DETECTION OF ICMPV6 DDOS ATTACKS USING HYBRIDIZATION OF RNN AND GRU

Om Vasu Prakash Salmakayala, Saeed Shiry Ghidary
and Christopher Howard

School of Digital, Technology, Innovation and Business
at Staffordshire University, Stoke on Trent,
Staffordshire-ST4 2DE, United Kingdom

ABSTRACT

The internet, crucial for information exchange, operates on IPv6 and IPv4 protocols, which are vulnerable to DDoS attacks. Despite secure-edge advancements, these attacks still cause significant losses. This paper presents a Deep Neural Network (DNN) architecture to address these vulnerabilities. Model 1 integrates Recurrent Neural Networks (RNN) with Gated Recurrent Units (GRU), inspired by Ahmed Issa, while Model 2 employs Convolutional Neural Networks (CNN) with Long Short-Term Memory (LSTM). These models were tested on Mendeley, NSL-KDD, and Sain Malaysian datasets, achieving accuracies of 80%, 80% 97.01%, 95.06%, 72.89%, and 64.94%, respectively. The objective is to verify the practical feasibility of these combinations to detect DDoS-attacks. The same architecture was implemented in Model 1 for further evaluation using NSL-KDD as used by Issa, Mendeley IPv4, and Sain Malaysian datasets. A new ICMPv6 datasets were deployed with different architecture layers on the proposed model resulting in promising accuracies of 99.36% and 94.48%.

KEYWORDS

DDoS attacks, IPv4, ICMPv6, IPv6, Deep Neural Networks (CNN, LSTM, RNN, GRU).

1. INTRODUCTION

With the proliferation of cutting-edge technologies in computing domains like Cloud Computing and the Internet of Things, the incidence of Distributed Denial of Service (DDoS) attacks has surged significantly. This escalating frequency poses a substantial threat, rendering DDoS attacks among the most formidable challenges in the realm of cybersecurity [1]. This technology also opens up extensive avenues for various network attacks, specifically targeting critical services and causing system malfunctions in servers affecting the entire enterprise networks. Such disruptions lead to business paralysis, manifesting as downtime and resulting in significant financial losses. A denial of service (DoS) can be briefed as an attack inundates a server with traffic, rendering a website or resource inaccessible. In the case of a distributed DDoS attack, multiple computers or machines collaborate to flood a specific target with overwhelming traffic, exacerbating the impact of the attack [2]. Attackers meticulously investigate unprotected entry points, such as vulnerabilities in software or system configurations, and skillfully exploit them. Leveraging these entry points, they attempt to compromise the system by depleting its resources, thereby denying access to legitimate users. In alternative attack scenarios, malicious bots are deployed to inundate the target system with an overwhelming number of packets, ultimately leading to a server crash [3].

David C. Wyld et al. (Eds): DMSE, CSEIT, NeTCoM, SPM, CIoT, NCS, NLPD, ArIT, CMLA – 2024
pp. 297-317, 2024. CS & IT - CSCP 2024 DOI: 10.5121/csit.2024.141423

1.1. Basic Attack Scenario of DDoS using ICMP/ICMPv6 Protocol

There are different ways to launch DDoS attacks, Figure 1 provides one of them that outlines the steps involved in a threat attacker's initiation of a DDoS attack. The attacker begins by exploring methods like phishing to infiltrate a system and install malware. Once control of a compromised machine is secured, the attacker distributes bots to other systems through lateral movement. With control over these systems, the attacker deploys command and control, using scripts to command Power Shell to unleash a flood of ICMP packets at the target server. This flood overwhelms the server's resources, causing Distributed Denial of Service (DDoS) critical conditions, and preventing legitimate users from establishing connections due to resource depletion. [4]

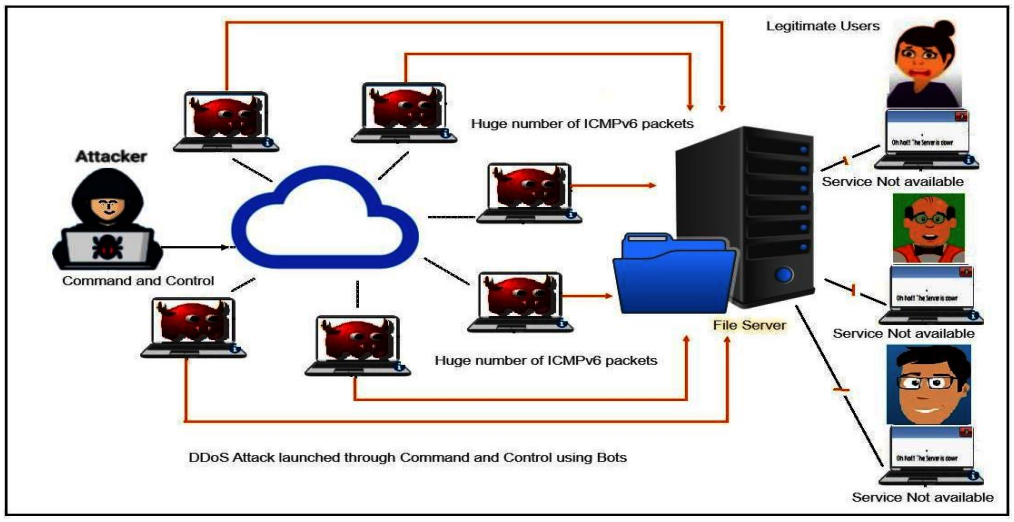

Fig.1. DDoS attack -Implementation of DDoS attack

To avoid these ICMP attacks then one can argue to disable the ICMP packets. However, disabling ICMP will have adverse effects on network functionality and diagnostics where some of them are listed below:

1. Troubleshooting: Disabling ICMP makes it harder to diagnose network issues. ICMP error messages provide valuable information about why a particular communication failed.
2. Ping and Traceroute: Tools like ping and traceroute heavily rely on ICMP. Disabling ICMP will prevent accurate measurement of network latency and route tracing.
3. MTU Issues: Path MTU discovery won't function correctly if ICMP is disabled, potentially leading to fragmentation issues and degraded performance.
4. Reachability: ICMP destination unreachable messages won't be sent to find out whether communication between nodes or systems can communicate and also to test any issues in establishing a connection[5].

This research paper mainly focuses on ICMPv6 DDoS attacks and to understand the ICMPv6 header fields for Echo Request and Echo Reply messages, which are used in DDoS attacks besides TCP flood, UDP flood, SYN flood, etc.[33]., Figure 2 shows the packet structures. ICMPv6 header contains information about the source and destination addresses, among other things like type, code, checksum, identifier, sequence number, and data specific to the ICMPv6 message type.

