Anomaly Based Intrusion Detection System in Network

Introduction: Telecommunication advancements are swiftly progressing, fostering greater global connectivity annually. Contemporary industries, like automotive transport, smart agriculture, and public safety, are embracing new standards and technologies such as Internet-of-Things (IoT) devices. An average IoT setup comprises three tiers: the perception layer, network layer, and application layer**.** The perception layer detects and collects environmental data, forwarding it to the network layer. For instance, surveillance cameras employ sensors to identify anomalies like motion. Acting as a conduit between the perception layer and the cloud, the network layer employs various internet protocols and communication technologies like Zigbee, 5G, MQTT, and Wi-Fi for data exchange [**1**]. For instance, a surveillance camera might utilize the home router and Wi-Fi to transmit a motion detection signal to the central server. Finally, the application layer processes the data from the network layer to deliver necessary operations or services to users [**2**]. For instance, a cloud service could alert a homeowner via a mobile app that motion was detected by one of their surveillance cameras. The Internet of Things (IoT) is exposed to security risks across all its architectural levels and has been grappling with security issues since its inception [**3**]. Broadly, the perception layer faces threats such as malicious code injection, eavesdropping, and interference [**2**], [**4**]. Similarly, the network layer is vulnerable to spoofing, denial of service, man-in-the-middle, and routing information attacks [**4**]. Additionally, the application layer is susceptible to viruses, worms, and phishing attempts. Andrea et al. [**5**] categorized these attacks into four groups: physical, software, network, and encryption. A physical attack occurs when the perpetrator is physically close to the system, while a software attack involves exploiting bugs to gain unauthorized access. Network attacks manipulate IoT networks to cause harm, while encryption attacks compromise IoT encryption protocols.

A botnet represents a targeted form of cyber attack leveraging Internet of Things (IoT) devices. Angrishi et al. [**6**] define a botnet as a large collection of internet-connected devices manipulated to inundate a specified server (or servers) with simultaneous requests, rendering it incapable of responding to genuine requests, effectively halting its operation. This assault constitutes a distributed denial of service (DDoS) attack. There are two primary techniques employed in such attacks: reflection and amplification, both resulting in the depletion of the target's bandwidth and resources. These attacks have become increasingly sophisticated, making detection challenging [**6**]. IoT botnets pose a threat not only to the owners of IoT devices but also to all internet users. Because DDoS attacks require

substantial network traffic to disrupt services, IoT devices are ideal hosts due to their vast numbers and generally weak security measures, rendering them easy targets [**7**].

In their research, Das et al. [**8**] illustrate Mirai, a recent example of such a botnet, wherein a virus seeks out susceptible devices and links them to Command-and-Control Servers (C&C servers). Das et al. [**8**] and Tushir et al. [**9**] observed that IoT devices linked to Mirai Botnets are primarily utilized to execute DDoS assaults on targeted devices. Tushir et al. [**9**] investigated the impact of Mirai attacks on IoT devices, revealing a 40% surge in energy consumption and a 50% increase in storage usage. The severity of the threat posed by botnets became evident in 2016 with the release of the Mirai botnet, infecting 4000 devices per hour and amassing approximately half a million actively infected devices, orchestrating a groundbreaking 1.1Tbps attack. The infected IoT devices were distributed across 164 countries. Subsequently, numerous iterations and variants of the Mirai botnet emerged, such as Persirai, Hajime, and BrickerBot [**10**]. Targets of DDoS attacks encompassed websites, cloud providers, individuals, educational institutions, telecommunication companies, and DNS providers (Dyn), which serviced multiple websites including Reddit, Amazon, Spotify, Airbnb, among others [**6**].

As per Statista [**11**], the global count of IoT-connected devices reached nearly 8.74 billion in 2021, with a Cisco white paper [**12**] projecting a rise to approximately 30 billion by 2023, compared to roughly 18 billion in 2018. By 2024, it's estimated that there will be 83 billion devices connected to the Internet of Things (IoT) **[13]**. Moreover, the same paper **[12]** anticipates a surge in Distributed Denial of Service (DDoS) attacks to around 15 million by 2023, contrasting with 7 million recorded in 2018 [**14**]. According to statistics, the frequency of attacks is doubling annually, resulting in significant financial losses, amounting to tens of millions of dollars specifically from ransomware attacks [**15**].

One effective strategy for thwarting such assaults involves implementing a robust Intrusion Detection System (IDS) [**15**] capable of identifying various forms of intrusion. Presently, IDSs employ two main detection approaches: signature-based and anomaly-based. Signature-based detection systems are hindered in their effectiveness by their incapacity to recognize emerging cyber threats and their reliance on manual updates to the signature database which can be laborious. Conversely, anomaly-based methods analyze data, relying on the system's comprehension of typical behavior to flag any incoming connections that appear aberrant. Network intrusion detection seeks to assess diverse network data using different behavioral analyses to uphold its security. Numerous methods exist for

detecting anomalies in networks. Although machine learning has proven indispensable and efficient in promptly identifying cyber-attacks, Deep learning using extensive datasets for training can potentially avoid overfitting issues, as it possesses greater capacity for generalization compared to conventional learning models [**14**]. Although several anomaly detection techniques are employed, there have been fewer comparative studies of different deep learning models for anomaly detection. Since intrusions entail a sequence of linked malevolent actions executed by an internal or external perpetrator to compromise the security of the designated system, our attention will be directed towards Recurrent Neural Networks, primarily tailored for sequential data processing.
Constructing an IDS for automatic cyber-attack detection necessitates a suitable dataset for training. We are going to explore the IoT-23 dataset by Garcia et al. [**16**] is a recent release specifically tailored to address cyber-attacks involving IoT devices, introduced in early 2020. Our proposed IDS use several deep learning-based models for the binary classification. Several factors for selecting binary classification are listed below.

- Binary classification simplifies the problem to distinguishing between normal (benign) and abnormal (malicious) activities. This can make the system easier to design, implement, and maintain.
-Since the model only needs to learn two classes, training and prediction can be faster and require less computational resources.
-By focusing on identifying anomalies as a whole, the IDS can be more robust in catching new or unknown types of attacks that deviate significantly from normal behavior.

Literature Review: Li et al. [17] introduced several methods utilizing convolutional neural networks (CNNs) for intrusion classification. They divided the dataset into four segments based on feature correlations and transformed the one-dimensional feature data into a grayscale graph through clustering. Each segment was individually trained and tested using the same CNN architecture (CNN1, CNN2, CNN3, CNN4) for binary classification. Additionally, they created a combined model, CNN0, which integrated the outputs of the four separate CNNs and was trained on the entire dataset. All models were trained on the NSL-KDD dataset [18] and evaluated on the $KDDTest^{+}$ and $KDDTest^{-21}$ datasets. The CNN1 model achieved the highest accuracy on both test sets, with 82.62% and 67.22% respectively. The ensemble model, combining all four CNNs, achieved even higher accuracies of 86.95% and 76.67% on the two test sets. Xu et al. [19] introduced an autoencoder (AE) model utilizing Long Short-Term Memory (LSTM) for intrusion detection. They trained this model using five distinct datasets: ARP, Fuzzing,

