International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Improved DDoS Detection Models using Autoencoders and Generative Adversarial Networks for Internet of Things-based Networks

Marram Amitha*[1], Dr.Muktevi Srivenkatesh[2]

Submitted: 29/01/2024 **Revised**: 07/03/2024 **Accepted**: 15/03/2024

Abstract: Denial-of-service (DDoS) attacks represent the primary threat to the continuous and efficient performance of the Internet. It may have been an element in server delays disconnections, host issues, lost revenue and production, and website vulnerability. Standard machine learning algorithms suffer from increased false-positive rates and reduced rate of detection when new threats develop. Therefore, the DDoS detection devices must include high-performance machine learning classifiers with low false-positive rates and high prediction accuracy. This research paper presents an in-depth study into the scalability and resource efficiency of Deep Learning-based DDoS detection models, specifically convolutional neural networks (CNNs), recurrent neural networks (RNNs), and auto encoders. With a rapid growth of internet of things (IoT) devices, there has been an increase in both the amount and intensity of network attacks. Attacks which result in a denial of service (DoS) or distributed denial of service (DDoS) are considered to be the most prevalent in IoT networks in recent years. Since a majority of current security solutions—firewalls, intrusion detection systems, etc.—filter all valid and malicious data through static, predefined standards, they have been unable to recognize advanced DoS and DDoS attacks. However, when coupled with techniques based on artificial intelligence (AI), these solutions can become reliable as well as effective. We also investigate the effectiveness of transfer learning and model stacking techniques to improve detection performance. Various DDoS scenarios are simulated in a cloud computing environment to assess real-time performance metrics such as latency, throughput, and resource utilization. Experimental results demonstrate that while deep learning models provide high accuracy and F1 scores, the application of transfer learning and model stacking further enhances these metrics. Importantly, we also find that certain architectures demonstrate superior scalability and consume fewer resources, making them better suited for real-world cloud computing applications. The findings from this research contribute valuable insights for the deployment of scalable and resource-efficient DDoS detection systems in cloud computing.

Keywords: *Distributed denial of service, Internet of Things, intrusion detection, Deep Sparse Auto encoder*

1. Introduction

The entire world has evolved into a globally connected place in the modern period. The evolution is caused by the reality that knowledge can be transmitted more easily, communication has become increasingly flexible, and many aspects of life have been computerized. Information thus becomes widely available wherever we look. Information systems, independent of an organization's size or area of activity, have become essential in this environment. Information systems have been attacked by malware infections more frequently recently, while advanced attacks have proliferated, malware attacks have also seen significant changes in category. These types of attacks can be detected and destroyed simply network-based intrusion detection systems just simply searching the network against threats. Such circumstances have resulted in a development of the area of cyber security research and a consequent rise in the complexity of cyber security.[1]. To provide an amount of safety that can recognize and avoid these risks, multiple procedures and advances in technology are developed. To attempt to decrease service quality, a DDoS attacker uses a variety of available network devices to send huge numbers

of produced packets at the target server inside the same network using different source IP addresses. The attacker attempts to prevent those with permission from using the services offered by the victim server by flooding the victim's device with these packets. A system for early detection and mitigation of these attacks is required because of this probability. Effective systems for intrusion detection (IDS) would protect intellectual property, preserve network performance, enhance data security, and decrease potential responsibility for compromised notes or network data[1].

Around the world, it is predicted that crime will cost $6 trillion annually. However, based on Cisco data, there will be 15.4 million distributed Denial of service (DDoS) of Service) attacks by 2023. The procedure of identifying malware and cyberattacks has never been easy. Organizations in the public and private sectors have been dedicating an enormous amount of time and resources to reducing the impact of these threats.

A Distributed Denial of Service (DDoS) attack on Internet of Things (IoT) devices is a type of cyber attack where multiple compromised devices are used to flood a targeted system or network with a large volume of traffic. These

[1] *Department of Computer Science GITAM Deemed to be University, India ORCID ID : 0000-0002-0526-1707*
[2] *Department of Computer Science GITAM Deemed to be University, India ORCID ID : 0000-0001-9631-6402*
** Corresponding Author Email: 121962504006@gitam.in*

attacks can overwhelm the target's resources, making it unable to respond to legitimate requests from users or clients.

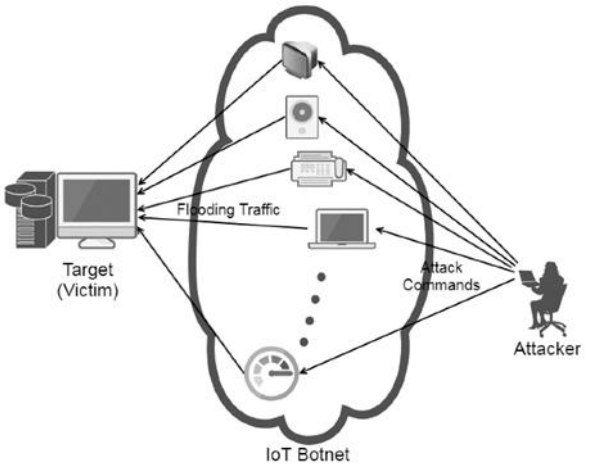

Fig.1. Ddos attacks in IoT network

There are several reasons why IoT devices are particularly vulnerable to DDoS attacks as shown in figure 1:

Large Attack Surface: IoT devices are often designed with minimal security measures to keep costs down, making them easier targets for attackers.