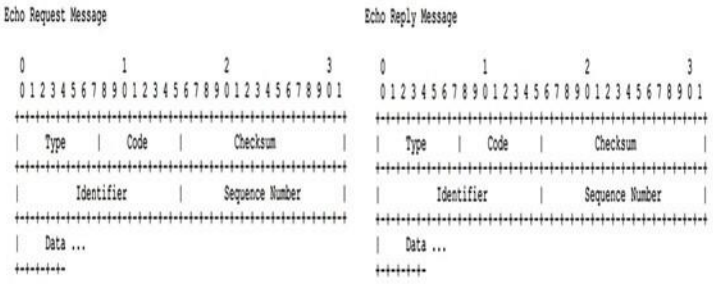

Fig.2. ICMPV6 Echo-reply header [22]

1. Type: Differentiates between Echo Request (128) and Echo Reply (129).
2. Code: Always 0 for these messages.
3. Checksum: Includes the ICMPv6 message and parts of the IPv6 header, ensuring data integrity.
4. Identifier: Helps in pairing Echo Requests with their corresponding Echo Replies.
5. Sequence Number: Allows tracking of individual Echo Request messages.
6. Data: The payload data which is echoed back in the Echo Reply [22].

Liu has provided the type code parameters in the statistical form as per the DDoS attack that is illustrated in Figure 3.

Initial Stage	ICMPv6 Type	Message Name	Attack Type
Source Address Spoofing	2	Packet Too Big	Reflection amplification
	4	Parameter Problem	Reflection amplification
	128	Echo Request	Reflection amplification
	130	General Query	Reflection amplification
	132	Multicast Listener Done	Protocol exploit
	134	Router Advertisement	Protocol exploit
	136	Neighbor Advertisement	Protocol exploit
Pure Flooding	128	Echo Request	Flooding
	129	Echo Reply	Flooding
	133	Router Solicitation	Flooding
	134	Router Advertisement	Flooding
	135	Neighbor Solicitation	Flooding
	136	Neighbor Advertisement	Flooding

Fig.3. DDoS ICMPv6 attack Type codes [23]

Despite various intelligent intrusion detection systems (IDS) in place, they are still falling short of detecting such attacks due to the smart approach of the threat attackers when the packets are jumbled with Flag values that don't have sequential values [6]. The threat actor employs stealthy intrusion tactics to remain undetected. Initially, they focus on crafting custom attack methods devoid of recognizable markers or leveraging vulnerabilities in network protocols and the target system[7]. For instance, they may manipulate out-of-order fragments to deceive scanning systems, ensuring their malicious activities go unnoticed and enabling successful compromise of the target server. In traditional detection methods primarily rely on signature/hash anomaly, time interval analysis, data mining, and packet flow examination. The Hybrid approach integrates combinations of these classifications for comprehensive detection either by using multiple ML techniques or DNN techniques or both. [8]. In addressing these challenges, numerous researchers have proposed diverse solutions, leading to promising outcomes. Among these, researcher Ahamad Issa introduced an intriguing approach, hybrid solutions that integrate CNN layers for automated input feature extraction and LSTM for sequence prediction[9]. This research proposes a similar model with a combination of a Recurrent neural network and a Gated recurrent unit that is used to process sequential data by maintaining a hidden state that captures information about previous inputs. This technique should determine the continuous flow of packets based on the

features trained and detect the DDoS attack avoiding the Gradient decent constraint by GRU that incorporates gating mechanisms that control the flow of information within the network[26]. The experiments are compared and evaluated with Ahamad Issa's model results.

1.2.Related Work

With newly developed DDoS strategies, threat attackers aim to outperform typical DDoS countermeasures due to a lack of performance ability, scalability, complexity, and adaptability in large networks [10].

(a) Ahmed Issa introduced an innovative deep learning classification approach by combining two widely used algorithms, CNN and LSTM. The model was designed with 7 layers of Deep neural learning network consisting of CNN 1D with Kernel, strides, in the CNN layer, Maxpooling 1D, with activation function Relu and Softmax for the output connected layers. His model was evaluated using the NSLKDD dataset that consists of 40 features with multiple attacks achieving an impressive accuracy rate of 99.20%. Figure 4 illustrates the Issa's model architecture[9].

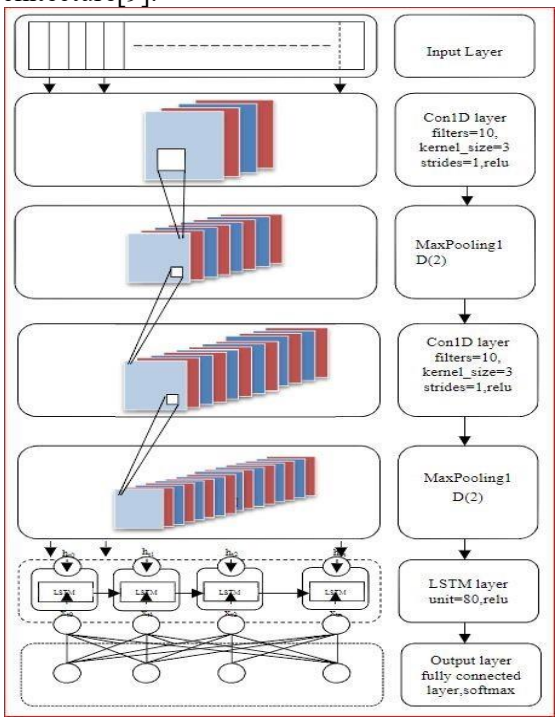

Fig.4. Ahmed Issa Model Architecture (CNN with LSTM) [9]

(b) Omar Elejla introduced an innovative method for the detection of ICMPv6 flooding DDoS attacks in IPv6 networks. This approach leverages deep learning and incorporates an ensemble feature selection technique, which utilizes chi-square and information gain ratio methods to identify crucial features for accurate attack detection. The model employs an LSTM network to train on the selected features, resulting in impressive detection accuracy rates: 87.1% for RNN, 99.4% for LSTM, and 99.11% for GRU [11].

(c) Saif conducted a comprehensive comparative analysis between GRU and. They employed traditional machine learning classifiers such as Naive Bayes (NB) and Sequential Minimal Optimization (SMO) in their evaluation. The study utilized the CICDDoS2019 dataset, where feature extraction was performed on numerical types across 13 categories that represented various DDoS attacks based on protocols including NTP, UDP, DNS, LDAP,

NetBIOS, SNMP, SYN, Web-DDoS, TFTP, etc. They achieved impressive results with an accuracy of 99.91% with the GRU model, surpassing the performance of the RNN model in the comparison [12].

(d) Kumar conducted a comprehensive comparative analysis of different Deep Learning techniques including LSTM, Bidirectional LSTM, Stacked LSTM, and GRU. The study utilized the CICDDoS2019 datasets, with pre-processing involving standardization for numerical values and label encoding for class values. The dataset was divided into 80% for training and 20% for evaluation. Notably, the stacked LSTM technique outperformed others, achieving an outstanding result of an accuracy rate of 99.55% [13].

(e) Tian developed a DDoS detection classification algorithm tailored for low-traffic networks, leveraging federated learning. His approach incorporated LwResnet-FL and DCNN LSTM, achieving an impressive accuracy of 99.20%. To validate the model, he compared it with Random Forest, AE-DNN, and RNN, ensuring robust performance across various evaluation metrics [14].

(f) Ullah proposed a comprehensive intrusion detection system leveraging binary and multiclass classification CNNs across 1D, 2D, and 3D data. The model was designed to detect intrusions in Internet of Things (IoT) environments. His model was validated using diverse datasets, including intrusion detection datasets. The results demonstrated impressive performance, with the CNN 1D model achieving an accuracy of 99.74%, the CNN 2D model achieving 99.42%, and the CNN 3D model achieving 99.03% [15].

