

Machine Learning-Driven Approach for DDOS Attacks Detection using Neural-Based Networks: A Proficiency Study

Abdul Ghani Ansari ^a, Fareed Ahmed Jokhio ^b, Muhammad Shehram Shah Syed ^{c,*}, Hina Dharejo ^a, Fayaz Ahmed Memon ^d

a Department of Information Technology, QUEST Nawabshah,

b Department of Computer Systems Engineering, QUEST Nawabshah.

c Department of Software Engineering, Mehran University of Engineering & Technology, Jamshoro.

d Department of Software Engineering, QUEST Nawabshah.

* Corresponding author: shehram.shah@faculty.muet.edu.pk

Received: 6 August 2024, Revised: 27 June 2025, Accepted: 30 June 2025, Published: 01 July 2025

KEYWORDS	ABSTRACT
Knowledge Discovery Dataset (KDD) Artificial Neural Network (ANN) Distributed Denial of Service (DDoS) Attacks Resilient Backpropagation Fletcher-Powell Conjugate Gradient Scaled Conjugate Gradient	Cyber-attacks pose significant threats to the Internet and its connected resources, causing harm to institutions and governments. Advanced technologies like cloud computing, the Internet of Things (IoT), and Artificial Intelligence (AI) have made these attacks harder to detect. Botnets, controlled by malicious actors, are at the core of many Internet attacks, including Distributed Denial of Service (DDoS) attacks. Despite years of DDoS incidents, effective defense mechanisms are lacking. In our research, we use Resilient Backpropagation, Fletcher-Powell Conjugate Gradient (FPCG), and Scaled Conjugate Gradient with three Artificial Neural Network (ANN) models: Feedforward Neural Network (FFNN), Cascade-Forward Neural Network (CFFNN), and Fitnet Neural Network (FNN) to detect DDoS attacks. The results showed that the FNN achieved higher accuracy in less time.

1. Introduction

In recent years, cyber-attacks such as Distributed Denial of Service (DDoS) attacks have surged in size and intensity, becoming the dominant cyber threat. They pose a significant risk to Internet resource security and availability. DDoS attacks use coordinated distributed resources to disrupt legitimate users' access to online services[1]. They can target various assets, such as system resources, network bandwidth, and critical infrastructure. What's concerning is their ability to mimic normal traffic, making detection difficult [2]. DDoS attacks come in different forms, generating high traffic volume in a short period [3], with the latter being especially challenging to counter. Evolving technology has made DDoS attacks more elusive, where they leverage technologies such as cloud computing, the Internet of

Things (IoT), and other network resources. IoT devices can be used to launch DDoS attacks, which can dramatically increase the attack's power and cause severe harm to the targets. IP spoofing and flooding attacks are two particular kinds of DDoS attacks. In IP spoofing, attackers mimic trusted sources [4], whereas a flooding attack overwhelms the target with excessive packets. Flooding attacks include Transmission Control Protocol (TCP) flood, Internet Control Message Protocol(ICMP) flood (Smurf attack), User Datagram Protocol (UDP) flood, and Domain Name System (DNS) amplification attack, where attackers falsify the victim's source address to prompt disruptive DNS responses. Detecting DDoS attacks is tough due to their resemblance to legitimate traffic. Initial identification involves spotting abnormal traffic patterns[5]. DDoS attacks aim to disrupt network

operations and resource availability, causing downtime and service disruptions in malicious flood attacks.

This research focuses on DDoS attack detection with Resilient Back-propagation, Fletcher-Powell Conjugate Gradient, and Scaled Conjugate Gradient training algorithms using three types of Artificial Neural Networks, including Feedforward Neural Networks (FFNN), Cascade-Forward Neural Networks (CCFNN), and Fitnet Neural Network (FNN). These networks require precise training, validation, and extensive testing to identify fraudulent data and network attacks. The research explores DoS flooding attacks and proposes a practical method for detecting DDoS attacks. The paper outlines the DDoS problem, and existing state-of-art detection methods, ultimately providing an effective way to safeguard networks against DDoS attacks.

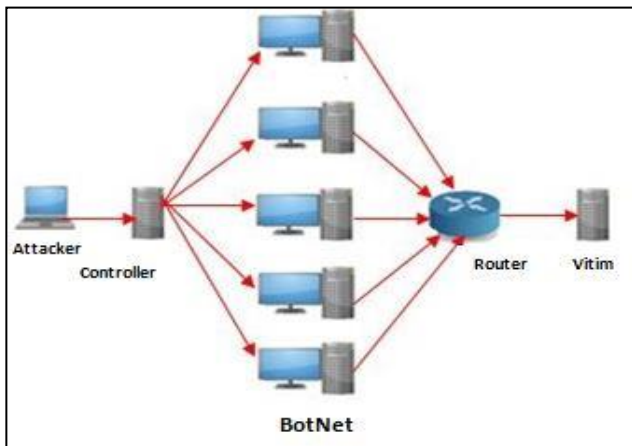


Fig. 1 DDoS Attack Scenario [6]

Fig. 1. shows how DDoS attacks employ multiple systems working together to target a cloud service. Attackers gain control over victim systems, often called zombies or slaves, through malware. These enslaved systems create a botnet. Attackers can then disrupt a service by instructing the botnet to generate fake traffic, rendering cloud resources inaccessible to legitimate users[6].

2. Classification of DDOS Attacks

The classification of DDOS attacks is depicted in Fig. 2, while detail about the classification of DDOS attacks is given as under:

1. Application layer attacks (Layer 7) aim to exploit software vulnerabilities, disrupting specific applications or services on a network.
2. Infrastructure layer attacks target network foundational components like routers, switches, and protocols. They include Volumetric Attacks (e.g., UDP or ICMP

floods) and Protocol-Based Attacks (e.g., SYN/ACK floods).