Default Credentials: Many IoT devices come with default usernames and passwords that users don't change, making it easy for attackers to gain access.

Limited Processing Power: IoT devices typically have limited processing power and memory, making them easy to overwhelm with a flood of traffic.

Always Connected: IoT devices are usually connected to the internet 24/7, providing a constant target for attackers.

Lack of Security Updates: Manufacturers may not provide regular security updates for IoT devices, leaving them vulnerable to known exploits.

To mitigate the risk of DDoS attacks on IoT devices, it's important to take the following steps:

Change Default Credentials: Always change default usernames and passwords on IoT devices to unique, strong credentials.

Update Firmware: Regularly update IoT device firmware to patch security vulnerabilities and protect against known exploits.

Network Segmentation: Segment IoT devices onto separate network segments to limit the impact of a compromised device on other parts of the network.

Traffic Monitoring: Implement traffic monitoring and anomaly detection systems to detect and mitigate abnormal traffic patterns that may indicate a DDoS attack.

Use Firewalls and Access Controls: Configure firewalls and access controls to restrict incoming and outgoing traffic to IoT devices, blocking unauthorized access and malicious traffic.

By implementing these measures, organizations and users can reduce the risk of DDoS attacks on IoT devices and enhance overall cybersecurity posture.

Detecting DDoS attacks on IoT devices

It involves monitoring network traffic and looking for patterns or anomalies that indicate a potential attack. Here are some methods to help find DDoS attacks on IoT devices:

Traffic Analysis: Use network monitoring tools to analyze incoming and outgoing traffic on your IoT devices. Look for sudden spikes in traffic volume or unusual patterns such as a high number of requests from a single source.

Anomaly Detection: Implement anomaly detection techniques to identify deviations from normal traffic behavior. This can involve setting thresholds for acceptable traffic levels and flagging any traffic that exceeds these thresholds as suspicious.

DDoS Protection Services: Consider using DDoS protection services or solutions specifically designed for IoT devices. These services can help detect and mitigate DDoS attacks in real-time, reducing the impact on your network and devices.

Behavioral Analysis: Monitor the behavior of IoT devices over time to establish a baseline of normal behavior. Any deviations from this baseline could indicate a DDoS attack or other security threat.

Logging and Alerting: Enable logging and configure alerting mechanisms to notify you of suspicious activity or potential DDoS attacks. Alerts can be based on criteria such as traffic volume, packet rates, or specific types of traffic patterns associated with DDoS attacks.

Flow Analysis: Utilize flow analysis tools to examine the flow of traffic between devices and identify any unusual or malicious traffic flows that may indicate a DDoS attack.

Collaborate with ISPs: Work closely with your Internet Service Provider (ISP) to monitor traffic entering your network. ISPs often have tools and capabilities to detect and mitigate DDoS attacks at the network level before they reach your IoT devices.

By combining these methods and staying vigilant, you can improve your ability to detect and respond to DDoS attacks targeting IoT devices, helping to protect your network and devices from disruption and damage.

Since recently identified malware lacks a signature in the anti-malware database, antivirus programs typically fail to detect it. To tackle those problems, our study proposed the use of deep learning for the identification of DDoS

cyberattacks from the beginning.

DDoS detection methods include methods to recognize and establish typical features among the enormous quantities of traffic that are used to flood a target's network with DDoS attacks [2]. We aim to create a hybrid model that uses deep neural networks and autoencoders to identify deceptive network traffic and recognize DDoS attacks in Internet of Things.

Predetermined malicious patterns cannot overfit due to the suggested model. This objective was initiated by the concept that implementing an autoencoder in addition to a deep neural network model will produce a more accurate classifier model for detecting malicious software network traffic that will work similarly to a conventional neural network model.

2. Related Work

Kawamura et al. explained the steps for event detection and measured the deviation of the system clock from the universal clock under two attack scenarios (with DDoS attacks and without DDoS attacks), to achieve this author developed a module focusing on the behavior of the system under DDoS attacks and utilized NTP as a synchronization service. Synchronization of the clock between client and server is of utmost importance, a miss by a few seconds will lead to great complications. When an IoT device receives a large amount of data, a delay in communication occurs due to the alteration in the system clock. So NTP is preferred over other protocols like Precision Time Protocol since NTP is right up to one nanosecond. He concluded that all the attacks are detected successfully by showing high values of recall and precision in realistic scenarios.

Sadique et al. discussed the importance of the deployment of IoT devices for effective communication. He also studied parameters for the security of IoT devices as heterogeneity, dynamic, and their number. Since we come across IoT devices in every sphere of life, so they are increasing at a faster pace[3]. The more is the number of IoT devices, the more are vulnerabilities.