1.3. DNN Techniques

Deep learning is a subset of machine learning, characterized by several notable distinctions. It requires extensive datasets for robust data. Additionally, deep learning automates the extraction of features from data. Unlike traditional machine learning approaches that often require problem decomposition into subproblems, deep learning directly addresses the main task at hand. In this study, two deep learning techniques, RNN and GRU are integrated to devise an innovative combination[10].

1.3.1. Recurrent Neural Network

RNNs encompass two key architectures:- feedforward and bidirectional propagation. RNNs process data inputs iteratively, with outputs relying on past computations. In feed-forward RNNs, learning progresses sequentially, with each output feeding into the subsequent hidden layer node, retaining essential information for future tasks[26]. Conversely, bidirectional RNNs employ two hidden layers operating in opposing directions, accessing both preceding and succeeding states, thus enriching hidden layers with contextual information. RNNs find extensive utility across various domains like image processing and speech analysis, excelling in sequential data handling due to their recurrent architecture. Computational processes involve input vectors guided by equations, activation functions, and adjustments to weights via backpropagation. Despite their strengths, RNNs face challenges like the vanishing gradient problem, stemming from diminishing weight updates over time [18].

1.3.2. Merits of RNN

Due to their ability to process one input at a time, RNNs are suitable for real-time applications like online object recognition that can be applied to identify the attributes of malicious packets. RNNs can process input sequences of varying lengths, making them versatile for different types of sequential data, such as threat attackers trying to change the packet length text and packet series with missing or obfuscation values. RNNs can be used for various tasks related to the flow of packet data, network traffic, protocol identification, packet analysis, and more.

However, conventional RNNs have weaknesses in capturing extended dependencies because of the vanishing gradient problem, which can make them struggle with retaining information from distant past time steps. To address these limitations, more advanced RNN architectures have been developed, such as LSTM networks and Gated Recurrent Units GRUs. These architectures incorporate gating mechanisms that enable the network to control the flow of information and gradients, making them better suited for capturing long-term dependencies [24].

1.3.3. Gated Recurrent Unit

It is a formidable asset in handling sequential data tasks like language modelling, speech recognition, and time series prediction. At its core, it maintains a hidden state vector that evolves with each time step, influenced by both the present input and the preceding state. Key to its functionality is two gating mechanisms: the reset gate, which determines what to discard from the prior state, and the update gate, regulating the infusion of fresh state information. These gates strike a delicate equilibrium between assimilating new data and retaining pertinent historical context, effectively addressing the challenge of capturing distant dependencies while mitigating the vanishing gradient predicament[19].

1.3.4. Merits of GRU

It has a simpler structure compared to LSTMs, with fewer gates (reset and update gates) and no separate memory cell. This makes them easier to understand and implement. Due to the simplified architecture, GRUs have fewer parameters than LSTMs. This can lead to faster training times and reduced computational resource requirements. GRUs are generally faster to train compared to LSTMs because of their reduced complexity. This efficiency can be crucial when working with large datasets or when computational resources are limited. Above all their simplified architecture, fewer parameters, and effectiveness in handling dependencies contribute to their popularity and utility in various applications[25].

2. PROPOSED MODEL

The proposed model represents a hybrid approach merging RNN and GRU components into a unified architecture comprising nine layers which is illustrated in Figure 5. This integration aims to enhance the efficacy of DDoS attack detection. This architecture fusion is detailed further in the subsequent paragraph.

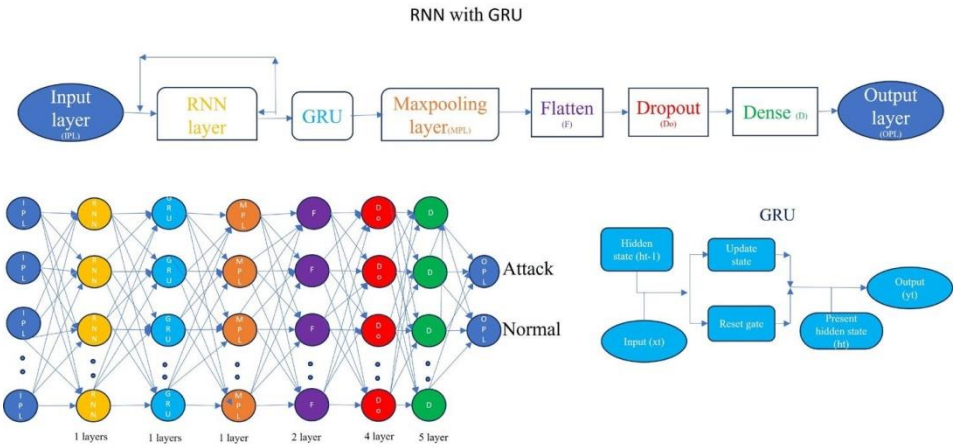

Fig.5. Model Architecture

Normally in each model Input Layer (IPL) and output layer (OPL) are present. In between there exist hidden layers of the model designed by arranging them depending upon the chosen algorithms and respective activation function Relu at the required layer. The 14 parameters are the network attributes used to improve the output of the model. At any given time t, the current input is a combination of input at IPL(t) and IPL(t-1). The output at any given time is fetched back to the network to improve on the output.

The model architecture starts with an initial input layer consisting of an RNN with a kernel filter size of 128, utilizing the Rectified Linear Unit (ReLU) activation function. Following this, a GRU layer is employed with 64 filters, maintaining sequential order. The third layer comprises a MaxPooling layer and a one-dimensional convolutional layer with a pooling size of 1. Subsequently, a Flatten layer reshapes the tensor into a vector, facilitating a seamless transition between connected layers, particularly interfacing with the dense layer. Dropout regularization is then applied, randomly deactivating 10% of neurons in a layer during training to mitigate overfitting.

The final layer is a dense layer designed for classification tasks. The output of this dense layer typically undergoes a Leaky ReLU activation function, generating probabilities for each class, such as ”Normal” and ”Attack.”

2.1. Model Learning

During the training phase, the weights were adjusted using the back-propagation technique. This process utilized the Sparse Categorical Cross-entropy loss function to calculate error loss, which was then propagated backward across the network. All intermediate nodes between layers were interconnected, contributing their error values to the forward propagation. The entire network was enveloped by both forward and backward propagation mechanisms. For weight updating, the stochastic gradient descent optimizer for Adaptive Moment Estimation (ADAM) [20] was employed with a learning rate of 0.01, and parameter tuning set a minimum delta of 0.000001. To ensure effective training, the networks underwent 5 epochs, where each epoch involved one pass forward and backward of all data in the training set or a comprehensive training cycle with a batch size of 5000. This iterative process enabled the network to gradually refine its weights and improve its performance over each epoch.