Mirai, SSDP Flood, and Video Injection, provided by the Mirsky team [20]. Each dataset was trained separately with a binary classification approach. Since each dataset contained only one type of attack, it simplified the pattern recognition for the model. They then compared their model's performance with two traditional machine learning techniques. Their model achieved F1-scores between 92.4 and 96.8, consistently outperforming Support Vector Machine (SVM), K-Nearest Neighbors (KNN), a general Autoencoder (AE), and a stacked autoencoder. Lopez-Martin et al. [21] proposed a model that uses a linear classifier based on a Neural Network (NN) with linear activations. This model incorporates feature transformations using kernel approximation algorithms such as Nystrom, Random Fourier Features, and Fastfood transformation, which add the necessary complexity and non-linear characteristics to the model. To test their model, they chose three datasets but only performed binary classification on the NSL-KDD [18] dataset. The highest accuracy achieved in this binary classification was 80%. As evident, the results were not particularly promising. Kim et al. [22] developed a hybrid model integrating a convolutional neural network (CNN) and a long short-term memory network (LSTM) for the binary classification. They tested this model on two publicly available datasets, CSIC-2010 and CICIDS2017, achieving accuracies of 91.5% and 93.0%, respectively. Susilo et al. [23] utilized three algorithms—Random Forests, Multilayer Perceptron (MLP), and Convolutional Neural Network (CNN)—to identify network intrusions. They employed the Bot-Iot dataset, created by UNSW Canberra [24] for multi-class classification. The CNN algorithm achieved the highest accuracy, reaching 91.25%. Yin et al. [25] introduced a deep learning method for intrusion detection using recurrent neural networks (RNN-IDS) and assessed its performance in both binary and multiclass classification tasks. They trained their model using the KDDTrain+ dataset and tested it with the $KDDTest^{+}$ and $KDDTest^{-21}$ datasets, the latter being a subset of the former. By experimenting with different hyperparameters (such as the number of nodes and learning rate), they achieved the highest accuracy of 83.28% on the $KDDTest^{+}$ dataset and 68.55% on the KDDTest21 dataset, with the optimal configuration being 80 hidden nodes and a learning rate of 0.1. Li et al. [26] employed two recurrent neural networks, specifically Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), with the Border Gateway Protocol (BGP) and NLS-KDD datasets. The BGP dataset comprised three tiny and highly imbalanced separated datasets. Both models achieved accuracies between 90% and 95% on these subsets. The models were trained with KDDTrain+ and tested with KDDTest+ and KDDTest-21, resulting in performance ranging from 80% to 83%. Imrana et al. [27] introduced a bidirectional Long-Short-Term-Memory (BiDLSTM) model for the binary classification. Their dataset included a training set with 125,973 traffic samples from KDDTest+, and two distinct test sets:

KDDTest+ with 22,544 samples and KDDTest-21 with 11,850 samples. The datasets were well-balanced. To demonstrate the superiority of their model, they also implemented a conventional LSTM model, which achieved accuracies between 80% and 90%. In contrast, their proposed model achieved accuracies between 88% and 94%. Yang et al. [28] introduced a fuzzy aggregation method that combines the modified density peak clustering algorithm (MDPCA) with deep belief networks (DBNs). DBNs are multi-layer probabilistic generative models consisting of several layers of restricted Boltzmann machines (RBMs) with a classifier on the top layer, designed to learn a deep hierarchical representation of the training data. In MDPCA, the Gaussian kernel is utilized to measure distances in high-dimensional space, enhancing the clustering of complex data. To address the issues of training set size and sample imbalance, MDPCA divides the training set into multiple subsets with similar attribute sets. Each subset trains its own sub-DBNs classifier, and the outputs of all sub-DBNs classifiers are aggregated using fuzzy membership weights. Their model was tested on the NSL-KDD and UNSW-NB15 datasets, achieving highest accuracies of 82.08% on NSL-KDD and 90.21% on UNSW-NB15. Wu et al. [29] introduced a multiclass classification model utilizing Convolutional Neural Networks (CNN) to automatically extract significant traffic features from raw data. To tackle the problem of imbalanced datasets and enhance the detection rates of rare attack types, the cost function was modified based on class distribution. Additionally, the raw traffic data was transformed into an image format to boost the CNN's performance. The model was trained using the $KDDTrain^{+}$ dataset, achieving accuracy rates of 79.48% on the $KDDTest^{+}$ dataset and 60.71% on the $KDDTest^{-21}$ dataset. Naseer et al. [30] presented a comprehensive study on the application of deep neural networks for network anomaly detection. By leveraging various DNN architectures like CNNs, Autoencoders, LSTMs and comparing them with conventional machine learning methods like Support Vector Machine (SVM), Random Forest (RM), the authors demonstrate the superior performance of deep learning models in handling the complexities of modern network traffic data. The models were trained on the NSL-KDD dataset and evaluated on the NSL-KDDTest+ and NSL-KDDTest-21 test datasets. The model that achieved the greatest accuracy was the LSTM, with a performance rate of 89%. Ding et al. [31] suggested training an Intrusion Detection System (IDS) model using Convolutional Neural Networks (CNN) for multiclass classification. This model is tailored to process the one-dimensional nature of network traffic data by treating it as a three-dimensional image with a height of 1 and a single channel, making it compatible with CNN input requirements. They utilized the NSL-KDD dataset and achieved a maximum accuracy of 80.13%. Vinayakumar et al. [32] proposed a collaborative method combining Network-based Intrusion Detection Systems (NIDS) and Host-based Intrusion Detection

Systems (HIDS). They conducted various experiments with Deep Neural Networks (DNNs) and several machine learning classifiers to create an optimal intrusion detection system for both binary and multiclass classifications. They employed advanced text representation techniques like TF-IDF from natural language processing with DNNs for host-level events. Additionally, they introduced a scalable hybrid intrusion detection framework (SHIA) designed to handle large datasets in real time, enhancing scalability and detection accuracy. Their highest achieved accuracies were 92.90% for binary classification and 93.00% for multiclass classification on the KDD99 dataset utilizing DNN. Chouhan et al. [33] offered a novel Channel Boosted and Residual learning based deep Convolutional Neural Network (CBR-CNN) architecture for the detection of network intrusions. The proposed method, termed Channel Boosted and Residual learning-based CNN (CBR-CNN), integrates the concept of channel boosting and residual learning to enhance the detection of network anomalies. The architecture includes a new architectural block based on the concept of Split-Residual Transform-Merge, aiming to create a lightweight yet effective CNN classifier. The detection system models normal network traffic using Stacked Autoencoders (SAE). This transforms the original feature space into a reconstructed feature space, which helps in distinguishing between normal and anomalous traffic. The CNN architecture is designed to learn features at different levels of granularity, enhancing its representational capacity with fewer parameters and layers. The performance of the CBR-CNN method is evaluated on the NSL-KDD dataset and achieved 89.41% of accuracy. Sokolov et al. [34] explored the use of Recurrent Neural Networks (RNNs), specifically Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), for intrusion detection for the binary classification in Industrial Control Systems (ICS), due to their ability to handle sequential data. The study emphasized the importance of considering both network traffic and the state of industrial processes for effective intrusion detection. The experiments compared the performance of LSTM and GRU networks in detecting intrusions using the Gas Pipeline dataset [35][36]. GRU networks showed slightly better performance with an accuracy of 91.70% compared to LSTM's 90.68%. The study found that GRU networks learn faster and are computationally more efficient than LSTMs. Sokolov et al. [37] introduced the Difficult Set Sampling Technique (DSSTE) algorithm to address the class imbalance issue in network intrusion detection systems (NIDS) for the multiclass classification. The algorithm improves the learning process for classifiers by focusing on the minority class, which represents the malicious activities often hidden within the majority normal traffic. Edited Nearest Neighbor (ENN) algorithm was used to divide the imbalanced training set into a difficult (near-neighbor) set and an easy set (far-neighbor). Then, KMeans Clustering compressed the majority samples in the difficult set to reduce