3. A zero-day attack exploits previously unknown vulnerabilities, catching the target unaware, as no prior defense preparation is possible. Developers have no time to create patches or fixes [7].

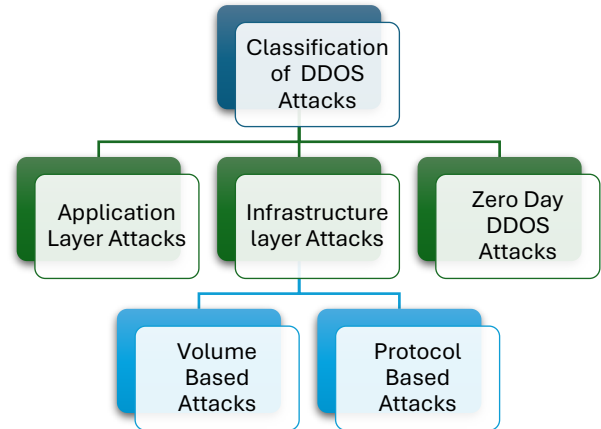


Fig. 2. Classification of DDOS Attacks [7]

3. Impacts of DDOS Attacks on Network Resources and Online Business

DDoS attacks have increased in frequency and complexity, impacting bandwidth and memory, causing website or application issues [8], [9]. They also have lesser-known consequences:

- **Enhanced Packet Loss and Latency:** DDoS attacks overload networks, causing higher latency and packet loss, leading to sluggish connections and network dysfunction.
- **Network Congestion:** DDoS attacks worsen network congestion, causing latency, packetloss, dropped connections, and potential outages.
- **Resource Depletion:** DDoS attacks deplete server or application resources (CPU/memory), leading to poor response times, crashes, and potential business losses.
- **Security Vulnerabilities:** DDoS attacks can create chaos for attackers to launch malware or phishing campaigns, and their data volume can mask other attacks.
- **Reputational Damage:** DDoS attacks result in downtime, poor user experiences, and data breaches, harming a company's reputation and potentially causing financial losses and brand damage.

4. Neural-Based Networks (NBN)

Neural-based Networks emulate biological nervous systems. It learns from instances and adapts to tasks like pattern recognition or data classification by adjusting neuron connections. They excel at deriving

meaning from complex or imprecise data and identifying subtle patterns. Various types of Neural-Based Networks exist in machine learning. In our study, we used a feed-forward neural network, a Cascade Forward Neural Network, and a Fitting Neural Network to detect DDoS attacks [10], [11].

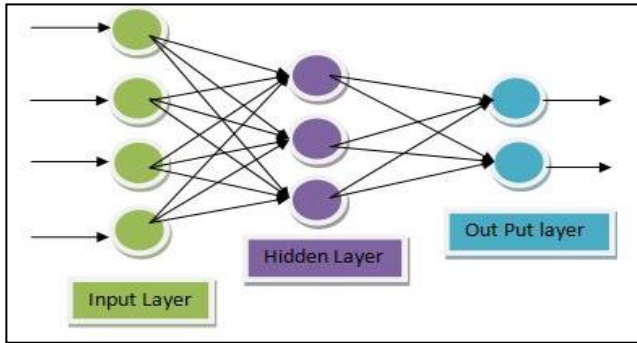


Fig. 3 Neural-Based Network

Fig.3 shows a typical Neural-based Network consisting of interconnected layers with nodes, each having activation function [12]. A typical NBN has three main layers:

1. **Input Layer:** This layer receives attribute values and passes patterns to hidden layers without modifying data.
2. **Hidden Layer:** Hidden layers transform input data through weighted connections, and there can be multiple layers. Weighted inputs are summed to produce an output.
3. **Output Layer:** Connected to hidden layers, this generates output values, often predicting a response variable, with a single output node in classification tasks.

The network's effectiveness depends on weight selection, a process distinct from conventional information processing.

4.1 Feed-Forward Neural Network (FFNN).

The FFNN or Multilayer Perceptron is a basic type of Neural-Based Network used for tasks like regression and classification. Unlike Recurrent Neural Networks (RNNs), FFNNs process data in a single direction without looped connections. They're straightforward and commonly used in supervised learning [13]. An illustration of FFNN is provided in Fig. 4. It also uses weights, biases, and an activation function. The network itself learns values of weights and biases during training. Some possible activation functions include rectified linear unit (ReLU), hyperbolic tangent, and sigmoid function. In this research, a sigmoid function (logistic function), is used as an activation function in the hidden layer because it is smooth and differentiable. However, it slows down the

training process because sometimes its derivative becomes very small for its inputs.

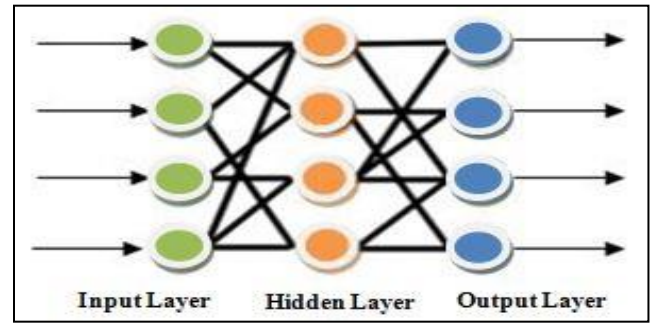


Fig.4 Feed Forward Neural Network [13]

4.2 Cascade-Forward Neural Network (CCFNN)

A Cascade-Forward Neural Network is similar to a feed-forward network but includes connections between the input and previous layers. The output layer is also directly connected to the input layer with the oversized layer of a network with three layers[14]. This preserves both linear and non-linear relationships. Refer to Fig.5 for clarification.