Model checkpoints were applied which are snapshots of the proposed model weights and parameters saved at specific points during training. The parameters used are from the Keras methods where the parameters are assigned are given below:

- **save_best_only=True**: Saves only the model with the best performance based on the monitored metric. This parameter saves storage space and avoids saving models that do not represent the best performance during training. The "best" model is defined as the one with the optimal metric value (e.g., lowest validation loss).
- **save_weights_only=True**: Saves only the model weights, not the entire model (architecture and optimizer state). This parameter saves space and time by storing only the weights, which can later be loaded into a model with the same architecture for further use or evaluation.
- **monitor='val_loss'**: Monitors validation loss to determine the best model. This parameter determines which metric to track to decide when to save the model. This parameter saves the occurrence when this metric shows the best performance of the lowest validation loss
- **mode='min'**: Saves the model when the validation loss is minimized. This parameter indicates that the model will be saved when the validation loss decreases and reaches a new minimum value.

- **verbose=1**: Provides basic information about when the model is saved. This parameter provides basic feedback during training about when the model checkpoints are being saved. [29][30]

Early stopping is a regularization technique used to prevent overfitting and improve model generalization by halting training when the performance on a validation dataset starts to degrade. The parameters applied are:

- **monitor** = **val_loss**: Tracks the validation loss of val_loss. This parameter provides the training process evaluated based on the model performance on the validation dataset with respect to metric i.e., accuracy in our case.
- **min_delta** = 0.000001: Requires a minimum change of 0.000001 to consider the validation loss as improved. This parameter determines whether the change in the validation loss is significant enough to be considered for an improvement. Changes smaller than 0.000001 will not be treated for improvement.
- **patience = 3**: Continues training for up to 3 more epochs without significant improvement. This parameter allows the model to continue training for a few more epochs after the last improvement, accounting for possible fluctuations or noise in the validation metric.
- **mode = min**: Aim to minimize the validation loss. This parameter provides about the training to stop if the validation loss does not decrease by at least min_delta for the number of epochs specified by patience
- **verbose = 2**: Provides detailed output. In practice, common verbosity levels are 0, 1, or 2. While verbose=>2 might not be standard, it typically means that lot of detailed information will be printed out during training.
- **restore_best_weights**: Ensures that the model's weights are set to the bestperforming epoch based on validation loss. This means when early stopping is triggered, the model's weights are set to those from the epoch with the best-observed performance on the validation set. This helps in achieving the best possible model state based on validation data. [28][31]

Reduce- Learning Rate On Plateau is to fine-tune the model's training process and the parameters applied are:

- **monitor = 'val_loss'**: Observes the validation loss to Adjust the learning rate based on changes in the validation loss
- **factor=0.1**: Reduces the learning rate to 10% of its current value. This parameter helps in reducing the learning rate when the monitored metric has stopped improving.
- **patience = 3**: Waits for 3 epochs without significant improvement before reducing the learning rate. This parameter determines how many epochs to continue training without reducing the learning rate if no significant improvement is observed.
- **min_delta = 0.000001**: Considers only changes larger than this value as improvements. This parameter helps to avoid unnecessary learning rate adjustments due to very small changes that might not be meaningful
- **mode ='min'**: Seeks to minimize the validation loss. This parameter ensures that the learning rate is adjusted when the validation loss does not decrease significantly.
- **verbose=1**: Provides feedback on learning rate changes during training about when and how the learning rate is changed. [27][32]

3. METHODOLOGY

Figure 6 illustrates the block diagram of an implementation of the method used for the proposed model. In section 1.3 RNN, GRU merits were briefed, and subsequent sections provide the details of the dataset, implement pre-processing techniques, present RNN-GRU model architecture, and conclude with an overview evaluation of the proposed model by comparing it with the results of the Ahmed Isaa Model (CNN-LSTM).

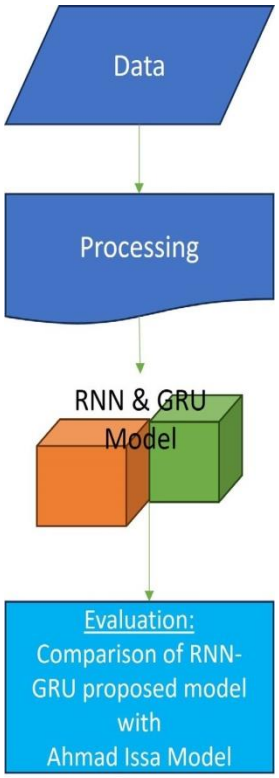

Fig.6. Methodology flowchart

3.1.1. Sain Malaysian Dataset

Omar Elejila meticulously crafted ICMPv6 datasets, tailored to the network topology forming the cornerstone of his research endeavours. The data synthesis spanned a rigorous 2-hour window, meticulously capturing network activity, culminating in a dataset measuring 15.8 MB. A distilled subset of this traffic, condensed into a 2.8 MB Excel sheet, has already been curated to serve as training and testing datasets that contain 11 features depicted in Figure 7[11].

S. no.	Feature	Description	Calculation
1	*ICMPv6Type*	The ICMPv6 type of the sent packet	Originally in the flow as a key
2	*PacketsNumber*	Number of sent packets within the flow	By counting the packets
3	*TransferredBytes*	Number of bytes sent from the source to the destination	Summation of packets length
4	*Duration*	Time length of the flow	Last packet time—first packet time
5	*Ratio*	Ratio of bytes transferring during the flow duration	Transferred bytes/duration
6	*Length_SD*	The variation in the Lengths of flow's packets	Standard deviation of the packets Lengths
7	*FlowLabel_SD*	The variation in the Flow labels of flow's packets	Standard deviation of the packets Flow labels
8	*HopLimit_SD*	The variation in the Hop limits of flow's packets	Standard deviation of the packets Hop limits
9	*TraffiicClass_SD*	The variation in the Traffic classes of flow's packets	Standard deviation of the packets Traffic classes
10	*NextHeader_SD*	The variation in the Next headers of flow's packets	Standard deviation of the packets Next headers
11	*PayloadLength_SD*	The variation in the Payload lengths of flow's packets	Standard deviation of the packets Payload lengths

Fig.7. Sain Malaysian Dataset features

3.1.2. Mendeley Dataset

The datasets, curated by Housman from Universitas Muhammadiyah Malang, contain IPv4 data used for research purposes. They focus on Distributed Denial of Service (DDoS) attacks within Software-Defined Networking (SDN), including ICMP, TCP, and UDP flood incidents. These attacks were simulated using the Mininet Emulator with Scapy, resulting in a 34.7 MB .pcap file capturing the traffic. The RYU controller was augmented to store attack information of up to 37.9 MB. Comprising 25 features, these datasets are illustrated in Figure 8, detailing the features list of Mendeley datasets [16]

Fig.8. Mendeley dataset features

3.1.3. NSL-KDD (Benchmark) Datasets

Mahbod Tavallaee describes the KDD dataset as a compilation of data from the DARPA'98 Intrusion Detection System (IDS) evaluation program, totalling around 4 gigabytes of compressed raw TCP dump data spanning 7 weeks of network traffic. This dataset includes approximately 4,900,000 single connection vectors, each comprising 41 features and labelled as normal or an attack, with one specific attack type. The NSLKDD dataset, an extension of the original, incorporates multiple attack types and was utilized for the Issa Ahmed Model and Proposed Model. Further details on the features are provided in Figure 9 [17].