their dominance. After that, Zoom Augmentation technique increased the number of minority samples by adjusting their continuous attributes to synthesize new samples. The effectiveness of the DSSTE algorithm was validated on the NSL-KDD and CSE-CIC-IDS2018 datasets. The experiments involved multiple classification models, including Random Forest, Support Vector Machine, XGBoost, Long Short-Term Memory (LSTM), AlexNet, and Mini-VGGNet. They achieved a peak accuracy of 82.84 using DSSTE+AlexNet on the NSL-KDD KDDTest+ dataset. **In any classification, an imbalanced dataset means one class significantly outnumbers the other. If the dataset is imbalanced, the model tends to become biased toward the majority class, leading to poor performance in detecting the minority class (anomalies). Therefore, a balanced dataset helps the model generalize better to new, unseen data. Also, high accuracy on an imbalanced dataset can be misleading. For example, if 95% of the data is normal and 5% is anomalous, a model predicting every instance as normal would achieve 95% accuracy. However, this model would completely fail to detect intrusions. Balancing the dataset helps ensure that accuracy and other metrics truly reflect the model's performance. Moreover, balancing the dataset often leads to better training dynamics. Gradient-based learning algorithms, like those used in many deep learning models, can benefit from balanced datasets as the gradients are more stable and representative of both classes, facilitating better convergence during training. For these reasons, we first balanced our dataset. To achieve a more comprehensive evaluation, we included additional metrics such as the False Positive rate and the False Negative rate, which were not discussed in most of the previously mentioned papers. Finally, achieving a balance between false negatives and false positives is often the goal. However, the operational context of the IDS deployment also influences the prioritization. For instance, in a more stable and predictable environment, reducing false positives might be more critical to maintaining operational efficiency. In contrast, in a highly dynamic network environment with constantly changing traffic patterns, an IDS with high sensitivity (low false negative rate) might be preferred to adapt quickly to emerging threats. Also in some high-security environments (e.g., military, financial institutions), ensuring that no genuine intrusion goes undetected (low false negative rate) might be paramount, even if it results in more false positives. Frequent false alarms can lead to alert fatigue which can be ignored, where a missed detection could result in a successful attack going unnoticed, leading to data breaches or system compromise. Hence, our primary aim is to minimize false negatives while maintaining an acceptable false positive rate.**

Dataset

This study utilized the IoT-23 dataset, which comprises network traffic collected from Internet of Things (IoT) devices. It includes 20 instances of malware and 3 instances of benign activity. The aim of this dataset is to provide a substantial collection of real-world labeled IoT malware infections and benign IoT traffic, facilitating the development of machine learning algorithms for researchers. It consists of 23 captures, referred to as scenarios, wherein 20 involve malicious activity and 3 involve benign activity. Each capture from infected devices may include the name of the potential malware sample executed in that scenario.

Here are the descriptions for each label in the IoT-23 dataset:

Attack: This denotes a form of assault originating from the infected device towards another host, exploiting vulnerabilities.

Benign: Indicates that no suspicious or malicious activities were detected within the connections.

C&C: Signifies that the infected device established a connection with a Command and Control (C&C) server.

DDoS: Indicates that the infected device is executing a Distributed Denial of Service attack.

FileDownload: Represents the downloading of a file to the infected device.

HeartBeat: Refers to packets sent through this connection to monitor the infected host by the C&C server.

Mirai: Implies connections displaying characteristics of a Mirai botnet.

Okiru: Implies connections displaying characteristics of an Okiru botnet.

PartOfAHorizontalPortScan: Denotes connections utilized for conducting a horizontal port scan to gather information for subsequent attacks.

Torii: Indicates connections demonstrating characteristics of a Torii botnet.

Furthermore, Zeek functions as software for conducting network analysis. The IoT-23 dataset utilized is in the conn.log.labeled format, derived from the original pcap file through Zeek network analyzer. Table 1 displays the description of the IoT-23 dataset.

Table 1 illustrates the distribution of flows within each category (label) derived from the labeled dataset.

Table 2 ([44], [45]) displays the dataset comprising 20 attributes.

Specific attributes, such as conn_state and history, hold significance with values or characters carrying particular implications. Refer to Table 3 and 4 from [45] for descriptions of these attributes values.

Table 1: Number of flows for each category

	Category	Number of flows
1	PartOfAHorizontalPortScan	825939
2	Okiru	262690
3	Benign	197809
4	DDoS	138777
5	C&C	15100
6	Attack	3915
7	C&C-HeartBeat	349
8	C&C-FileDownload	43
9	C&C-Torii	30
10	FileDownload	13
11	C&C-HeartBeat-FileDownload	8
12	C&C-Mirai	1
	Total	**1444674**

Table 2: Features Description

	Feature	Description
1	ts	The time of the first packet
2	uid	A unique identifier of the connection
3	proto	The transport layer protocol of the connection
4	id.orig_h	Originator/Source IP Address
5	id.orig_p	Originator/Source Port number
6	id.resp_h	Responder/Destination IP Address
7	id.resp_p	Responder/Destination Port number
8	service	An identification of an application protocol

9	duration	How long the connection lasted
10	orig_bytes	The number of payload bytes the originator sent
11	resp_bytes	The number of payload bytes the responder sent
12	conn_state	The possible connection state values
13	local_orig	If the connection is originated locally, this will be T and F for remotely
14	local_resp	If the connection is responded locally, this will be T and F for remotely
15	missed_bytes	Indicate the number of bytes missed in content gaps
16	history	Records the state history of connections as a string
17	orig_pkts	Number of packets that the originator sent
18	orig_ip_bytes	Number of IP level bytes that the originator sent
19	resp_pkts	Number of packets that the responder sent
20	resp_ip_bytes	Number of IP level bytes that the responder sent

Table 3: Description of conn_state values		
	State	Meaning
1	S0	Connection attempt seen, no reply
2	S1	Connection established, not terminated (0 byte counts)
3	SF	Normal establish & termination (>0 byte counts)
4	REJ	Connection attempt rejected
5	S2	Established, ORIG attempts close, no reply from RESP.
6	S3	Established, RESP attempts close, no reply from ORIG.
7	RSTO	Established, ORIG aborted (RST)
8	RSTR	Established, RESP aborted (RST)
9	RSTOS0	ORIG sent SYN then RST; no RESP SYN-ACK
10	RSTRH	RESP sent SYN-ACK then RST; no ORIG SYN
11	SH	ORIG sent SYN then FIN; no RESP SYN-ACK (“half open”)
12	SHR	RESP sent SYN-ACK then FIN; no ORIG SYN
13	OTH	No SYN, not closed. Midstream traffic. Partial

Table 4: Description of history values (for the originator, the value is in uppercase and lower for the responder)	
State	Meaning
S	a SYN without the ACK bit set
H	a SYN & ACK (“handshake”)
A	a pure ACK
D	packet with payload (“data”)
F	packet with FIN bit set
R	packet with RST bit set
C	packet with a bad checksum
I	Inconsistent packet (Both SYN & RST)

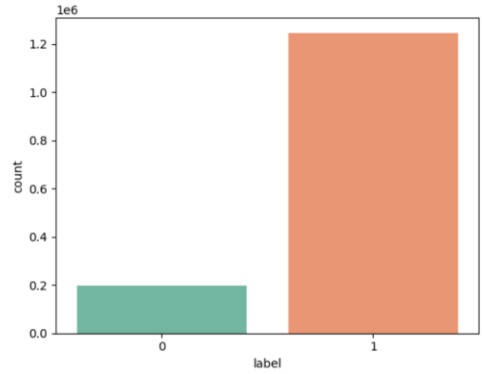
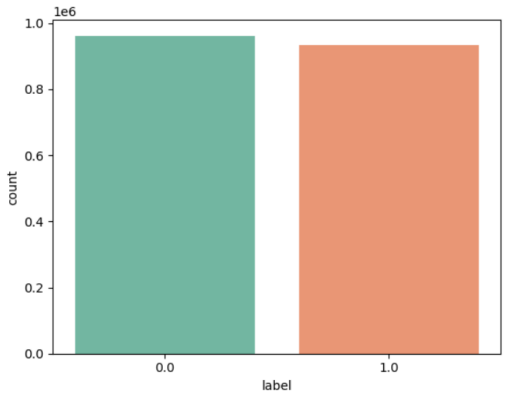
We encountered a minimal number of missing values, which we addressed by substituting them with zeros. Features `local_orig', `local_resp' which were empty for all the files (scenarios) and hence they were dropped. Based on previous work on pre-processing of intrusion detection datasets like NSL-KDD, CICIDS2017, features id.orig_h, id.orig_p, id.resp_h and id.resp_p contained IP adresses and port numbers were dropped. Additionally, the 'history' attribute, representing a sequence detailing the connection history, was initially dropped.