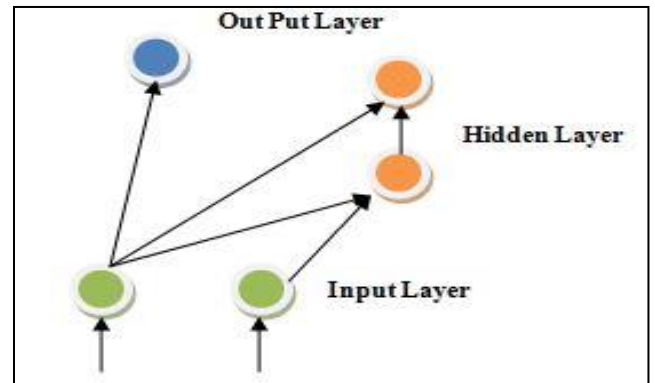


Fig.5 Cascade-Forward Neural Network [14]

4.3 Fit Net Neural Network (FNN)

The FNN includes initial layers, which can be considered hidden or encoders, in addition to the input and output layers. In a shallow network, these layers are not deeply hidden. Shallow networks can fit any function but require full weighting, resulting in a significant number of parameters. In essence, a 'shallow' neural network typically has just one hidden layer [15], as opposed to deep neural networks with multiple hidden layers of different types, as shown in Fig. 6.

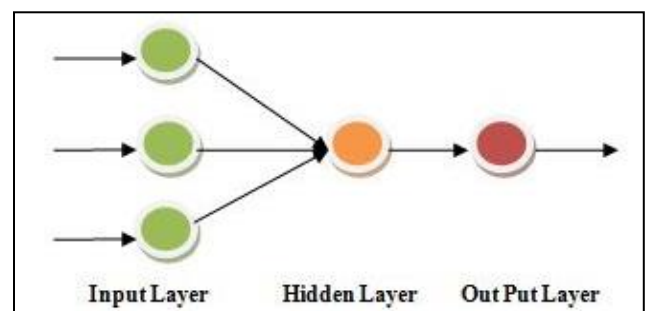


Fig.6 Fit Net Neural Network [15]

5. Training Algorithms

In this research, three training algorithms are used (i) Resilient Back-propagation algorithm, (ii) Fletcher-Powell Conjugate Gradient algorithm, and (iii) Scaled Conjugate Gradient.

5.1 Resilient Back-Propagation (Rprop)

Rprop algorithm is an adaptive variation of traditional backpropagation for training NBN's. Unlike standard backpropagation, Rprop focuses on how weight changes affect the error, adjusting step sizes for each weight based on the direction and magnitude of changes. This adaptability helps Rprop converge faster, making it efficient for training NBN's, especially in noisy gradient information or complex, high-dimensional problems. Multilayer neural networks employ sigmoid transfer functions to condense the input data within a bounded output range [16]. This leads to minimal weight adjustments based on partial derivatives, potentially hindering network convergence. Resilient back propagation resolves this issue by solely determining the weight update direction through partial derivatives, resulting in reduced memory requirements, consider equation (1).

$$dX = \text{delta}X \cdot \text{sign}(gX) \quad (1)$$

Where $\text{delta}X$ = derivative of performance,

gX = gradient, dX = change in weight

5.2 Fletcher-Powell Conjugate Gradient (FPCG)

The FPCG method, also known as the Powell-Fletcher-Reeves method, is an optimization approach employed to locate the minimum of a function. It's commonly applied in unconstrained optimization scenarios. The train CGF algorithm is a variation of the Conjugate Gradient method [17], capable of training networks with derivative functions for weights, net input, and transfer functions through Backpropagation. Adjustments to each vector M_i follow a specific equation (2).

$$M = M + a(dM) \quad (2)$$

Where, dM indicates the search direction, with 'a' as the parameter for minimizing performance in that direction. The line search function is used to find the minimum point. Initially, the search direction aligns with the negative of the performance gradient. In later iterations, it's determined based on the updated gradient and the previous search direction, as per the provided equation (3).

$$dM \cdot gM = -\beta \cdot dM \quad (3)$$

Where, gM refers to the gradient. The parameter β can be determined through various methods. In the

case of the Fletcher-Reeves variant of the conjugate gradient, it is calculated using the Equation (4) [18].

$$\beta_k = \frac{g_k^T g_k}{g_{k-1}^T g_{k-1}} \quad (4)$$

Where the norm is square of the previous gradient and is the norm square of the current gradient.

5.3 Scaled Conjugate Gradient (SCG)

The SCG algorithm is an optimization technique employed for the training of artificial neural networks, especially those of the feed-forward variety. It represents a variation of the conjugate gradient optimization method and is particularly renowned for its efficiency in the training of neural networks, especially in scenarios where other optimization methods might encounter challenges [19]. The fundamental back-propagation algorithms modify weights by considering the negative gradient, but this may not lead to rapid convergence. To address this, it integrates elements of the model-trust region approach found in the Levenberg-Marquardt algorithm along with the conjugate gradient approach, necessitating numerous interim updates, consider equation (5) [16].

$$X_{k+1} = X_k + \alpha_k p_k \quad (5)$$

Where p_k = search direction, α_k = scalar coefficients

6. Literature Review

Many researchers have committed their efforts to constructing classification models using machine learning techniques. This summary encompasses discussions of specific studies that have contributed to the advancement of DDoS detection systems. Gupta and colleagues [20] developed a machine learning-based strategy that examines unique characteristics of IoT networks to identify packet-level attacks. They employed six ML classifiers to differentiate between normal and malicious traffic patterns with an accuracy ranging from 97% to 99%. Additionally, they applied local router filtering to mitigate attack traffic. In [21] authors proposed a method to classify infected smart devices in Smart home applications. They used machine learning classifiers to recognize smart devices based on their network traffic features. By comparing attributes during regular and attack scenarios, this approach identifies devices that may exhibit abnormal behavior during an attack, achieving a 99.79% accuracy based on smart device traffic flow features. An innovative approach has been introduced to detect intrusions in IoT networks using Deep Learning (DL) principles to classify traffic flows [22]. They leveraged an IoT dataset to extract generic characteristics from packet-level field data. Their Feed