#	Column
0	duration
1	protocol_type
2	service
3	flag
4	src_bytes
5	dst_bytes
6	land
7	wrong_fragment
8	urgent
9	hot
10	num_failed_logins
11	logged_in
12	num_compromised
13	root_shell
14	su_attempted
15	num_root
16	num_file_creations
17	num_shells
18	num_access_files
19	num_outbound_cmds
20	is_host_login
21	is_guest_login
22	count
23	srv_count
24	serror_rate
25	srv_serror_rate
26	rerror_rate
27	srv_rerror_rate
28	same_srv_rate
29	diff_srv_rate
30	srv_diff_host_rate
31	dst_host_count
32	dst_host_srv_count
33	dst_host_same_srv_rate
34	dst_host_diff_srv_rate
35	dst_host_same_src_port_rate
36	dst_host_srv_diff_host_rate
37	dst_host_serror_rate
38	dst_host_srv_serror_rate
39	dst_host_rerror_rate
40	dst_host_srv_rerror_rate
41	label

Fig.9. NSL-KDD dataset features

3.1.4. LTVM – Datasets

Virtual Dataset was generated on a single Laptop machine with an Intel i7 11th generation 2.30 GHz processor, 64 GB RAM, and a 2 TB hard disk. Within a VMware environment, four virtual machines were set up: one Linux machine (LVPC), two Windows machines (WVPC-1 and WVPC-2), and one Windows server (WVS). These virtual machines were interconnected through VMware network adapters on a single NIC card of the host machine. An ICMPv6 DDoS attack was launched using a Scapy script from two virtual machines (LVPC and WVPC-1) targeting the Windows server (WVS) to simulate an attack scenario. Network traffic, including normal activity and the ICMPv6 attack, was captured on the Windows server (WVS) using Wireshark, resulting in a dataset size of 18.3 MB (60,000 bytes/sec). The captured traffic was then transformed into an Excel sheet named scdtsets.csv, with a size of 58.2 MB. These datasets were utilized for the proposed Model analysis and the features are depicted in Figure 10.

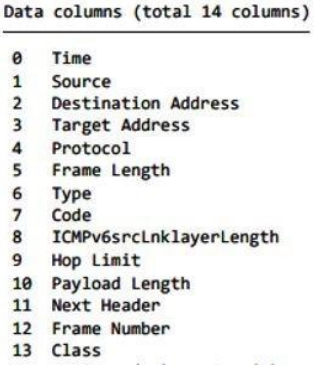

Fig.10. LTVM dataset features

3.2. Pre-Processing

1. The final processed datasets that are treated as data points indicating rows and the features indicated by columns are fed to the input layer of the Model. These data points are then processed in the hidden layers based on the feature extracted. The output layer produces the result to identify if it's an Attack that is assigned as 1 or Normal assigned as 0 based on the normalization done in prepossessing.
2. The ICMPv6 datasets from Sain Malaysian University have been pre-processed. The first dataset includes 11 features, with a designated "Class" column categorizing data into Normal and Attack. It contains 92,944 rows and 12 columns, resulting in a matrix dimension of 92,944 x 12.
3. The NSL-KDD datasets consist of 42 features, with an additional "Label" column for categorizing data into Normal, as well as four distinct types of attacks: DoS, Probe, R2L, and U2R. This dataset comprises 125,973 rows and 43 columns, resulting in a matrix dimension of 125,973 rows × 43 columns. The Mendeley datasets comprise 24 features, accompanied by an additional column named as "Label" for categorizing data into DDOS-ICMP and NORMALICMP. It includes 140,000 rows and 25 columns, resulting in a matrix dimension of 140,000 rows × 25 columns.
4. The Primary Dataset-1, upon extraction into the "scdtsets.csv" file, demonstrated satisfactory quality. It includes 14 features, with an additional column named "Class" introduced to categorize data into two classes: Normal and Attack. This dataset encompasses 100,130 rows and 14 columns, resulting in a matrix dimension of 100,130 rows x 14 columns.

Figure 11 illustrates the block diagram of the preprocessing of the above data sets and fed to the proposed model.

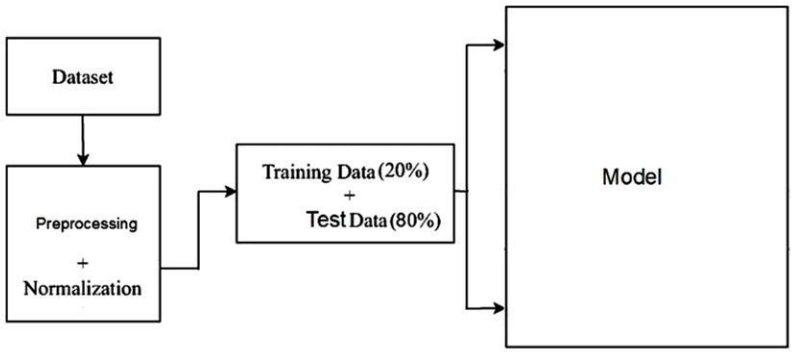

Fig.11. Pre-processing

a. Cleaning: In case of any noise characters like commas, dashes, blank spaces, or any other special characters, they need to be removed.
b. Data Transformation: This step involves transforming the data into a more suitable format for modelling. This may include scaling features to have similar ranges (Min-Max or standardization scaling), transforming classified variables into numerical values (one-hot encoding or label encoding), or transforming numerical variables to make their distribution more Gaussianlike (e.g., log transformation).
c. Feature Selection/Extraction: In some cases, not all features in the dataset are relevant to the model or contribute to its performance. Feature selection techniques like filtering methods or wrapper methods are used to opt for the most relevant attributes. In our experiments, the attributes are manually selected mainly focusing on the ICMPv6 header fields.
d. Data Normalization: Normalizing the data ensures that different features are on a similar scale, which can help gradient-based optimization algorithms converge faster. Normalization techniques include scaling features to have zero as mean and one as standard deviation or scaling features to a fixed range (e.g., [0, 1]).
e. Data Splitting: Finally, the dataset is split into training and test sets. The training set is used to train the model, tune hyper-parameters and evaluate model performance during training, and the test set is used to evaluate the final model's performance. The dataset was divided into 80% for training and 20% for testing.
f. Imbalanced Data: It is a situation where the classes in a dataset are not represented equally. To overcome this there are techniques like SMOTE, ADASYN, etc. to apply where they add synthetic samples nearly equal to the major category of the datasets. However, at this stage, the results seem to be good enough even though the generated datasets are imbalanced. In the later stages, this technique is also applied to test the difference and finally select the highest score.

4. EXPERIMENTS RESULTS

The requisite experiments were conducted in Google Colab using Python, leveraging its versatile libraries such as NumPy, Scikit Learn, Keras, TensorFlow, among others, in crafting the proposed model.

4.1. Evaluation Criteria

Various performance metrics are employed to assess the accuracy of the Model, and their effectiveness relies on diverse parameters. Achieving optimal results confirms that the classifier is well-suited for its intended purpose. Key factors influencing performance include the selection of datasets, features, algorithm/technique type, formula choice, and training time. The outcomes are reflected in metrics such as True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). These metrics quantify the classifier's output, with a higher count of True Positives signifying a better-fitting model and resulting in increased accuracy. [21]

Accuracy can be defined as the ratio of the number of correct instances to that of total number of instances.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

However, this paper being a part of the research project these experiment results were mainly focused on the performance metric "Accuracy". While progressing in the research other metrics like Recall, Precision, F1 measure, etc will be used and their corresponding results will be included at later stage.