Since our focus was on binary classification, we assigned a value of '1' for all attack instances and '0' for benign instances in the "label" column using Python's 'map' function. Following the encoding of object type features, we examined each column and observed that the majority of values in certain features were zero. Consequently, we decided to drop those features from consideration as well [Table 5].

Table 5: Additional Dropped features after encoding

Column	Unique Values	Value Counts
Conn_state_RSTO	0	1444521
	1	153
Conn_state_RSTOS0	0	1444644
	1	30
Conn_state_RSTR	0	1444123
	1	551
Conn_state_S2	0	1444647
	1	27
Conn_state_S3	0	1444217
	1	2457

After removing all those columns, we now have a total of 15 columns, including the label column.

 1 1246865 0 197809	 0.0 961750 1.0 934971
Imbalanced	Balanced
Fig. 1	

If we notice Figure 1(Left) above, we'll see a higher prevalence of attacking data compared to benign data. If we employ an algorithm on this dataset, it may exhibit bias towards detecting intrusions. Thus, it's necessary to augment the benign instances to rebalance the dataset while retaining its information. To achieve this, we utilized a semi-supervised technique. Initially, we divided the dataset into two segments: training set (75%) and test set (25%). Subsequently, we constructed a sequential deep neural network (DNN) comprising four layers, culminating in an output layer with a single node for binary classification. Training the network on the training set yielded an accuracy of 89.3%.

Our objective now is to generate some randomly distributed normal data:

We first constructed a dataset comprising only benign/normal rows, identified by a label column value of '0'.

Next, we calculated the standard deviations of all 14 columns using the 'std()' function.

We then scaled down each standard deviation by a factor of 0.1 to reduce and normalize the deviation values, converting them into a 'numpy' array.

A function named 'random_val()' was created to generate random values using a normal distribution via the 'tf.random.normal' function, with parameters including

a seed value range of 1 to 256, a shape of 148534 rows and 14 columns, a mean of 0, and standard deviations derived from each column of the normal data.

Finally, we employed our trained model to predict benign data from the generated random values. By iterating through a loop, we obtained the desired quantity of benign data and appended it to the original benign data. Following this procedure, we eliminated the duplicated entries and ultimately obtained a dataset that is reasonably balanced, as depicted in Figure 1(Right).

Methodologies

Deep Neural Networks model

Deep Neural Networks (DNNs) represents a subset of machine learning methods inspired by the structure and function of the human brain. Their primary objective is to replicate the brain's information processing capabilities. DNNs excel at recognizing intricate patterns and correlations within datasets, a capability that has propelled their popularity in recent years owing to advancements in computational power and data availability.

A defining feature of DNNs is their layered architecture, characterized by interconnected nodes. In a dense or fully connected DNN, these layers include an input layer and one or more hidden layers arranged sequentially. Each node within these layers receives input from preceding nodes or the input layer. This iterative process continues through the network until the final layer generates the desired output.

The activation function determines whether a neuron should be triggered based on the weighted sum plus bias calculation. Its role is to inject non-linear characteristics into the neuron's output. Fig. **1** shows the basic structure of a DNN. As the figure shows, our proposed sequential deep neural network (DNN) operates with an input feature ($X_1 \ldots X_n$) size of 14, incorporating four hidden layers along with one output layer (O).

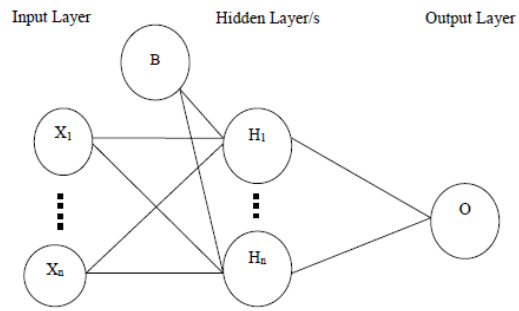


Fig. 2

Utilizing the Rectified Linear Unit (ReLU) activation function across all hidden layers with bias, B, our model employs a linear function that activates a node only when the weighted sum of its inputs is positive; otherwise, it deactivates the node permanently meaning during the backpropagation only the activated nodes work, resulting in faster computation due to its simplicity. For the output layer, given the binary classification nature of the task, we employ the "sigmoid" activation function, which scales the output values between zero and one.

Backpropagation, also known as backward propagation of errors, is an algorithm that traces errors from output nodes back to input nodes. It calculates the gradient of the loss function for each weight using the chain rule, progressing layer by layer. This iterative process starts from the final layer, preventing redundant computation of intermediate terms.

Connections between nodes in different layers are assigned weights, determining the extent of influence one node has on another. Throughout the training phase, the neural network fine-tunes these weights to minimize the error between predicted and actual outputs. Techniques such as backpropagation and gradient descent facilitate this adjustment process, enhancing the network's predictive or classificatory accuracy.

As data traverses from one node to another, the neural network progressively gains insights, culminating in the production of the desired output from the output layer.

Our model is optimized using the “adam” optimizer, a widely used and optimized gradient descent algorithm, with the “binary_crossentropy” serving as the loss function.

Layer (Type)	Output Shape	Number of parameters
Input(Dense)	(None, 100)	1500
Dense	(None, 50)	5050
Dense	(None, 40)	2040
Dense	(None, 40)	1640
Output(Dense)	(None, 1)	41
Total parameters: 10,271 Trainable parameters: 10,271 Non-trainable parameters: 0		

Table 6: Arch of DNN

Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a type of artificial neural network designed for sequence processing tasks, where the input data is presented in a sequential order. The main and most important feature of RNN is its Hidden state, which remembers some information about a sequence. RNNs are trained using backpropagation through time, an extension of backpropagation algorithm. BPTT calculates gradients with respect to the parameters of the network by unfolding it through time and applying the chain rule.

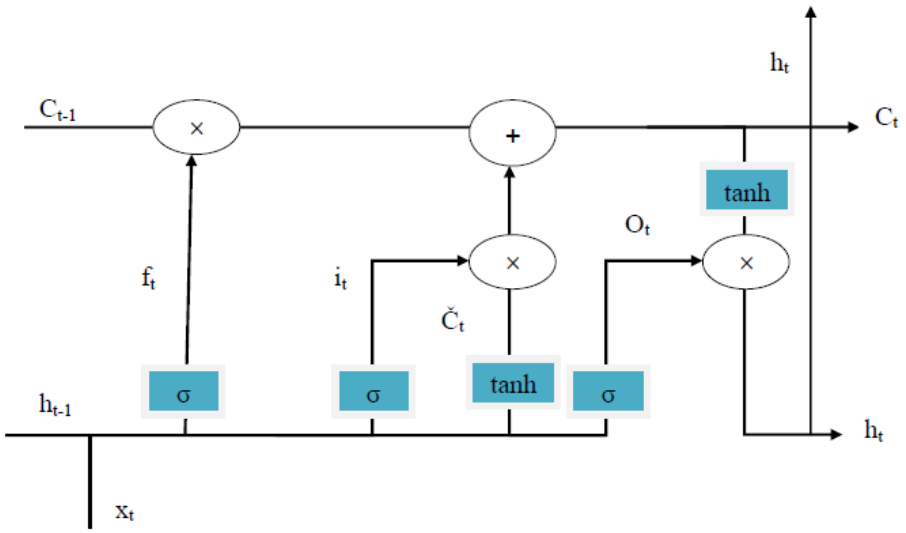
We employed a basic sequential RNN architecture comprising two hidden layers, each containing only 20 nodes, as increasing it to 40 nodes resulted in decreased accuracy. The 'tanh' activation function was utilized as the default. For the Output Layer, a single-node dense layer with a sigmoid activation function was employed to yield a single output. Despite these configurations, unstable gradients and short-term memory issues hindered achieving satisfactory accuracy. Consequently, we explored alternatives such as Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRUs).