Forward Neural Network (FFNN) model effectively classified attacks against IoT devices, including DoS, DDoS, and investigation, with high accuracy. Authors in [23] designed a DDoS detection system utilizing machine learning techniques. They conducted a comparative analysis of Naive Bayes and Random Forest algorithms using existing datasets. Naive Bayes outperformed Random Forest with a 90.90% accuracy rate compared to 78.18%. Researchers in [24] applied a deep intelligence technique, utilizing a radial basis function with varying levels of abstraction to extract intelligence. They used NSL KDD and UNSW NB15 datasets and claimed superior accuracy compared to other methods. Authors in [25] implemented a DDoS attack detection technique and compared it to five widely recognized machine learning methods. The KNN method achieved a satisfactory precision level of 85%. The accuracy of a model for detecting DoS attacks using machine learning and neural networks was optimized by parameter tuning and achieved a 99.95% accuracy on the CICIDS 2017 dataset using the Random Forest algorithm [26]. Authors in [27] proposed a model to distinguish legitimate users from botnets used in DDoS attacks at the application layer. They used seven extracted features from user sessions and employed the SVM algorithm for detection, effectively identifying application-layer DDoS attacks. A system utilizing recurrent deep neural networks (RNNs) to analyze network traffic and monitor network activities was implemented [28]. Their experiments were conducted on well-known datasets, and they claimed to have achieved improved results. Authors in [29] developed a deep learning technique for threat analysis, using an Artificial Neural Network (ANN) based on supervised learning. Their experiment focused on analyzing IP traces to detect and classify normal and attack packets in an IoT network, achieving a promising accuracy rate of 99.4%.

7. Methodology for DDOS attack detection

The research utilized three Neural-Based Networks as classifiers, employing the training algorithms discussed earlier. They are evaluated using a modified Knowledge Discovery Database (KDD). The KDD data classes are categorized into attacks and non-attacks. The system's sequential process includes KDD dataset, data preprocessing, three neural networks exploitation with training algorithms for DDoS attack detection, and prediction accuracy, as depicted in Fig. 7.

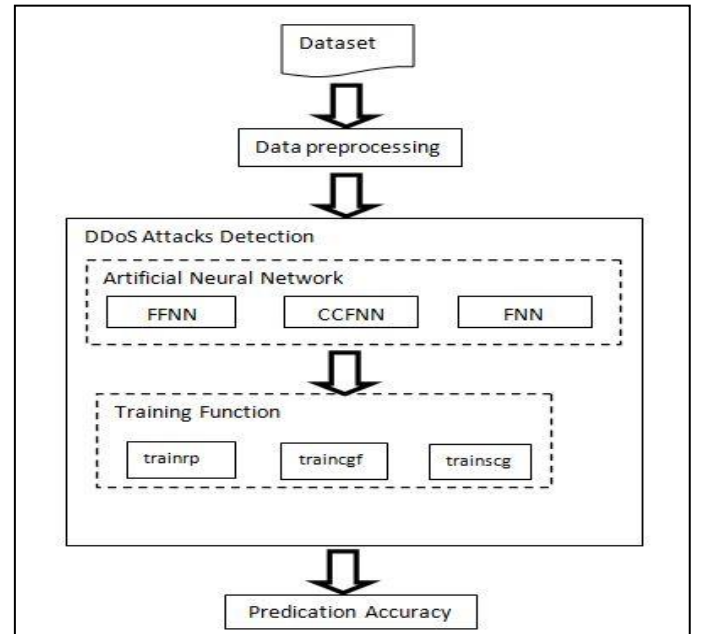


Fig. 7. Methodology for DDoS Detection [29]

8. KDD Dataset Acquisition

We start by obtaining the initial KDD dataset, specifically the KDD-Cup 1999 dataset, which contains records of attacks and intrusions.

1. Data Preparation: Next, we clean and organize the dataset, assigning values to protocol types, attack categories, and flags. Neural Network Model Creation: With the prepared data, we construct the proposed neural network model.
2. Neural Network Training: We use FFNN, CCFNN, and FNN for training, comparing Resilient Backpropagation, Fletcher-Powell Conjugate Gradient and Scaled Conjugate Gradient algorithms to determine the most effective for detecting DDoS attacks.
3. Outcome Assessment: After training, we evaluate and identify the algorithm with the highest effectiveness in detecting attacks.

9. Experimental Results & Analysis

This study employed the MATLAB 2018a software tool. Initially, preprocessing was performed on the KDD dataset, assigning values to protocol types, attack classifications, and flags. Afterwards, neural network models were constructed and trained. Upon the termination of the training phase, results about DDoS attacks were obtained. Specifically, three algorithms, namely Resilient Back propagation (trainrp), Fletcher-Powell Conjugate Gradient (traincgf), and Scaled Conjugate Gradient (trainscg), were utilized to train the neural networks, updating weight and bias values. The data has both numeric and alphanumeric inputs; however, MATLAB only accepts numeric inputs. Therefore, all alphanumeric values were replaced with numerical values.