4.2. Findings

Four distinct datasets outlined in the Datasets section were utilized for model development and deployment. Upon deployment, the CNN-LSTM model exhibited promising outcomes, achieving accuracies of 80%, 97.01%, and 72.89% on the NSLKKD, Mendeley, and Sain Malaysian datasets respectively. Similarly, the RNNGRU Model yielded satisfactory results, attaining accuracies of 80%, 95.06%, and 64.95% on the NSL-KKD, Mendeley, and Sain Malaysian datasets correspondingly.

Figure 12 illustrates the statistics and comparison of both models with the mentioned datasets.

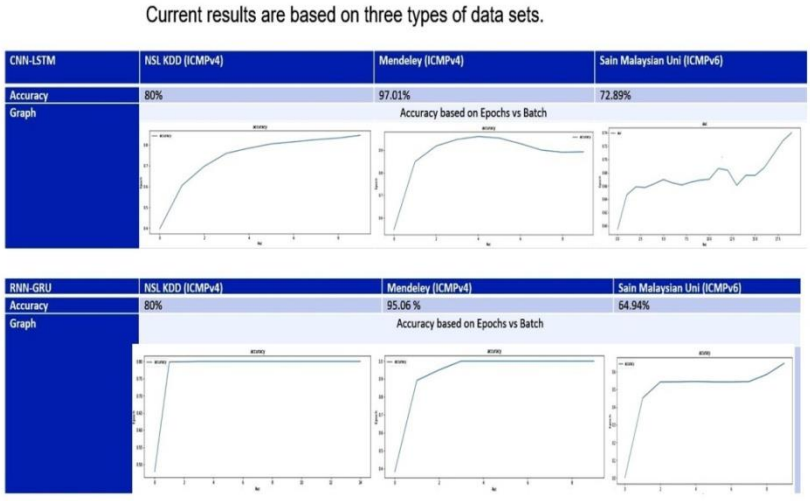

Fig.12. Statistical comparison of results related to 3 different data sets

Keeping these statistics in view the results varied significantly. This variability in results showed that the design of the models was robust enough to detect ICMPv6 datasets effectively and for evaluation newly generated datasets were used as a sample to find the robustness of RNN-GRU Model capability. Figure 13 shows a statistical view of results related to both models were deployed on newly generated datasets.

Issa Ahmed Model			**LTVM Data sets**
	CNN-LSTM		
		Accuracy	0.9447
	RNN-GRU		
		Accuracy	0.9450
Proposed Model			
	CNN-LSTM		
		Accuracy	0.9936
	RNN-GRU		
		Accuracy	0.9448

Fig.13. Statistical comparison of results using Generated ICMPv6 datasets.

The Issa Ahmed model implementation has an accuracy of 94.47% using the combination of CNN-LSTM and 94.50% using the combination of RNN-GRU. The proposed model implementation has an accuracy of 99.36% using the combination of CNN-LSTM and 94.48% using the combination of RNN-GRU. The

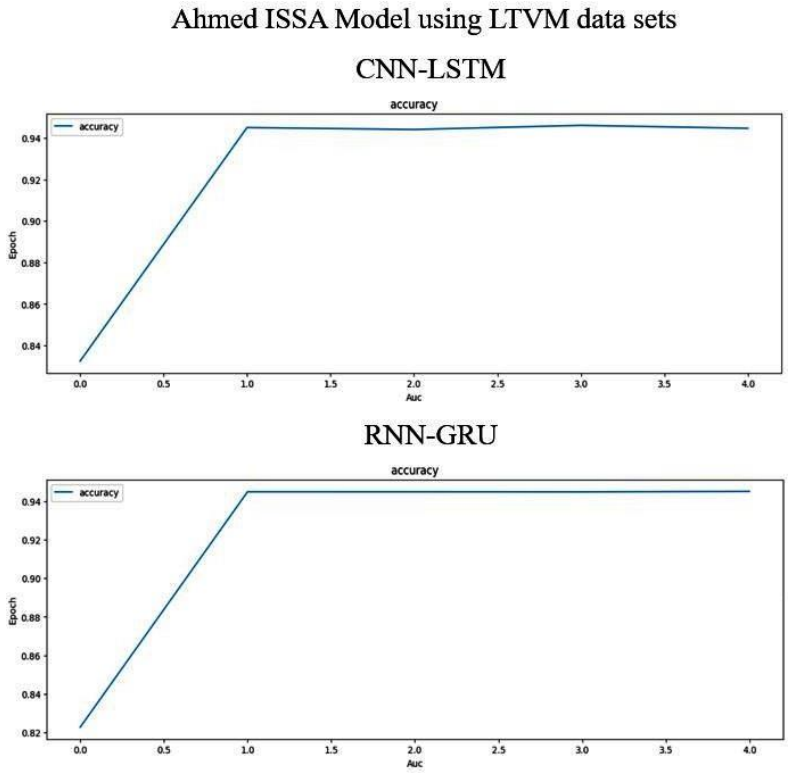

Fig.14a. Graphical format of the results of Issa model and Proposed model.

Figures 14a and 14b display the performance graphs of the Isaa Model and the proposed model. With the exception of the RNN-GRU graph in Figure 14b, all other graphs demonstrate ideal performance in terms of Area Under Curve (AUC). Although the accuracy metric stands at 94.51%, indicating room for improvement, it reflects promising performance overall. When compared to the results of the Isaa Model, the Proposed Model shows superior accuracy, highlighting its effectiveness and advantage.