Gurulakshmi demarcates between usual and weird traffic using the SVM algorithm and speculates weird traffic. The underlying approach works in phases, in the very first phase XOIC tool is used to produce traffic from numerous sources to a single destination. Since Traffic is of DDoS type, traffic (which is monitored) is nothing but a sequence of packets having source address, destination address, and packet count. Further to analyze the real instances of a packet and to save these instances for the future, Wireshark (an open source tool) has been used for obtaining packets and to limit the amount of space required for computation feature selection has been applied.

Khan et al. discusses IoT layers and associated protocols. Then security issues at these layers have been discussed and block chain is considered to be the best solution for resolving security issues keeping in mind the implications these layers face.

Ahmet et al. discussed about crunches in intelligent home systems and in smart cities applications. He concluded that despite of the fact that applications at home are very contented, they are almost used by half of the population and inferred that all the domains of smart cities deal online thereby creates glamorous city environment to be attacked by robber. It is obvious how life will standstill when admission of attacks is done to signaling system in subway and traffic, electricity cut off, stopping smart vehicles, preventing production lines from functioning. The damage that occurs to IoT devices depends on ransom money that attackers are willing to give back to these smart systems. A Distributed Denial of Service (DDoS) attack is a malicious method to interrupt regular access to a system, resource, or network by overloading the target or related infrastructure with traffic. Consequently, there has been an important decrease in the security of the network environment. Numerous works have been submitted and provided in previous research, but as attacker patterns and techniques are always changing, new ones are frequently proposed. A DDoS attack detection technique based on extreme learning machines (ELMs) has been proposed by Kushwah and Ali (2019). The NSL-KDD dataset was employed to estimate the solution. CNN and LSTM have been employed in a model-based deep learning architecture with Pekta and Acarman for evaluating the fundamental spatial-temporal features of network flows. The model generated accuracy, recall, precision, and F1-score of 99.09%, 99.08%, and 99.10%, accordingly, when assessed on the CIDOS 2019 dataset [3]. Researchers discussed the various kinds and purposes of cyberattacks and methods to avoid them and minimize their effect on society. Their inspection goals were to examine closely at common advancements in the field of cyber security as well as evaluate the challenges, drawbacks, and benefits of the proposed methods.

3. Dataset

Use The many new descendant attacks are thoroughly examined. The history of early cyber-security evaluates is examined together with standard security frameworks. Furthermore, included are recent developments, security issues, risks, and new trends in cyber security.

Concerning the datasets used in IDS, the most widely used ones are KDD99, NSL-KDD, and UNSW-NB15. Statistics from 2015 to 2018 showed that 38% of the total NSL-KDD dataset, 23% for KDD99, and 12% for UNSW-NB15 had been used. Previously, researchers additionally utilized the KDD99 and NSL-KDD datasets for developing machine learning and data dimensional reduction methods. The experiment in the current study carried out employing the

UNSW-NB15 dataset due of its important advantages compared to the KDD99 and NSL-KDD datasets. In addition, work performed by IDS during the years 2015 and 2018 shows the low classifying effectiveness of their methods employing the UNSW-NB15 dataset. Thus, one challenge is improving the method of classification to achieve better results on this dataset. Using the IXIA tool, the UNSW-NB15 dataset was developed for the purpose to extract offensive and current conduct from ACSC research completed in 2015. The data includes 2,540,044 instances, 49 features, and 9 various kinds of attacks in this dataset. Table 1 includes the features in order of appearance.

[4]. This dataset includes nine various kinds of attacks, such as analysis, backdoor, doS, exploits, fuzzers, generic, reconnaissance, shellcode, and worms. A component of the dataset has been separated into training and testing datasets, which have been frequently utilized in professional experiments. Table 2 presents detailed information about the datasets. In the experiments, we implement the training dataset (82.332 instances) and test dataset (175.341 instances) separately provided in UNSW-NB15 to avoid overfitting compared to performing k-fold cross-validation.

Combining the UNSW-NB15 dataset to the NSL-KDD dataset indicates several potential advantages. Firstly off everything, it involves equally broad attack operations and current, acceptable conduct. Second, there's a similarity among the training and test the data set's distributions of probability. Finally, it combines an array of features from the package's header and content in order to produce effective network packets as shown in fig 2.

id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate
1	0.121478	tcp	-	FIN	6	4	258	172	74.08749
2	0.649902	tcp	-	FIN	14	38	734	42014	78.473372
3	1.623129	tcp	-	FIN	8	16	364	13186	14.170161
4	1.681642	tcp	ftp	FIN	12	12	628	770	13.677108
5	0.449454	tcp	-	FIN	10	6	534	268	33.373826

Fig.2. Sample data in UNSW-NB15

Finally, the amount of data necessary for analyzing UNSW-NB15 applying modern methods of classification shows the dataset's intricate patterns. This means that the dataset could be used to evaluate conventional and novel classification methods.In above dataset screen first row represents dataset column names and remaining rows represents dataset values and in last column we can see class label as Normal or Attack Name. So by using above dataset we will train and test performance of GAN model.