Table 1

Consolidated Mapping of Protocols, Flags, and Attack Types to Numeric Values

Category	Original Value	Mapped Value
Protocol	TCP	1
	UDP	2
	ICMP	3
	FTP	4
Flag	S0	1
	SF	2
	S1	3
	REJ	4
Attack	Normal	1
	Smurf	2
	Neptune	3
	Teardrop	4

Table 1 provides a consolidated view of how various categorical features in the KDD dataset were numerically encoded for neural network training. Network protocols, connection flags, and attack types—originally in textual format—were mapped to numeric values to meet the input requirements of MATLAB and ANN models. This conversion was essential for ensuring proper data handling and model compatibility. The "Other Attacks" category combines several low-frequency attack types into a single class to simplify classification and manage data imbalance, enhancing the overall learning performance of the models.

Table 2:

Summary of KDD Dataset Used in Experiments

Attribute	Description
Dataset Name	KDD Cup 1999
Total Samples	500,000
Number of Features	41
Feature Types	Numeric and Alphanumeric (converted to numeric for ANN input)
Number of Classes	5 (After grouping multiple attack types)
Class Descriptions	1: Normal, 2: Smurf, 3: Neptune, 4: Teardrop, 5: All other attack types grouped
Most Frequent Attacks	Smurf and Neptune
Minority Classes	Teardrop and grouped rare attacks

Table 2 summarizes the standard KDD Cup 1999 dataset used in the research work. It comprises over

500,000 samples with 41 features, all converted to numeric form for neural network training. To simplify classification, attacks were grouped into five classes: one normal and four attack types, with the fifth class combining less frequent attacks. Smurf and Neptune were the most common, while others like Teardrop appeared less often. This grouping helps manage class imbalance and improve model performance.

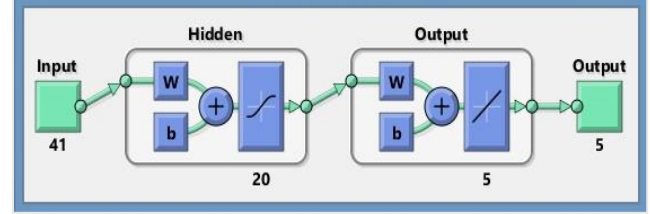
**Fig. 8.** FFNN with Input, Hidden, and Output layers

Fig. 8 shows the model of a Feed-Forward Neural Network. There are 41 inputs, with 20 neurons in the hidden layer and 5 neurons in the output layer. In the dataset, there are more than 20 classes. To simplify matters, DDoS attacks have been categorized into five distinct classes.

9.1 Results of the Feed Forward Neural Network using Resilient Backpropagation

FeedForward Neural Networks Resilient Backpropagation Confusion Matrix					
Output Class	1	2	3	4	5
	97038 19.6%	98 0.0%	8 0.0%	8 0.0%	2712 0.5%
	25 0.0%	280692 56.8%	0 0.0%	0 0.0%	0 0.0%
	11 0.0%	0 0.0%	107191 21.7%	0 0.0%	23 0.0%
	0 0.0%	0 0.0%	0 0.0%	971 0.2%	93 0.0%
	204 0.0%	0 0.0%	2 0.0%	0 0.0%	4945 1.0%
Target Class					
	1	2	3	4	5
	99.8% 0.2%	100.0% 0.0%	100.0% 0.0%	99.2% 0.8%	63.6% 36.4%
					99.4% 0.6%

Fig. 9. Confusion Matrix of FFNN using Resilient Backpropagation.

Fig.9 shows the classification results for the Feed-Forward Neural Network using Resilient Backpropagation; here, total accuracy is 99.4%. In this research, Feed Forward networks used a sigmoid activation function in hidden layers, which is non-linear and can learn various types of complex patterns in the input data. By nature, the input data is non-linear; therefore, the Resilient Back-propagation algorithm helps the Feed Forward Neural Network to learn in conditions where input patterns are changing. The training algorithm sets the weights according to the changing patterns in input data. As can be seen in the confusion matrix shown in Fig. 9, the misclassification of normal requests is only 0.2%, and the misclassification of other types of attacks is also

very less. In the confusion matrix, it can be seen that the misclassification of the fifth class is 36.4%. This is because this class consists of various types of attacks, and the Resilient Back propagation algorithm could not capture the features of all these attacks. As the total number of attacks available in this class is less than five thousand, the impact of this misclassification is not that much, and overall misclassification is only 0.6%. Total training time for this experiment is 0:05:46, five minutes and forty-six seconds.

9.2 Results of the Feed Forward Neural Network using Fletcher-Powell Conjugate Gradient

To check the efficiency of the Feed-Forward Neural Network, the model is trained with another training algorithm named Fletcher-Powell Conjugate Gradient and it belongs to the family of methods that require memory. However, it requires less memory as compared with other optimization algorithms and is suitable for problems where the amount of data is very large.

Output Class	1	2	3	4	5	
1	97012 19.6%	14 0.0%	5 0.0%	6 0.0%	362 0.1%	99.6% 0.4%
2	18 0.0%	280762 56.8%	0 0.0%	0 0.0%	0 0.0%	100.0% 0.0%
3	2 0.0%	0 0.0%	107195 21.7%	0 0.0%	13 0.0%	100.0% 0.0%
4	0 0.0%	0 0.0%	0 0.0%	970 0.2%	0 0.0%	100% 0.0%
5	246 0.0%	14 0.0%	1 0.0%	3 0.0%	7398 1.5%	96.6% 3.4%
	99.7% 0.3%	100.0% 0.0%	100.0% 0.0%	99.1% 0.9%	95.2% 4.8%	99.9% 0.1%
	1	2	3	4	5	

Fig. 10. Confusion Matrix of FFNN using Fletcher-Powell Conjugate Gradient

Fig. 10 shows the classification results of a Feed-forward neural Network using the Fletcher-Powell Conjugate Gradient; here, overall accuracy is 99.9%. It can be seen in the confusion matrix that the misclassification of the first four classes is much less. In class five, misclassification is 4.8%. Here, it is concluded that this training algorithm has learned the data patterns efficiently as compared with the Resilient Backpropagation algorithm. The main reason for better performance is due to the usage of memory by the Fletcher-Powell Conjugate Gradient. Although the accuracy results of this algorithm are better, however, due to complexity and memory usage,

it requires more training time. Here training time for this experiment is 0:21:41, Twenty-one minutes and forty-one seconds.