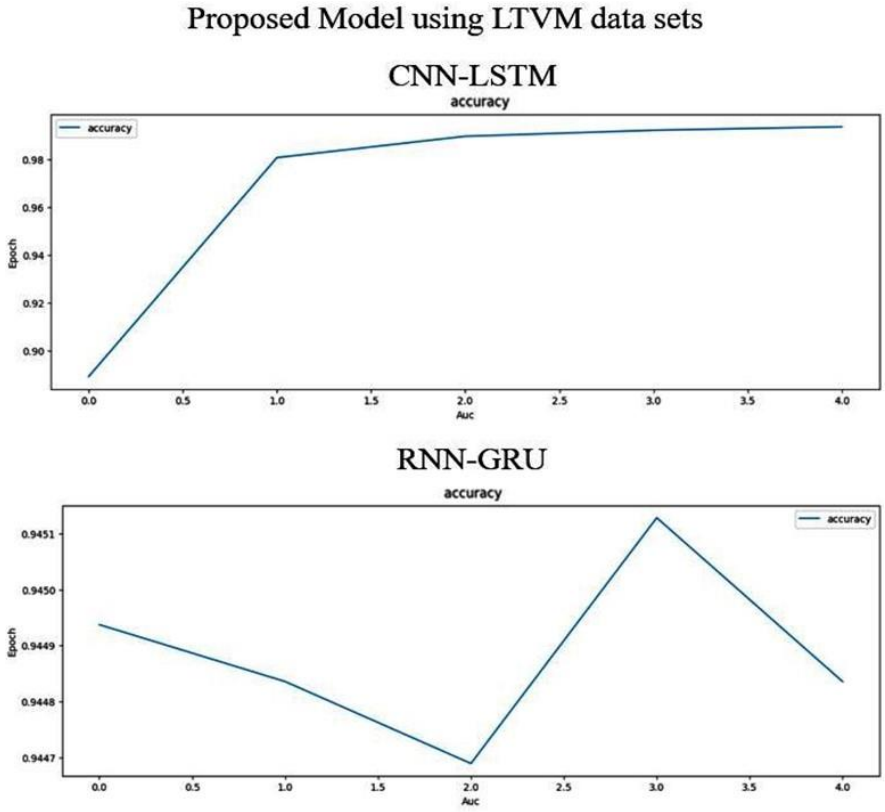

Fig.14b. Graphical format of the results of Issa model and Proposed model

5. CONCLUSION

An endeavour has been undertaken to propose a model that combines Recurrent Neural Networks (RNN) and Gated Recurrent Units (GRU). The architecture of the model is elucidated and deployed across four distinct datasets. In the experimental section, the results are juxtaposed, accompanied by graphical representations illustrating the findings.

Primary Datasets	Newly Generated ICMPv6 Datasets		LTVM Data sets
Issa Ahmed Model	CNN-LSTM	Accuracy	94.47%
	RNN-GRU	Accuracy	94.50%
Proposed Model			
	CNN-LSTM	Accuracy	99.36%
	RNN-GRU	Accuracy	94.48%
Secondary Datasets			
	NSL–KDD Datasets (ICMPv4)		
	CNN-LSTM	Accuracy	80.00%
	RNN-GRU	Accuracy	80.00%
	Mendeley Data sets (ICMPv4)		
	CNN-LSTM	Accuracy	97.07%
	RNN-GRU	Accuracy	95.06%
	Sain Malaysian University Datasets (ICMPv6)		
	CNN-LSTM	Accuracy	72.89%
	RNN-GRU	Accuracy	64.94%

Fig.15. Statistical comparison of results of all datasets.

Figure 15 depicts the comparison results and reveals discrepancies between the CNN-LSTM and RNN-GRU models across different datasets and employed Ahmed Issa model architecture and as well proposed model architecture. Additionally, the choice of CPU or GPU resources, particularly in platforms like Google Colab, impacts the reported results, with GPU computations often yielding higher performance closer to the CNN-LSTM model. Furthermore, variations in results are noted based on a number of features, type of dataset version, resources available, etc.

5.1. Critical Evaluation

Figure 16 illustrates the critical validation of the proposed architecture against state-of-the-art methods, indicating a comparative performance gap when considering the IP version. On observation, IPv6 related to the Omar Elejla model, when using RNN and LSTM, the proposed model, has the upper hand in the case of RNN. Further, techniques like SMOTE (Synthetic Minority Over-sampling Technique) or ADASYN (Adaptive Synthetic Sampling) could be employed to mitigate the imbalanced data problem between minor and major classes. These techniques are primarily used in binary classification tasks where imbalanced datasets can lead to biased models that underperform on the minority class.

Authors	Datasets	IP Version	Algorithm	Metric: Accuracy
Ahmed Issa	NSL-KDD	IPv4	CNN and LSTM	99.20%
Omar Elejla	Sain Malaysian	IPv6	RNN, LSTM,GRU	87.1%. 99.4% and 99.11%
Saif	ICDDoS2019	IPv4	GRU and RNN	99.91% - GRU
Kumar	CICDDoS2019	IPv4	LSTM, Bi- LSTM, Stacked LSTM, GRU	99.55% -Stacked LSTM
Tian	CICDDoS2019	IPv4	Random Forest, AE-DNN, and RNN	99.20%
Ullah	BoT-IoT, IoT Network Intrusion, MQTT-IoTIDS2020, and IoT-23	IPv4	CNNs across 1D, 2D, and 3D	99.74% 1D CNN
Proposed Mode	ICMPv6 Generated dataset	IPv6	RNN with GRU	94.48%

Fig.16. RNN-GRU Validation.

However, based on the accuracy 94.48% achieved on the face of it, when using ICMPv6 datasets the RNN-GRU proposed model demonstrates robustness and effectiveness in detecting DDoS attacks. The RNN-GRU proposed model emerges as a viable option for cost-effective computing and data streams such as time series, packet flow attacks, natural language processing, etc. Its efficacy is contingent upon factors such as the selection of the activation function and optimization function.

5.2. Future Research

This research demonstrates that combining RNNs and GRUs effectively detects DDoS attacks, with RNNs managing immediate packet sequences and GRUs handling historical patterns. This hybrid approach not only identifies and mitigates network threats such as DDoS and insider attacks but also excels in contexts with sequential data, dynamic patterns, and long-term dependencies, like phishing detection, and enhancing email security. The RNN-GRU model, which uses RNNs for short-term traffic patterns and GRUs for long-term dependencies, improves network traffic prediction and capacity planning. It aids in real-time traffic analysis and anomaly detection, helping to distinguish between normal fluctuations and potential threats. Further research could refine dynamic bandwidth allocation, optimize routing protocols, and enhance overall network performance by adapting routes based on congestion predictions. This combination can be deployed in domains related to mobile health monitoring, video analysis, speech recognition, natural language processing, time-series anomaly detection, object detection, real-time driving decision-making, and forecasting tasks such as stock market predictions.

REFERENCES

[1] Aydın H., Orman, Z. and , M.A., 2022. A long short-term memory (LSTM)-based distributed denial of service (DDoS) detection and defense system design in public cloud network environment. Computers and Security, 118, p.102725.

[2] Gaurav, A., Gupta, B.B. and Panigrahi, P.K., 2022. A novel approach for DDoS attacks detection in COVID-19 scenario for small entrepreneurs. Technological Forecasting and Social Change, 177, p.121554.