he UNSW-NB15 dataset is a widely used dataset for network intrusion detection research, including DDoS attacks. It contains network traffic data captured from a realistic IoT environment, making it suitable for analyzing various types of cyber threats, including DDoS attacks. Features relevant to IoT DDoS attacks in datasets like UNSW-NB15 may include:

spkts: Source packets - The number of packets sent by the source (originating) device in a network communication.

dpkts: Destination packets - The number of packets received by the destination device in a network communication.

sbytes: Source bytes - The total number of bytes sent by the source device.

dbytes: Destination bytes - The total number of bytes received by the destination device.

rate: Rate - The data transfer rate or throughput, typically measured in bits per second (bps) or packets per second (pps).

sttl: Source time to live - The remaining time-to-live (TTL) value of packets sent from the source. TTL is a field in IP packets that limits the lifespan of a packet in the network.

dttl: Destination time to live - The remaining TTL value of packets received at the destination.

sload: Source load - The traffic load or utilization on the source side, often measured in bits per second (bps) or packets per second (pps).

dload: Destination load - The traffic load or utilization on the destination side.

sloss: Source loss - The number of packets lost or dropped on the source side during communication.

dloss: Destination loss - The number of packets lost or dropped on the destination side.

sinpkt: Source inter-arrival time - The time interval between successive packets sent by the source.

dinpkt: Destination inter-arrival time - The time interval between successive packets received by the destination.

sjit: Source jitter - The variation in inter-arrival times of packets sent by the source.

djit: Destination jitter - The variation in inter-arrival times of packets received by the destination.

swin: Source window size - The TCP window size advertised by the source device.

stcpb: Source TCP base sequence number - The initial sequence number used in TCP communication from the source.

dtcpb: Destination TCP base sequence number - The initial sequence number used in TCP communication at the destination.

dwin: Destination window size - The TCP window size advertised by the destination device.

tcprtt: TCP round-trip time - The time is taken for a TCP packet to travel from source to destination and back (RTT).

synack: SYN-ACK segment count - The number of SYN-ACK segments exchanged during TCP connection establishment.

ackdat: ACK data segment count - The number of data segments acknowledged by ACK during TCP communication.

smean: Source mean packet size - The average size of packets sent by the source device.

These parameters provide detailed information about network traffic characteristics, TCP/IP communication attributes, packet loss, inter-arrival times, and other metrics that are valuable for network analysis, intrusion detection, and cybersecurity research.

3.1 Packet Attributes

Features related to packet characteristics, such as packet size, packet duration, packet rate, and packet payload content. These attributes can provide insights into the nature of network traffic generated by IoT devices during DDoS attacks[5].

3.2 Protocol Information

Features describing the communication protocols used by IoT devices, including TCP, UDP, ICMP, and other application-layer protocols. Protocol-specific features such as protocol type, protocol flags, and protocol-specific payload characteristics can help identify anomalous or suspicious traffic patterns indicative of DDoS attacks.

3.3 Connection Attributes

Features related to network connections established by IoT devices, such as source and destination IP addresses, source and destination port numbers, connection duration, and connection state. Analyzing connection-level attributes can reveal patterns associated with DDoS attack traffic, such as high connection rates, unusual port usage, or connections to known malicious IP addresses[6].

3.4 Traffic Behavior

Features capturing the behavior of IoT devices and their interactions with network resources, including traffic volume, traffic patterns, traffic directionality, and traffic anomalies. Behavioral features can help differentiate between legitimate and malicious activities, enabling the detection of DDoS attacks based on deviations from normal traffic behavior.

3.5 Device Characteristics

Features related to the characteristics of IoT devices involved in DDoS attacks, such as device type, device firmware version, device manufacturer, and device-specific attributes. Understanding the properties of compromised IoT devices can aid in identifying vulnerabilities exploited by attackers and implementing targeted mitigation strategies.

3.6 Botnet Indicators

Features indicative of botnet activity, such as botnet command and control (C&C) communication patterns, botnet propagation mechanisms, and botnet membership information. Detecting botnet-related features can help identify compromised IoT devices participating in DDoS attacks and disrupt botnet operations through appropriate countermeasures [7].

3.7 Anomaly Scores

Features derived from anomaly detection algorithms or machine learning models trained to detect DDoS attacks based on statistical deviations from normal network behavior. Anomaly scores calculated for various network attributes can serve as input features for detecting and classifying DDoS attacks in real-time or offline analysis scenarios. By leveraging these features, security analysts and researchers can develop effective detection and mitigation strategies to combat IoT DDoS attacks and safeguard IoT ecosystems against malicious activities. Analyzing these parameters within the context of the UNSW-NB15 dataset can help researchers and practitioners understand the behavior of DDoS attacks in IoT environments, develop detection algorithms, and devise effective mitigation strategies to protect IoT infrastructures from such threats.

4. Data preprocessing

4.1 Upload UNSW-NB15 Dataset

Through the use of this module, we can submit a dataset to an application that will then read all of the dataset values and generate a graph displaying the number of attacks that were identified throughout the dataset.

4.2 Pre-process Dataset

By using this module, the dataset will be filtered by replacing any values that are missing. All non-numeric data is subsequently converted to numeric values using a label encoder, and the filtered and processed dataset values will be normalized.