Layer (Type)	Output Shape	Number of parameters
Input(SimpleRNN)	(None, None, 20)	440
SimpleRNN	(None, 20)	820
Output(Dense)	(None, 1)	21
Total parameters: 1281 Trainable parameters: 1281 Non-trainable parameters: 0		
Table 7: Arch. of RNN		

LSTM & GRU

Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are advanced variations of Recurrent Neural Networks (RNNs) designed to address some of the limitations of traditional RNNs. Although RNNs are less complex in structure which makes them easier to train, require fewer parameters and making them more efficient in terms of computation, both LSTM and GRU networks use gated mechanisms to control the flow of information, which helps in mitigating both the vanishing gradient and short-term memory issues by allowing the model to selectively retain or forget information over time.

LSTM: We developed a very light-weighted sequential LSTM model comprising just three hidden (LSTM) layers with limited nodes. In the final hidden layer, we allocated only five nodes and excluded sequence return, opting instead for the dense layer as the output layer (Output at Final Time Step).



X_t : Input Vector	C_t: Memory from current block	σ : sigmoid function	×: Element-wise multiplication
C_{t-1} : Memory from previous block h_t : output of previous block	h_t : output of current block	tanh : hyperbolic tangent	+ : Element-wise summation/concatenation

Fig. 3 LSTM cell [15][38]. Shows in details architecture of long short term memory

Layer (Type)	Output Shape	Number of parameters
Input(LSTM)	(None, None, 20)	1760
LSTM	(None, None, 10)	1240
LSTM	(None, 5)	320
Output(Dense)	(None, 1)	6

Total parameters: 3326

Trainable parameters: 3326

Non-trainable parameters: 0

Table 8: Arch. of LSTM

Gated Recurrent Unit (GRU): To improve outcomes, we adopted a GRU model featuring three layers of hidden units. In the initial layer, we incorporated a one-dimensional Convolutional layer to compress the data, employing a filter count of five, a kernel size of two, a stride of two, and valid padding. For the second layer, we utilized just five nodes and ensured sequence retention. In the last hidden layer, we employed only three nodes without sequence retention.

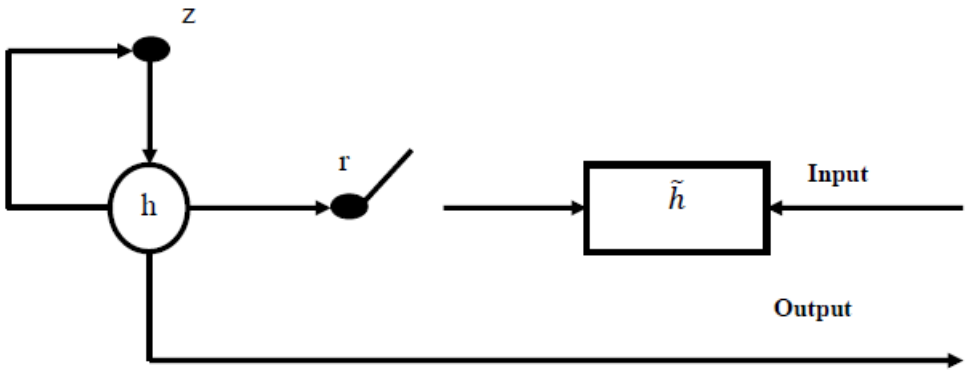


Fig. 4. The architecture of Gated Recurrent Unit (GRU)[38].

Layer (Type)	Output Shape	Number of parameters
Input(Conv1D)	(None, None, 5)	15
GRU	(None, None, 5)	180
GRU	(None, 3)	90
Output(Dense)	(None, 1)	4
Total parameters: 289 Trainable parameters: 289 Non-trainable parameters: 0		

Table 9: Arch. of GRU

Autoencoders(AE): are a type of deep learning algorithm that are designed to receive an input and transform it into a different representation. The encoder transforms the input data into a reduced-dimensional representation, which is often referred to as “latent space” or “encoding”. From that representation, a decoder rebuilds the initial input. We developed stacked AE.

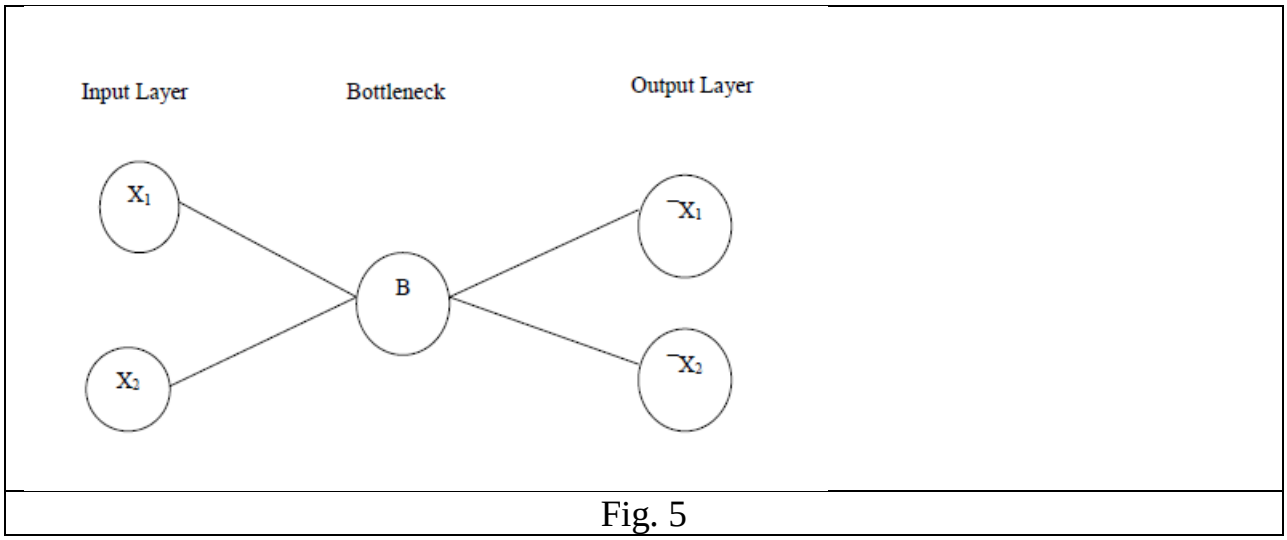


Fig. 5

LSTM-Autoencoder: In our auto-encoding setup, we utilized LSTM layers as the hidden layers due to their effectiveness in sequence learning. During the encoding phase, the model compressed the data from ten dimensions down to three (with three representing the latent representation). In the decoding phase, two hidden layers were employed to restore the dimensions from three back to ten.

Layer (Type)	Output Shape	Number of parameters
Input(LSTM)	(None, None, 10)	
LSTM	(None, None, 5)	
LSTM	(None, None, 3)	
RepeatVector	(None, 1)	
LSTM	(None, None, 5)	800
LSTM	(None, None, 10)	966
Output(Dense)	(None, 1)	

Total parameters: 1,766

Trainable parameters: 1,766

Non-trainable parameters: 0

Table 10: Arch. of Autoencoders(LSTM)

GRU-Autoencoder: Likewise, we employed the GRU layer for autoencoding purposes.