9.3 Results of the Feed Forward Neural Network Using Scaled Conjugate Gradient

Output Class	1	2	3	4	5	
1	96956 19.6%	16 0.0%	11 0.0%	6 0.0%	319 0.1%	99.6% 0.4%
2	22 0.0%	280761 56.8%	0 0.0%	0 0.0%	0 0.0%	100.0% 0.0%
3	7 0.0%	0 0.0%	107190 21.7%	0 0.0%	17 0.0%	100.0% 0.0%
4	0 0.0%	0 0.0%	0 0.0%	970 0.2%	2 0.0%	99.8% 0.2%
5	293 0.1%	13 0.0%	0 0.0%	3 0.0%	7435 1.5%	96.0% 4.0%
	99.7% 0.3%	100.0% 0.0%	100.0% 0.0%	99.1% 0.9%	95.7% 4.3%	99.9% 0.1%
	1	2	3	4	5	

Fig. 11. Confusion Matrix of FFNN using Scaled Conjugate Gradient

Fig. 11 shows the classification result for the Feed Forward Neural Network using a Scaled Conjugate Gradient, which is an overall accuracy of 99.9%. The confusion matrix illustrates minimal misclassification in the first four classes. Class five shows a misclassification rate of 4.3%. This indicates that the training algorithm has efficiently learned data patterns compared to Resilient Backpropagation and Fletcher-Powell Conjugate Gradient algorithms. The accuracy results of this algorithm are satisfactory. Despite using less training time than the Fletcher-Powell Conjugate Gradient, it achieves overall better accuracy than Resilient Backpropagation. The training time for this experiment is 8 minutes and 25 seconds.

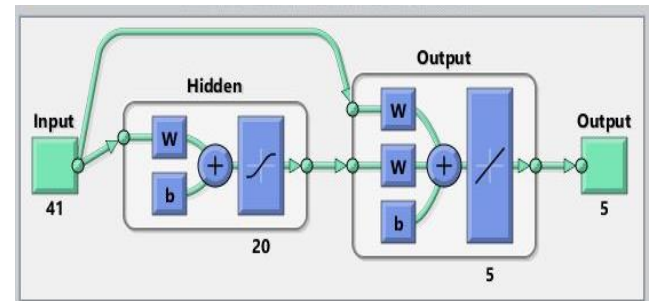


Fig. 12. CCFNN with Input, Hidden, and Output layers.

Fig. 12 shows the model of Cascade Neural Network. This model consists of 41 input elements, using 20 neurons in the hidden layer and 5 neurons in the output layer.

9.4 Results of the Cascade Neural Network using Resilient Backpropagation

Output Class	1	2	3	4	5	
1	96898 19.6%	122 0.0%	32 0.0%	7 0.0%	2998 0.6%	96.8% 3.2%
2	72 0.0%	280668 56.8%	0 0.0%	0 0.0%	74 0.0%	99.9% 0.1%
3	55 0.0%	0 0.0%	107166 21.7%	0 0.0%	233 0.0%	99.7% 0.3%
4	3 0.0%	0 0.0%	3 0.0%	972 0.2%	14 0.0%	98.0% 2.0%
5	250 0.1%	0 0.0%	0 0.0%	0 0.0%	4454 0.9%	94.7% 5.3%
	99.6% 0.4%	100.0% 0.0%	100.0% 0.0%	99.3% 0.7%	57.3% 42.7%	99.2% 0.8%
	1	2	3	4	5	
	1	2	3	4	5	

Fig.13. Confusion Matrix of Cascade Neural Network using Resilient Backpropagation.

Fig. 13 shows the classification result for the Cascade Neural Network using Resilient Backpropagation has an overall accuracy of 99.2%. In this experiment, the Cascade Neural Network uses a special function called sigmoid in the hidden layers. This function is good at understanding complicated patterns in the information we give it. Rprop is a smart way to adjust the weights in the network, making it learn faster during training. This is helpful for the Cascade Neural Network, helping it learn and get used to complex patterns quickly. Rprop is also good at handling noisy data and doesn't get bothered by changes in how fast it learns. It also helps improve how the Cascade Neural Network learns by adjusting weights based on how fast the information is changing, making the learning process steadier and more effective. Rprop is good at changing based on the dynamic nature of Cascade Neural Networks, making it a great way to improve this kind of neural network. The confusion matrix, which shows the misclassification of the fifth class, is 42.7%. This is because there are many different types of attacks in this class, and the Resilient Backpropagation algorithm can't recognize all of them. But since there aren't many attacks in this class (less than five thousand), it doesn't make a big problem. So, the overall mistake in recognizing things is only 0.8%. The whole time it took for this experiment was six minutes and twenty-seven seconds which is less than the Fletcher-Powell Conjugate Gradient.