4.3 Cleaning of Data

It involves recognizing inappropriate, insufficient, unnecessary, or absent data before changing, deleting, or replacing it. Most of the work is focused on ensuring that the input data is mistake-free and clean, as poor-quality data can lead to biased results, including low accuracy and high error rates in the model. The ones that follow are the pre-model training cleaning steps: We observed that while performing the EDA process, the input data included noise in a variety of missing or NaN values, infinity, and negative values.

In contrast to other research, we did not remove the noisy data; instead, we impute negative values with 0 and values that are infinite with NaN. Lastly, the Missforest (MF) approach is used to infer the missing or NaN data. As a non-parametric method, MF works well with both numeric and discrete data characteristics. It employs an iterative approach to impute by training a random forest model.

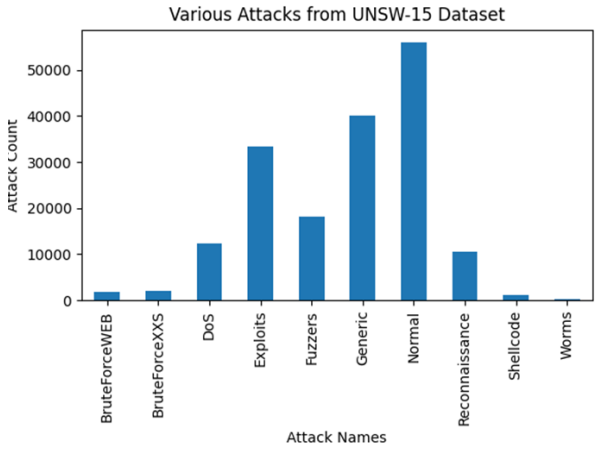

Fig.3. Various attacks in the dataset

4.4 Dataset Train & Test Split

The data set will be split between train and test using this module, with the program using 80% of the data set for training and 20% for testing.

4.5 Train Deep Learning GAN Algorithm

The generalized adversarial network (GAN) algorithm will employ 80% on the training data to develop a model, that will then be applied to 20% of the data to be tested to assess prediction accuracy.The comparison Graph will plot the accuracy and precision comparison between GAN algorithms.

This module will be utilized employed for uploading test data, that GAN will examine to generate a possible signature to determine if the test result is involves attacks.

5. Methodology

DDoS are becoming more sophisticated, it is essential to use intrusion detection systems; since they can help the organization understand the deficiencies in the early remediation. IDS can help organizations monitor illegitimate access that affects the confidentiality, integrity, or availability of information resources. They can also provide the capability to monitor and analyze the activities of both users and systems, assessing the integrity of file and systems, analyzing the unusual sequence of patterns, and possibly tracking if any user is violating the policies laid down by organizations. The signature-based intrusion detection system (IDS) can differentiate normal traffic from DDoS traffic. Techniques based on machine learning have shown promising results in intrusion detection since they can cater the complexity of security demands of a network. Hence, unlike machine learning, deep learning can handle huge amount of data and hence Intrusion detection system based on deep learning can be well suited to detect cyberattacks. They can identify if there is any anomaly in the network traffic from the previous traffic. Another challenge is finding the latest datasets, as much research is available on out-of-date datasets (KDD99 and NSL-KDD). These datasets must be continuously updated with the latest DDoS attack information to prove to be more efficient in detecting novel cyberattacks. In addition, it is recommended that the techniques be enhanced by optimizing the feature selection to enhance the capability to detect cyber-attacks with reduced features[8].

5.1. Proposed Algorithm

A hybrid model combining an auto encoder and a Generative Adversarial Network (GAN) is often referred to as a "GAN-DAE" (GAN with Denoising Auto encoder) or a "GAN-VAE" (GAN with Variational Auto encoder), depending on the type of auto encoder used. Both architectures aim to generate realistic data samples while learning useful representations.

Here's a high-level overview of how you could combine these two models:

5.2. Generator (GAN)

Synthetic data samples have been produced using the generator according to the GAN architecture. It seeks to generate data that is similar to actual data through utilizing random noise as input.

Discriminator (GAN): The discriminator in the GAN framework is responsible for distinguishing between real and synthetic data. It is trained to classify whether the input data is real or generated by the generator.

5.3. Auto encoder

The auto encoder knows how to recombine the data it receives. A decoder network then reconstructs the original data from this representation, when an encoder network compresses the input data into a lower-dimensional latent space representation. Identifying the objective functions for both the reconstruction loss minimization and the adversarial training is required for determining the mathematical formulations for the hybrid model that combines a with a Denoising Auto encoder (DAE). Let us analyse the derivation.

5.4. Generative Adversarial Network (GAN)

The adversarial training aims to minimize the Jensen-Shannon divergence between the distribution of real data and the distribution of generated data.

- Let D(x) represent the discriminator's output, indicating the probability that input x is real.