Layer (Type)	Output Shape	Number of parameters
Input(GRU)	(None, None, 10)	
GRU	(None, None, 5)	
GRU	(None, None, 3)	
RepeatVector	(None, 1)	
GRU	(None, None, 5)	800
GRU	(None, None, 10)	966
Output(Dense)	(None, 1)	

Total parameters: 1,766

Trainable parameters: 1,766

Non-trainable parameters: 0

Table 11: Arch. of Autoencoders(GRU)

Bi-directional RNN: A Bi-RNN consists of two separate RNNs: one that processes the sequence forward (from the start to the end) and another that processes it backward (from the end to the start).

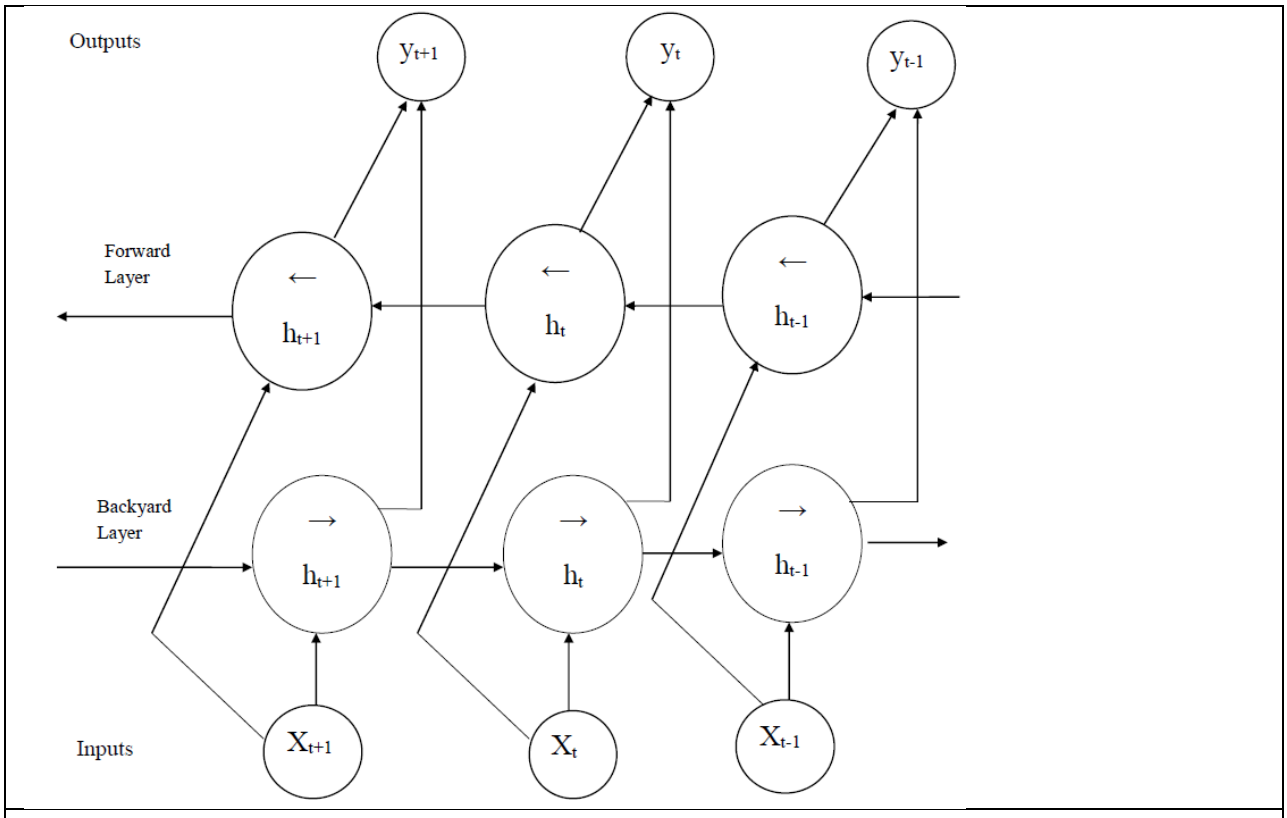


Fig.6

In the initial hidden layer, we utilized a basic RNN layer, while in the subsequent hidden layer, we employed an LSTM layer for bidirectional processing to address the previously mentioned challenges associated with RNNs.

Layer (Type)	Output Shape	Number of parameters
Input(SimpleRNN)	(None, None, 20)	440
Bidirectional(LSTM)	(None, None, 20)	2480
Output(Dense)	(None, 1)	21
Total parameters: 2,941 Trainable parameters: 2,941 Non-trainable parameters: 0		

Table 12: Arch. of Bi-RNN

Machine Learning & Deep Learning: We developed an architecture where a Deep Neural Network (DNN) was used for feature extraction, and a Random

Forest (RF) was employed for binary classification. The DNN was chosen to capture and represent the complexities and non-linearities in the dataset, which enabled the RF classifier to achieve more accurate intrusion detection. Our hypothesis proved correct, as this approach resulted in better accuracy compared to several other models, specifically in reducing both False Positives and False Negatives.