9.5 Results of the Cascade Neural Network using Fletcher-Powell Conjugate Gradient

Output Class	1	2	3	4	5	
1	96981 19.6%	12 0.0%	15 0.0%	6 0.0%	447 0.1%	99.5% 0.5%
2	27 0.0%	280773 56.8%	0 0.0%	0 0.0%	0 0.0%	100.0% 0.0%
3	4 0.0%	0 0.0%	107184 21.7%	0 0.0%	20 0.0%	100.0% 0.0%
4	0 0.0%	0 0.0%	0 0.0%	970 0.2%	40 0.0%	96.0% 4.0%
5	266 0.1%	5 0.0%	2 0.0%	3 0.0%	7266 1.5%	96.3% 3.7%
	99.7% 0.3%	100.0% 0.0%	100.0% 0.0%	99.1% 0.9%	93.5% 6.5%	99.8% 0.2%
	1	2	3	4	5	
	1	2	3	4	5	

Fig.14. Confusion Matrix of Cascade Neural Network using Fletcher-Powell Conjugate Gradient

Fig. 14 shows the classification result for the Cascade Neural Network using the Fletcher-Powell Conjugate Gradient, which is 99.8% as a whole accuracy. The Fletcher-Powell Conjugate Gradient (FPCG) algorithm is promising for Cascade Neural Networks (CNNs) due to its efficient weight adjustment, facilitating faster convergence in the learning process. It exhibits robustness in handling noisy data, reducing sensitivity to learning rate choices. FPCG enhances training dynamics by adjusting weights based on gradient information, ensuring stability and effectiveness in the learning process. Its adaptability aligns well with the dynamic nature of CNNs, making it a suitable optimization algorithm for improved performance and adaptability in CNNs. The confusion matrix indicates minimal misclassification in the first four classes, while 6.5% misclassification is recorded in class five. However, it necessitates more training time, taking a total of 24 minutes and nine seconds for this experiment.

9.6 Results of the Cascade Neural Network using Scaled Conjugate Gradient

Output Class	1	2	3	4	5	
1	96962 19.6%	18 0.0%	6 0.0%	8 0.0%	469 0.1%	99.5% 0.5%
2	61 0.0%	280772 56.8%	0 0.0%	0 0.0%	0 0.0%	100.0% 0.0%
3	6 0.0%	0 0.0%	107195 21.7%	0 0.0%	21 0.0%	100.0% 0.0%
4	0 0.0%	0 0.0%	0 0.0%	971 0.2%	0 0.0%	100% 0.0%
5	249 0.1%	0 0.0%	0 0.0%	0 0.0%	7283 1.5%	96.7% 3.3%
	99.7% 0.3%	100.0% 0.0%	100.0% 0.0%	99.2% 0.8%	93.7% 6.3%	99.8% 0.2%
	1	2	3	4	5	
	1	2	3	4	5	

Fig. 15. Confusion Matrix of Cascade Neural Network

Fig. 15 shows the classification result for the Cascade Neural Network using Scaled Conjugate Gradient which is 99.8%. Scaled Conjugate Gradient (SCG) stands out as an advantageous choice for Cascade Neural Networks when compared to other algorithms. Its key strengths lie in efficient learning rate adaptation, accelerating training convergence, and enhancing the network's ability to learn complex data patterns. SCG excels in handling noisy data and exhibits reduced sensitivity to learning rate variations, ensuring stable and effective training, especially in scenarios with uncertain or fluctuating data. The confusion matrix illustrates minimal misclassification in the first four classes. Class five shows a misclassification rate of 6.3%. This indicates that the training algorithm has efficiently learned data patterns compared to Fletcher-Powell Conjugate Gradient algorithms. The accuracy results of this algorithm are better as compared to Resilient Backpropagation. Despite using less training time than the Fletcher-Powell Conjugate Gradient. The training time for this experiment is 13 minutes and 03 seconds.

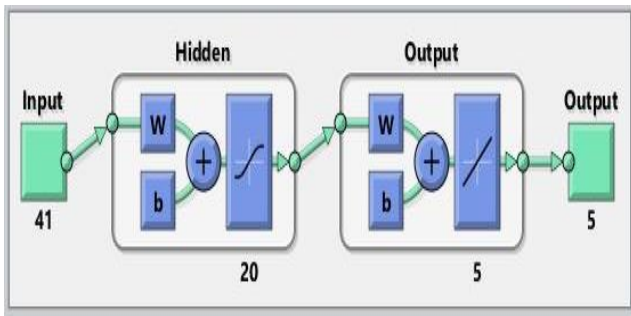


Fig. 16. Fitnet NN with Input, Hidden and Output layers.

Fig. 16 shows the model of FNN. This model encompasses a total of 41 input components, complemented by a hidden layer containing 20 neurons and an output layer with 5 neurons.

9.7 Results of the Fitnet Neural Network using Resilient Backpropagation

Output Class	1	2	3	4	5	
1	97031 19.6%	35 0.0%	10 0.0%	8 0.0%	2348 0.5%	97.6% 2.4%
2	15 0.0%	280755 56.8%	0 0.0%	0 0.0%	5 0.0%	100.0% 0.0%
3	5 0.0%	0 0.0%	107191 21.7%	0 0.0%	50 0.0%	99.9% 0.1%
4	11 0.0%	0 0.0%	0 0.0%	970 0.2%	69 0.0%	92.4% 7.6%
5	216 0.0%	0 0.0%	0 0.0%	1 0.0%	5301 1.1%	96.1% 3.9%
	1	2	3	4	5	99.4% 0.6%

Fig. 17. Confusion Matrix of Fitnet Neural Network

Fig. 17 shows the classification result for Fitnet Neural Network using Resilient Back-propagation Network; the overall accuracy stands at 99.4%. In this experimental result, the Cascade Neural Network utilizes the sigmoid activation function in its hidden layers, recognized for its non-linear characteristics. Resilient Backpropagation proves beneficial for the Fitnet Neural Network due to its adept weight adjustment during training. The algorithm's adaptive nature speeds up convergence, facilitating rapid learning and adaptation to obscure data patterns. Resilient Backpropagation excels in handling noisy data and shows reduced sensitivity to learning rate choices. Its effectiveness in adjusting weights based on gradient information enhances training process stability. In essence, its adaptability aligns seamlessly with the dynamic characteristics of Fitnet Neural Networks, making it a fitting optimization algorithm for this particular neural network type. The confusion matrix demonstrates minimal misclassification in the first four classes, while the misclassification rate for the fifth class is 31.8%. The overall impact of this misclassification is limited, resulting in an overall misclassification rate of only 0.6%. The total training duration for this experiment was five minutes and thirty-five seconds, which is less than Feed Forward Neural Network and Cascade Neural Network.