- Let G(z) represent the generator's output, where z is the input noise vector.
- The discriminator's objective is to maximize the probability of correctly classifying real and fake data:

$$\max_D E_{x\sim p_{data}(x)}[\log D(x)]+E_{z\sim p_z(z)}[\log(1-D(G(z)))] \quad (1)$$

The generator's objective is to minimize the probability of the discriminator correctly classifying fake data:

$$\min_G E_{z\sim pz(z)}[\log(1-D(G(z)))] \quad (2)$$

5.5. Reconstruction Loss Minimization (DAE):

The reconstruction loss minimization aims to minimize the difference between the input data and the reconstructed data. Let E(x) represent the encoder's output, compressing input x into a latent representation. Let D′(E(x)) represent the decoder's output, reconstructing the original input from the latent representation. The objective is to minimize the reconstruction loss between the original input and the reconstructed output:

$$\min_{E,D'} E_{x\sim pdata}(x)[\|x-D'(E(x))\|^2] \quad (3)$$

5.6. Hybrid Model Objective

Where λ1 and λ2 are hyper parameters controlling the relative importance of the adversarial loss and the reconstruction

5.6.1 loss Training Phase

This mathematical formulation provides a framework for training the hybrid model, combining the adversarial training of GANs with the reconstruction loss minimization of DAEs. Adjustments to hyper parameters and network architectures may be necessary for specific applications and datasets. During the training phase, the auto encoder is trained on the real data to reconstruct it accurately. This helps in learning a compact representation of the data.

The GAN framework is trained simultaneously [9]. The generator tries to generate synthetic data samples to fool the discriminator, while the discriminator tries to distinguish between real and synthetic data. The generator can use the latent space representations learned by the auto encoder to generate more realistic data samples. This is achieved by feeding random noise into the decoder part of the auto encoder to generate samples.

5.6.2 Generating Phase:

During the generating phase, you can use the trained generator to produce synthetic data samples. This is done by generating random noise and passing it through the decoder part of the auto encoder, utilizing the learned latent space representation. By combining these models, you can potentially leverage the benefits of both auto encoders (learning useful representations) and GANs (generating realistic data samples). This approach is often used in tasks such as data generation, image synthesis, and anomaly detection[10].

5.6.3 Stacked Auto encoder

A neural network with many layers comprised of multiple auto encoders, each of which feeds its output into the one before it, and so on, until the final encoder delivers its output into an array of decoders, is referred to as a stacked auto encoder. This gives the user greater authority at each stage of the process as they compress and decompress the input data step by step. An ideal stacked auto encoder's output is equal to its input, just like auto encoders [11].

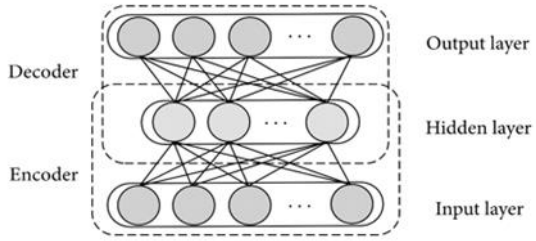

Fig.4. Structure of Auto encoder

The input data is transformed into a hidden representation utilizing an encoder, and the input data is rearranged from the hidden representation utilizing a decoder. They are considering the provided dataset unlabeled.

5.6.4 Single Auto encoder

Primarily composed of an encoder and a decoder, a single auto encoder is the fundamental building component. The encoder transforms the input data into a lower-dimensional representation by compression data; the decoder then uses the resulting representation to reassemble the original data [6].

5.6.5 Stacking Layers

Stacked auto-encoders are produced by layering these separate auto encoders across multiple layers. The output of the previous layer serves as an input for the subsequent layer of the auto-encoder.

5.6.6 Layer-wise Training

Layer-wise training is a common approach to training. To develop an appropriate representation of the input data, the first layer is trained separately as an auto-encoder. The next layer is placed on top of the first layer, which encoder is frozen once it has been trained. The above process is repeated for every stratum within the structure of the hierarchy.

5.6.7 Fine-tuning

Backpropagation is used to adjust the network as a whole once all layers have been stacked. To improve

reconstruction performance overall, all of the network's weights and biases have to be modified. These networks have been utilized in recommendation systems, computer vision, and natural language processing, amongst various other areas. These can be efficient tools for deriving useful features from processed input data.

5.7 Handling imbalanced data

When there are significantly fewer samples in one class than then are observations in the other, there is a class imbalance. Many techniques exist to handle data imbalance (Kraiem et al., 2021). To enhance the minority class samples and balance both classes, we used SVM together with the SMOTE over-sampling strategy. It is an advanced version of SMOTE variations where an SVM classifier is utilized to address the issue of missing data in borderline [8]. In addition, after training SVM on every piece of data, the border is established utilizing support vectors. Subsequently, a random sequence of lines is created to connect each minority class support vector to a subset of the closest neighbors in the synthetic data. The conventional GAN model is a generative model using DL architectures. Goodfellow et al. (2014) described the first GAN model which is difficult to train and unstable. In general, GAN model has a generator model and a discriminator model. The figure shows the generator model. The GAN generator model takes random vector data with fixed length as input and outputs the generated examples in the domain. A source of noise (random vector) is required for the generative purposes. They are obtained from the Gaussian distribution. After training, a compressed representation of data distribution is formed as called latent space. From the latent space, the generator model takes new points as input and generates new examples. The generated examples are discriminated by a discriminator model shown in figure 5.

