoencoder for video anomaly detection. In Computer Vision—ECCV 2020; Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 329–345.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 7, July 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1207053 |

14.Xu, W.; Jang-Jaccard, J.; Singh, A.; Wei, Y.; Sabrina, F. Improving Performance of Autoencoder-based Network Anomaly Detection on NSL-KDD dataset. IEEE Access 2021, 9, 140136–140146. [CrossRef]
15.Sadaf, K.; Sultana, J. Intrusion Detection Based on Autoencoder and Isolation Forest in Fog Computing. IEEE Access 2020, 8, 167059–167068. [CrossRef]
16.Aygun, R.C.; Yavuz, A.G. Network Anomaly Detection with Stochastically Improved Autoencoder Based Models. In Proceedings of the 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud), New York, NY, USA, 26–28 June 2017; pp. 193–198. [CrossRef]
17.Zenati, H.; Foo, C.S.; Lecouat, B.; Manek, G.; Chandrasekhar, V.R. Efficient gan-based anomaly detection. arXiv 2018, arXiv:1802.06222.
18.Mohammadi, B.; Sabokrou, M. End-to-End Adversarial Learning for Intrusion Detection in Computer Networks. In Proceedings of the 2019 IEEE 44th Conference on Local Computer Networks (LCN), Osnabrueck, Germany, 14–17 October 2019; pp. 270–273.
[CrossRef]
19.Dumoulin, V.; Belghazi, I.; Poole, B.; Mastropietro, O.; Lamb, A.; Arjovsky, M.; Courville, A. Adversarially learned inference.
arXiv 2016, arXiv:1606.00704.
20.Donahue, J.; Krähenbühl, P.; Darrell, T. Adversarial feature learning. arXiv 2016, arXiv:1605.09782.

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi crossref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details