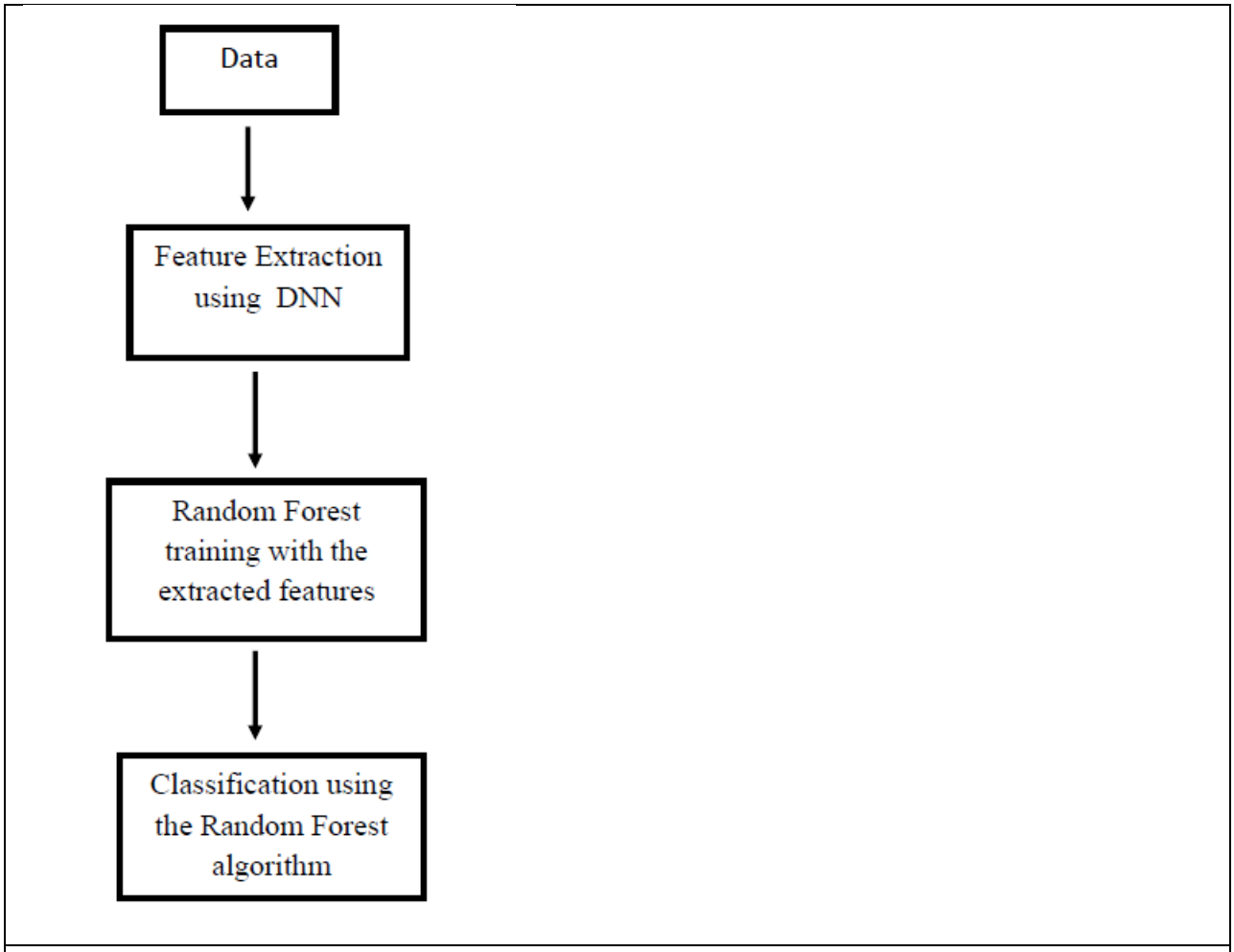


Fig. 7

To implement this, we first divided our dataset into two parts: 75% for training and 25% for testing. The training set was further split into 60% for actual training and 40% for validation. The DNN was trained using the 60% training data and validated with the remaining 40%. The DNN outputted extracted features as ten-dimensional vectors. These features, along with their corresponding labels, were

then used to train the RF classifier. Finally, the classifier was tested on the 25% testing data, leading to superior results.

Layer (Type)	Output Shape	Number of parameters
Input(Dense)	(None, 100)	1500
Dense	(None, 50)	5050
Dense	(None, 40)	2040
Dense	(None, 10)	410
Total parameters: 9000 Trainable parameters: 9000 Non-trainable parameters: 0		
Table 13: Arch. of DNN		

Experiments

Research Environment:

In this study, we used Keras on the backend Tensorflow (Version: 2.10.0), known for its speed, simplicity and popularity in deep learning. The experiment was conducted using Jupyter Notebook (Version: 6.4.12) on a HP Pavilion personal computer equipped with an Intel Core i7-1195G7 CPU @ 2.90GHz and 16 GB of RAM.

Evaluation Methodologies:

1) True Positives (TP): The outcome where the model correctly predicts the positive class.

2) False Positives (FP): The outcome where the model incorrectly predicts the positive class.

3) True Negatives (TN): The outcome where the model correctly predicts the Negative class.

4) False Negatives (FN): The outcome where the model incorrectly predicts the Negative class.

5) Precision: Precision is described as a measure of calculating the correctly identified positives in a model and is given by:

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

6) Recall: It is a measure of actual number of positives that are correctly identified and is given by:

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

7) Since F1-score is a geometric mean of recall and precision and our dataset was a little bit imbalanced, we included this as well.

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Experimental Results:

Model (Thr>0.5)	Accuracy	Label	Precision	Recall	F1	False Positive (%)	False Negative (%)
DNN	94.41	0	1.00	0.89	0.94	10.9	0.2
		1	0.90	1.00	0.95		
LSTM	96.10	0	0.99	0.93	0.96	7.2	0.5
		1	0.93	0.99	0.96		
GRU	96.23	0	1.00	0.93	0.96	7.0	0.5
		1	0.93	1.00	0.96		
LSTM & AE	94.39	0	1.00	0.89	0.94	11.0	**0.0**
		1	0.90	1.00	0.95		
GRU & AE	94.44	0	1.00	0.89	0.94	10.9	0.1
		1	0.90	1.00	0.95		
Bi-Directional RNN	94.47	0	1.00	0.89	0.94	10.8	0.1
		1	0.90	1.00	0.95		
ML & DNN	96.21	0	0.99	0.93	0.96	6.9	0.5
		1	0.93	0.99	0.96		

Table 14: Performance Comparison on IoT-23 testing set

Discussion

Through multiple experiments, we have demonstrated the effectiveness of the proposed models in addressing not only reducing the model's features number but also enhancing its classification detection ability. Despite the notable improvements in detection accuracy attained by the proposed models, there remain certain limitations that warrant attention. In particular, the rate of false positives. Future research endeavors may include the exploration of alternative strategies to enhance the binary classification performance of the proposed models. This may involve investigating diverse model architectures, such as incorporating attention mechanisms or exploring novel loss functions that are better equipped to capture the unique characteristics of the dataset or generate more data to extract more insightful information. Additionally, efforts could be made to improve the interpretability of the model by analyzing the attention weights of the models and identifying significant features for intrusion detection. These approaches may culminate in further improvements in the overall detection performance and can have far-reaching implications beyond the scope of intrusion detection. Overall, this study contributes to the knowledge system by proposing several with novel approaches and demonstrating its effectiveness in addressing those issues in intrusion detection models, while also emphasizing the need for further research and improvement in this area.

Conclusion

The rapid rise in the use of IoT devices has turned them into unsuspecting vectors for cyber-attacks. This paper demonstrates that for certain attacks and IoT devices, deep learning methods such as RNN and BiRNN can effectively classify attacks with high accuracy. While there are numerous datasets available for intrusion detection, it is preferable to use a dataset specifically generated from IoT devices. Thus, this study utilizes the recent IoT-23 dataset. We implemented baseline models, including GRU+AE and DNN+ML. Our findings indicate that RNN-based models are particularly effective (0% False Negative) in identifying and classifying attacks. Additionally, we have shown the effective use of DNN for feature

extraction and ML for classification. Given the critical role of AE, future research could explore anomaly detection further using various types of AE, such as Sparse, Generative AE and Variational AE.

Ref:
DOi link demo : http://dx.doi.org/10.5281/zenodo.4743746.

[1] W. Yang, “Research on network security problems and countermeasures based on the Internet of Things technology,” J. Phys., Conf.Ser., vol. 1744, no. 4, Feb. 2021, Art. no. 042010, doi: 10.1088/1742-6596/1744/4/042010.

[2] S. Fenanir, F. Semchedine, S. Harous, and A. Baadache, “A semisupervised deep auto-encoder based intrusion detection for IoT,” Ingénierie Syst. Inf., vol. 25, no. 5, pp. 569_577, Nov. 2020, doi:10.18280/isi.250503.

[3] C. Vorakulpipat, E. Rattanalerdnusorn, P. Thaenkaew, and H. D. Hai, “Recent challenges, trends, and concerns related to IoT security: An evolutionary study,” in Proc. 20th Int. Conf. Adv. Commun. Technol. (ICACT), Feb. 2018, pp. 405_410, doi: 10.23919/ICACT.2018.8323774.

[4] M. Burhan, R. A. Rehman, B. Khan, and B.-S. Kim, “IoT elements, layered architectures and security issues: A comprehensive survey,” Sensors, vol. 18, no. 9, Sep. 2018, Art. no. 9, doi: 10.3390/s18092796.

[5] I. Andrea, C. Chrysostomou, and G. Hadjichristo_, “Internet of Things: Security vulnerabilities and challenges,” in Proc. IEEE Symp. Comput. Commun. (ISCC), Jul. 2015, pp. 180_187, doi: 10.1109/ISCC.2015.7405513.

[6] K. Angrishi, “Turning Internet of Things (IoT) into internet of vulnerabilities (IoV): IoT botnets,” 2017, arXiv:1702.03681.

[7] R. Ahmad and I. Alsmadi, “Machine learning approaches to IoT security:

A systematic literature review," Internet Things, vol. 14, Jun. 2021, Art. no. 100365, doi: 10.1016/j.iot.2021.100365.