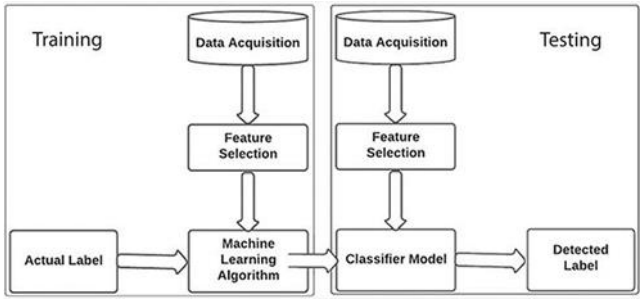

Fig.5. Block diagram of the GAN with Auto encoder system

To enhance data quality and extract useful knowledge, the model starts with data pre-processing, an essential initial stage in any machine learning process. Organizing and cleaning raw data to make it available for the development and training of machine learning models is referred to as data pre-processing. Initially, multiple statistical summaries and visualizations are implemented when combined with EDA to assist in recognizing the key characteristics of the various input data entities, such as rows and columns [7]. Depending on the input data, this method can be costly yet very successful. Concerning the data, the EDA generates several observations.

6. Scalability of Proposed Model

Model Size: more layers in the autoencoder can increase the model size this can potentially lead to higher memory requirements during training and inference, making scalability an issue on resource-limited systems.

Computational Resources: Training a stacked autoencoder requires significant computational resources, especially when dealing with large datasets or deep architectures. Scaling to larger datasets UNSW-NB15 or deeper networks may require distributed computing or specialized hardware (e.g., GPUs or TPUs) to speed up training. In this work, we used 12GB GPU in Google Colab.

Data Dimensionality: Stacked autoencoders may face challenges with high-dimensional input data. As the dimensionality of the input data increases, the complexity of learning meaningful representations also increases. Techniques such as dimensionality reduction or feature engineering may be necessary to improve scalability.

Generalization: With deeper architectures, there is a risk of overfitting, especially when dealing with limited training data. Regularization techniques such as dropout or L2 regularization can help mitigate overfitting and improve the generalization of the model.

Hyperparameter Tuning: As the complexity of the model increases, the number of hyper parameters also grows. Tuning these hyper parameters effectively becomes crucial for achieving good performance and scalability.

Deployment: Deploying stacked auto encoders in production environments may pose challenges, especially if real-time inference or low-latency requirements are necessary. Optimizing the model for inference speed and memory footprint is essential for scalable deployment. In summary, while stacked auto encoders offer powerful capabilities for learning hierarchical representations of data, achieving scalability requires careful consideration of model size, computational resources, data dimensionality, generalization, hyperparameter tuning, and deployment constraints. Proper optimization and tuning are crucial for effectively scaling stacked auto encoder models to larger datasets and deeper architectures.

7. Experimental Results

Computer network always get affected because of viruses and worms which can be spread by malicious attackers by intercepting communication between two computers in a network. In the past many algorithms were introduced but all those algorithms were depend on some signature matching to distinguish between normal and attack network packets and this algorithms accuracy will get decrease if

attackers used different signatures.

To overcome from above issue author employing deep learning algorithms to detect attacks from IOT networks. In propose work author has utilized unsupervised based Generative Adversarial Network (GAN) deep learning algorithm which is based on two different networks called Generator and Discriminator. Generator is mainly responsible to generate new dataset based on original dataset and then Discriminator will predict weather generated new test data is normal or contains any attack signatures. So, by using this model we can detect attacks on same or different variant signature.

Attackers cannot evade from detection even if they change signature as Generator can detect signature from all possible values and can detect all signatures of attacks. To train and test above algorithm performance we have utilized UNSW15 dataset which is suitable to train attacks from normal or IOT networks. This dataset contains 9 different attacks signature.

GAN model able to achieve accuracy between 95 to 97% and this performance we are measuring in terms of precision, recall, confusion matrix graph and F1score shown in figure 6

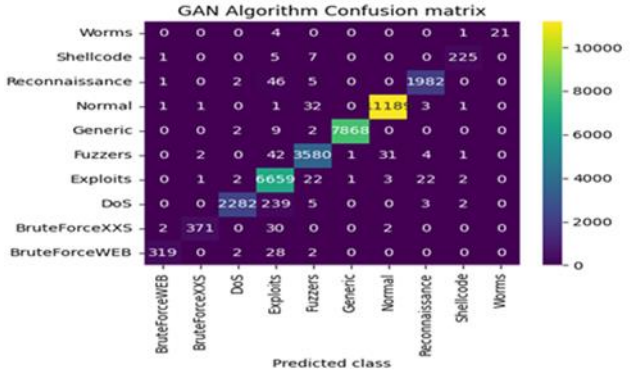

Fig.6. Confusion matrix

Table.1 Comparison of evaluation parameters

Performance	CNN	DNN	Proposed system
Accuracy	0.73	0.85	0.9836
F1score	0.74	0.72	0.9843
precision	0.72	0.76	0.9417
sensitivity	0.77	0.73	0.9614

Table 1 provided performance metrics for different models: Convolution Neural Network, Deep Neural Network, and a proposed system. These metrics include Accuracy, F1 Score, Precision, and Sensitivity. Comparison graph is shown in figure 7.