[3] Alharbi, A. and Alsubhi, K., 2021. Botnet detection approach using graph-based machine learning. Ieee Access, 9, pp.99166-99180.
[4] Mateen, H. and Shahzad, M., 2021, November. Factors Effecting Businesses due to Distributed Denial of Service (DDoS) Attack. In 2021 International Conference on Innovative Computing (ICIC) (pp. 1-7). IEEE
[5] Saqib, I., 2023. Comparison Of Different Firewalls Performance In A Virtual For Cloud Data Center. Journal of Advancement in Computing, 1(1), pp.21-28. —b6
[6] Tajdini, M., 2018. Developing an advanced IPv6 evasion attack detection framework. Liverpool John Moores University (United Kingdom).
[7] Mateen, H. and Shahzad, M., 2021, November. Factors Effecting Businesses due to Distributed Denial of Service (DDoS) Attack. In 2021 International Conference on Innovative Computing (ICIC) (pp. 1-7). IEEE.
[8] Tan, S., Zhong, X., Tian, Z. and Dong, Q., 2022. Sneaking through security: Mutating live network traffic to evade learning-based nids. IEEE Transactions on Network and Service Management, 19(3), pp.2295-2308.
[9] Issa, A.S.A. and Albayrak, Z., 2023. DDoS attack intrusion detection system based on hybridization of cnn and lstm. Acta PolytechnicaHungarica, 20(2), pp.1-19.
[10] Drewek-Ossowicka, A., Pietro laj, M. and Ruminski, J., 2021. A survey of neural networks usage for intrusion detection systems. Journal of Ambient Intelligence and Humanized Computing, 12(1), pp.497-514.
[11] Omar, Elejla.E., Belaton, B., Anbar, M., Alabsi, B. and Al-Ani, A.K., 2019. Comparison of classification algorithms on ICMPv6-based DDoS attack detection. In Computational Science and Technology: 5th ICCST 2018, Kota Kinabalu, Malaysia, 29-30 August 2018 (pp. 347-357). Springer Singapore.
[12] Saif.Ur Rehman, , Khaliq, M., Imtiaz, S.I., Rasool, A., Shafiq, M., Javed, A.R., Jalil,Z. and Bashir, A.K., 2021. DIDDOS: An approach for detection and identification of Distributed Denial of Service (DDoS) cyberattacks using Gated Recurrent Units (GRU). Future Generation Computer Systems, 118, pp.453-466.
[13] Kumar, K. and Behal, S., 2021, March. Distributed denial of service attack detection using deep learning approaches. In 2021 8th International Conference on Computing for Sustainable Global Development (INDIACom) (pp. 491-495).
[14] Tian, Q., Guang, C., Wenchao, C. and Si, W., 2021, May. A lightweight residual networks framework for DDoS attack classification based on federated learning. In IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS) (pp. 1-6). IEEE.
[15] Ullah Imtiaz. and Mahmoud, Q.H., 2021. Design and development of a deep learningbased model for anomaly detection in IoT networks. IEEE Access, 9, pp.103906-103926.
[16] Housman, O.G., Isnaini, H., Sumadi, F. and Setiawan, D., 2020. SDN-DDOS (ICMP, TCP,UDP). Mendeley Data, 2.
[17] Tavallaee, M., Bagheri, E., Lu, W. and Ghorbani, A.A., 2009, July. A detailed analysis of the KDD CUP 99 data set. In 2009 IEEE symposium on computational intelligence for security and defense applications (pp. 1-6). Ieee.
[18] Nazih, W., Hifny, Y., Elkilani, W.S., Dhahri, H. and Abdelkader, T., 2020. Countering ddos attacks in sip-based voip networks using recurrent neural networks. Sensors, 20(20), p.5875.
[19] Assis, M.V., Carvalho, L.F., Lloret, J. and Proen¸ca Jr, M.L., 2021. A GRU deep learning system against attacks in software-defined networks. Journal of Network and Computer Applications, 177, p.102942.
[20] Reyad, M., Sarhan, A.M. and Arafa, M., 2023. A modified Adam algorithm for deep neural network optimization. Neural Computing and Applications, 35(23), pp.17095-17112.
[21] Elejla, O.E., Anbar, M., Hamouda, S., Faisal, S., Bahashwan, A.A. and Hasbullah, I.H.,2022. Deep-learning-based approach to detect ICMPv6 flooding DDoS attacks on IPv6 networks. Applied Sciences, 12(12), p.6150.
[22] Conta, A., Deering, S. and Gupta, M., 2006. Internet control message protocol (icmpv6) for the internet protocol version 6 (ipv6) specification (No. rfc4443).

[23] Liu Y., Yang, W., Zhou, Z.,, Q., Li, Z. and Li, S., 2022, May. P4-NSAF: defending IPv6 networks against ICMPv6 DoS and DDoS attacks with P4. In ICC 2022-IEEE International Conference on Communications (pp. 5005-5010). IEEE.

[24] Sumathi, S., Rajappa, S., Kumar, L.A. and Panneerselvam, S., 2021. Advanced Decision Sciences Based on Deep Learning and Ensemble Learning Algorithms: A Practical Approach Using Python. Nova Science Publishers.

[25] Alshra'a, A.S., Farhat, A. and Seitz, J., 2021. Deep learning algorithms for detecting denial of service attacks in software-defined networks. Procedia Computer Science, 191, pp.254263.

[26] Srinivas, T., Aditya Sai, G. and Mahalaxmi, R., 2022. A comprehensive survey of techniques, applications, and challenges in deep learning: A revolution in machine learning. International Journal of Mechanical Engineering, 7(5), pp.286-296.

[27] Yu, T. and Zhu, H., 2020. Hyper-parameter optimization: A review of algorithms and applications. arXiv preprint arXiv:2003.05689.

[28] Torres, J.F., Martínez-Álvarez, F. and Troncoso, A., 2022. A deep LSTM network for the Spanish electricity consumption forecasting. Neural Computing and Applications, 34(13), pp.10533-10545.

[29] Lei, J., Ren, C., Li, W., Fu, L., Li, Z., Ni, Z., Li, Y., Liu, C., Zhang, H., Chen, Z. and Yu, T., 2022. Prediction of crucial nuclear power plant parameters using long short-term memory neural networks. International Journal of Energy Research, 46(15), pp.21467-21479.

[30] Francois Chollet (2017) "Keras 3 API documentation". In Topic: Model Checkpoint, URL: https://keras.io/api/callbacks/model_checkpoint/; Accessed: on 20/07/2024.

[31] Francois Chollet (2017). "Keras 3 API documentation". In Topic: Early Stopping, URL: https://keras.io/api/callbacks/early_stopping/; Accessed: on 20/07/2024

[32] Francois Chollet (2017). "Keras 3 API documentation". In Topic:ReduceLROnPlateau, URL: https://keras.io/api/callbacks/reduce_lr_on_plateau/; Accessed: on 20/07/2024

[33] Kanellopoulos, D.N. and Wheeb, A.H., 2020. Simulated performance of TFRC, DCCP, SCTP, and UDP protocols over wired networks. International Journal of Interdisciplinary Telecommunications and Networking (IJITN), 12(4), pp.88-103.

AUTHORS

Mr. Om Salamkayala: I am a qualiiedM.Sc in Forensics Computing and also a PgCHPE fellow. Currently pursuing inal year PhD from Staffordshire University. I gained over 8 years of IT experience out of 6 years in core industry related to Digital Forensics. I am also currently working as Part Time Lecturer in Staffordshire University.

Dr. Saeed: works with Staffordshire University as a Lecturer. He did his Ph.D in Robotics and Intelligent Systems from Kobe University. He started his career with leadership roles like chairing the Amirkabir Robotic Center, and collaborating with international researchers from Japan, France, Australia, and the USA. Teaching was also a integral to his professional journey, with experience of designing the graduate and undergraduate courses spanning Machine Learning, Robotics, and Computer Architecture. His contributions to the academic community are evident through numerous publications in esteemed journals and conferences, showcasing expertise in AI, robotics, and related domains.

Mr. Howard: has 30+ years of professional experience in networking, both in technical and academic and a Chartered Electronics Engineer. He works with Staffordshire University in course director capacity and with his leadership abilities involved in design and development course curriculum development at various levels . His main teaching interests are in Cyber and Networks at both undergraduate and postgraduate levels. His expertise extends to international partner relations, quality assurance, and staff development across Greece, China, and Vietnam. Additionally, he pioneered lexible learning awards for part-time students, focusing on Cisco qualiications like CCNA, CCNP, CCNA Security, and active member of University's Cisco Academy.