[8] S. Das, P. P. Amritha, and K. Praveen, "Detection and prevention of mirai attack," in Proc. Soft Comput. Signal Process., Singapore, 2021, pp. 79_88.

[9] B. Tushir, H. Sehgal, R. Nair, B. Dezfouli, and Y. Liu, "The impact of DoS attacks on resource-constrained IoT devices: A study on the Mirai attack," 2021, arXiv:2104.09041.

[10] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other Botnets," Computer, vol. 50, no. 7, pp. 80_84, 2017, doi: 10.1109/MC.2017.201.

[11] IoT connected devices worldwide 2019-2030. Accessed: Jun. 17, 2021. [Online]. Available: https://www.statista.com/statistics/1183457/iotconnected-devices-worldwide/

[12] Cisco Annual Internet Report_Cisco Annual Internet Report (2018-2023) White Paper. Accessed: May 31, 2021. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive perspectives/annual-internet-report/white-paper-c11-741490.html

[13] S. Smith. (2020). IoT Connections To Reach 83 Billion By 2024, Driven By Maturing Industrial Use Cases. Accessed: Apr. 10, 2021. [Online]. Available: https://www.juniperresearch.com/press/press-releases/iot-connections-to-reach-83-billion-by-2024-driven

[14] S. Hajiheidari, K. Wakil, M. Badri, and N. J. Navimipour, "Intrusion detection systems in the Internet of Things: A comprehensive investigation," Comput. Netw., vol. 160, pp. 165-191, Sep. 2019, doi: 10.1016/j.comnet.2019.05.014.

[15] K. A. Jallad, M. Aljnidi and M. S. Desouki, "Anomaly detection optimization using big data and deep learning to reduce false-positive," J Big Data, Vol. 7, Aug. 2020, Art. No. 68 (2020), doi: 10.1186/s40537-020-00346-1.

DOi link demo : http://dx.doi.org/10.1186/s40537-020-00346-1.

[16] S. Garcia, A. Parmisano, and M. J. Erquiaga, “IoT-23: A labeled dataset with malicious and benign IoT network traf_c (version 1.0.0),” Zenodo, vol. 20, p. 15, Jan. 2020, doi: 10.5281/zenodo.4743746.

[17] Y. Li, Y. Xu, Z. Liu, H. Hou, Y. Zheng, Y. Xin, Y. Zhao, and L. Cui, "Robust detection for network intrusion of industrial IoT based on multi-CNN fusion," Measurement, vol. 154, Mar. 2020, Art. no. 107450, doi: 10.1016/j.measurement.2019.107450.

[18] M. M. Tavallaee, E. Bagheri, W. Lu and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 2009, pp. 1-6, doi: 10.1109/CISDA.2009.5356528.

[19] Y. Xu, Y. Tang, and Q. Yang, "Deep learning for IoT intrusion detection based on LSTMs-AE," in Proc. 2nd Int. Conf. Artif.Intell. Adv. Manuf., New York, NY, USA, Oct. 2020, pp. 64-68, doi:10.1145/3421766.3421891.

[20] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in Proc. Netw. Distrib. Syst. Secur. Symp., 2018, pp. 1-15.

[21] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Shallow neural network with kernel approximation for prediction problems in highly demanding data networks," Expert Syst. Appl., vol. 124, pp. 196-208, Jun. 2019, doi: 10.1016/j.eswa.2019.01.063.

[22] A. Kim, M. Park and D. H. Lee, "AI-IDS: Application of Deep Learning to Real-Time Web Intrusion Detection," in IEEE Access, vol. 8, pp. 70245-70261, 2020, doi: 10.1109/ACCESS.2020.2986882.

[23] B. Susilo and R. F. Sari, "Intrusion detection in IoT networks using deep learning algorithm," Information, vol. 11, no. 5, p. 279, May 2020, doi:10.3390/info11050279.

[24] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset," Future Gener. Comput. Syst., Vol. 100, pp.779–796, Nov. 2019. doi: 10.1016/j.future.2019.05.041.

[25] C. Yin, Y. Zhu, J. Fei and X. He, "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks," in IEEE Access, vol. 5, pp. 21954-21961, 2017, doi: 10.1109/ACCESS.2017.2762418.

[26] Z. Li, A. L. G. Rios, G. Xu and L. Trajković, "Machine Learning Techniques for Classifying Network Anomalies and Intrusions," 2019 IEEE International Symposium on Circuits and Systems (ISCAS), Sapporo, Japan, 2019, pp. 1-5, doi: 10.1109/ISCAS.2019.8702583.

[27] Y. Imrana, Y. Xiang, L. Ali, and Z. Abdul-Rauf, "A bidirectional LSTM deep learning approach for intrusion detection," Expert Syst. Appl., vol. 185, Dec. 2021, Art. no. 115524, doi: 10.1016/j.eswa.2021.115524.

[28] Y. Yang, K. Zheng, C. Wu, X. Niu, and Y. Yang, "Building an effective intrusion detection system using the modified density peak clustering algorithm and deep belief networks," Appl. Sci., vol. 9, no. 2, p. 238, Jan. 2019, doi: 10.3390/app9020238.

DOi link demo : http://dx.doi.org/ 10.1109/ACCESS.2020.3048198.

[29] K. Wu, Z. Chen and W. Li, "A Novel Intrusion Detection Model for a Massive Network Using Convolutional Neural Networks," in IEEE Access, vol. 6, pp. 50850-50859, 2018, doi: 10.1109/ACCESS.2018.2868993.

[30] S. Naseer et al., "Enhanced Network Anomaly Detection Based on Deep Neural Networks," in IEEE Access, vol. 6, pp. 48231-48246, 2018, doi: 10.1109/ACCESS.2018.2863036.

[31] Y. Ding and Y. Zhai, "Intrusion detection system for NSL-KDD dataset using convolutional neural networks," in Proc. 2nd Int. Conf. Comput. Sci. Artif. Intell. (CSAI), 2018, pp. 81-85, doi: 10.1145/3297156.3297230.

[32] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat and S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System," in IEEE Access, vol. 7, pp. 41525-41550, 2019, doi: 10.1109/ACCESS.2019.2895334.

[33] N. Chouhan, A. Khan, and H.-U.-R. Khan, "Network anomaly detection using channel boosted and residual learning based deep convolutional neural network," Appl. Soft Comput., vol. 83, Oct. 2019, Art. no. 105612, doi: 10.1016/j.asoc.2019.105612.

[34] A. N. Sokolov, S. K. Alabugin and I. A. Pyatnitsky, "Traffic Modeling by Recurrent Neural Networks for Intrusion Detection in Industrial Control Systems," 2019 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), Sochi, Russia, 2019, pp. 1-5, doi: 10.1109/ICIEAM.2019.8742961.

[35] T. H. Morris, Z. Thornton, and I. Turnipseed, “Industrial control system simulation and data logging for intrusion detection system research,” 7th Annual Southeastern Cyber Security Summit, 2015.

[36] T. H. Morris and W. Gao, “Industrial control system cyber attacks,” Proc. first International Symposium for ICS & SCADA Cyber Security Research, pp. 22–29, 2013.

[37] L. Liu, P. Wang, J. Lin and L. Liu, "Intrusion Detection of Imbalanced Network Traffic Based on Machine Learning and Deep Learning," in IEEE Access, vol. 9, pp. 7550-7563, 2021, doi: 10.1109/ACCESS.2020.3048198.

[38]: Zahra Ebrahimi, Mohammad Loni, Masoud Daneshtalab, Arash Gharehbaghi, "A review on deep learning methods for ECG arrhythmia classification," Expert Systems with Applications: X, Vol. 7, September 2020, Art. no. 100033, doi: 10.1016/j.eswax.2020.100033.