7.1 Accuracy

The proportion of correctly classified instances out of the total instances. For example, an accuracy of 0.73 for the CNN model means that it correctly classified 73% of the instances.

7.2. F1 Score

The harmonic mean of precision and sensitivity. It provides a balance between precision and recall. It's a good metric to use when there is an imbalance between the classes in the data.

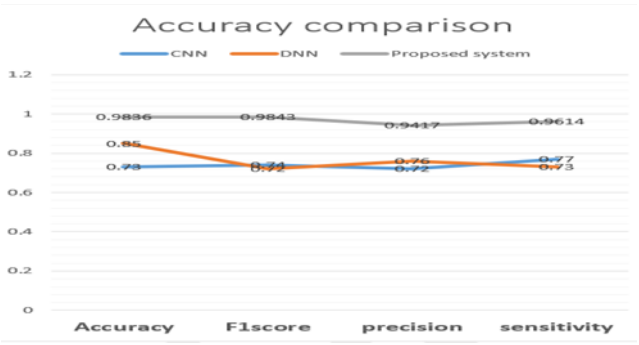

Fig.7. performance analysis graph

7.3. Precision The proportion of true positive instances (correctly predicted positive instances) out of all positive predictions. A precision of 0.72 for the CNN model means that out of all instances predicted as positive by the model, 72% were positive.

7.4. Sensitivity (Recall)

The amount of actual instances that were positive that the model properly recognized. A sensitivity of 0.77 for the CNN model means that out of all actual positive instances, the model correctly identified 77%. Based on these metrics, the hybrid system seems to perform the best overall, with the highest scores in Accuracy, F1 Score, Precision, and Sensitivity.

8. Conclusion

Based on the above result analysis, we can conclude that the hybrid model combining an auto encoder with a GAN can offer advantages in terms of data generation, representation learning, unsupervised learning, and transfer learning compared to traditional CNN and DNN architectures, especially in tasks were generating realistic data samples and learning informative representations are crucial. The synergistic interaction between the two components facilitates a more informative latent representation and prevents mode collapse, leading to better performance and generalization across various domains and tasks. Additionally, the hybrid model benefits from regularization effects, reducing overfitting and improving model versatility, making it a powerful framework for a wide range

of generative modeling applications.

References

[1] Pekta¸s, A.; Acarman, T. A deep learning method to detect network intrusion through flow-based features. Int. J. Netw. Manag. 2019, 29, e2050.

[2] Sindian, S.; Samer, S. An enhanced deep auto encoder-based approach for DDoS attack detection. Wseas Trans. Syst. Control 2020, 15, 716–725

[3] Kushwah and Ali, Survey of intrusion detection systems: Techniques, datasets, and challenges. Cybersecurity 2019, 2, 20

[4] Elsayed et al,.; De Roure, D.; Page, K.; Van Kleek, M.; Santos, O.; Maddox, L.T.; Burnap, P.; Anthi, E.; Maple, C. Design of a dynamic and self-adapting system, supported with artificial intelligence, machine learning and real-time intelligence for predictive cyber risk analytics in extreme environments–cyber risk in the colonization of Mars. Saf. Extrem. Environ. 2020, 2, 219–230.

[5] Raikar, M.M.; Meena, S.; Mulla, M.M.; Shetti, N.S.; Karanandi, M. Data traffic classification in software-defined networks (SDN) using supervised-learning. Proc. Comput. Sci. 2020, 171, 2750–2759.

[6] Perez-Diaz, J.A.; Valdovinos, I.A.; Choo, K.-K.R.; Zhu, D. A flexible SDN-based architecture for identifying and mitigating low-rate DDoS attacks using machine learning. IEEE Access 2020, 8, 155859–155872

[7] Raikar, M.M.; Meena, S.; Mulla, M.M.; Shetti, N.S.; Karanandi, M. Data traffic classification in software-defined networks (SDN) using supervised-learning. Proc. Comput. Sci. 2020, 171, 2750–2759

[8] Kraiem et al and Qinghui Liu, "A comprehensive review study of cyber-attacks and cyber security; Emerging trends and recent developments", Elsevier Energy Report (ISSN: 8176–8186), 2021

[9] Beny Nugraha and Rathan Narasimha Murthy, "Deep Learning-based Slow DDoS Attack Detection in SDN-based Networks", IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2020.

[10] Amitha Mathew, P. Amudha, and S. Sivakumari, "Deep Learning Techniques: An Overview", International Conference on Advanced Machine Learning Technologies and Applications, © Springer Nature Singapore Pte Ltd. 2021. https:// doi.org/ 10.1007/978-981-15-3383-9_54

[11] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A.A. Ghorbani, "Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy", IEEE 53rd International Carnahan Conference on Security Technology, Chennai, India, 2019.