9.8 Results of the Fitnet Neural Network using Fletcher-Powell Conjugate Gradient

Output Class	1	2	3	4	5	
1	97012 19.6%	29 0.0%	7 0.0%	8 0.0%	571 0.1%	99.4% 0.6%
2	44 0.0%	280761 56.8%	0 0.0%	0 0.0%	0 0.0%	100.0% 0.0%
3	4 0.0%	0 0.0%	107191 21.7%	0 0.0%	12 0.0%	100.0% 0.0%
4	1 0.0%	0 0.0%	1 0.0%	971 0.2%	6 0.0%	99.2% 0.8%
5	217 0.0%	0 0.0%	2 0.0%	0 0.0%	7184 1.5%	97.0% 3.0%
	1	2	3	4	5	99.8% 0.2%

Fig. 18. Confusion Matrix of Fitnet Neural Network

Fig. 18 shows the classification result for the Fitnet Neural Network using Fletcher-Powell Conjugate Gradient; the overall accuracy stands at 99.8%.

Fletcher-Powell Conjugate Gradient aids Fitnet Neural Networks by providing effective optimization during the training process, enhancing their ability to learn complex patterns in the data. The confusion matrix indicates minimal misclassification in the first four classes, while 7.6% misclassification was recorded in class five. However, it necessitates more training time, taking a total of 16 minutes and 16 seconds for this experiment, which is less than Feed Forward Neural Network and Cascade Neural Network.

9.9 Results of the Fitnet Neural Network using Scaled Conjugate Gradient

Output Class	1	2	3	4	5	Accuracy
1	97015 19.6%	18 0.0%	2 0.0%	1 0.0%	526 0.1%	99.4% 0.6%
2	61 0.0%	280772 56.8%	0 0.0%	0 0.0%	0 0.0%	100.0% 0.0%
3	12 0.0%	0 0.0%	107197 21.7%	1 0.0%	25 0.0%	100.0% 0.0%
4	2 0.0%	0 0.0%	1 0.0%	977 0.2%	96 0.0%	90.8% 9.2%
5	188 0.0%	0 0.0%	1 0.0%	0 0.0%	7126 1.4%	97.4% 2.6%
Target Class	1	2	3	4	5	Accuracy
	99.7% 0.3%	100.0% 0.0%	100.0% 0.0%	99.8% 0.2%	91.7% 8.3%	99.8% 0.2%

Fig. 19. Confusion Matrix of Fitnet Neural Network

Fig. 19 shows the classification result for Fitnet Neural Network using Scaled Conjugate Gradient, the overall accuracy stands at 99.8%. Scaled Conjugate Gradient benefits Fitnet Neural Networks by providing effective optimization during training, facilitating enhanced learning of complex data patterns while ensuring efficiency in convergence and reducing sensitivity to hyper-parameter choices. The confusion matrix indicates minimal misclassification in the first four classes, while 8.3% misclassification was recorded in class five. However, it necessitates more training time, taking a total of five minutes and 35 seconds for this experiment, which is less than Feed Forward Neural Network and Cascade Neural Network.

The following Table 13 shows the training time and accuracy of Neural Networks and algorithms.

Table 3

Training time and accuracy of neural networks

Sr. No.	NBN	Algorithm	Accuracy	Training Time in seconds (s)
1	Feed Forward Neural Network	Resilient Backpropagation	99.4%	0.05:46
2		Fletcher-Powell Conjugate Gradient	99.2%	0.21:41
3		Scaled Conjugate Gradient	99.9%	0.08:25
4	Case Cascade Neural Network	Resilient Backpropagation	99.2%	0.06:27
5		Fletcher-Powell Conjugate Gradient	99.8%	0.24:09
6		Scaled Conjugate Gradient	99.8%	0.13:03
7	Fit Net Neural Network	Resilient Backpropagation	99.4%	0.05:35
8		Fletcher-Powell Conjugate Gradient	99.8%	0.06:16
9		Scaled Conjugate Gradient	99.8%	00.5:35

Table 3 summarizes the performance of three ANN models trained with three optimization algorithms. Fitnet Neural Network showed consistently high accuracy (99.8%) with comparatively lower training time, particularly with the Scaled Conjugate Gradient method.

10. Conclusion and Future Work

This study evaluated the performance of three training algorithms, Resilient Backpropagation, Fletcher-Powell Conjugate Gradient, and Scaled Conjugate Gradient, applied to three types of Neural-Based Networks: FFNN, CFFNN, and FNN, for DDoS attack detection. Experimental results showed that the FNN consistently outperformed the other models in terms of accuracy and training time across all algorithms.

Though the results also revealed a consistent drop in performance for the fifth class, which includes miscellaneous attack types. This is mainly due to its heterogeneous nature, as it combines multiple rare attack types with differing patterns. Additionally, class imbalance and limited training samples hinder the model's ability to learn distinct features for this group, resulting in higher misclassification rates compared to dominant classes such as Smurf and Neptune. These findings emphasize the importance of selecting suitable ANN architectures and training strategies for effective DDoS detection. In future work, we plan to investigate more advanced neural network models, including deep learning techniques, and apply them to modern benchmark datasets such as CICIDS2017 and UNSW-NB15. Additionally, we aim to experiment with varying network depth and neuron configurations to enhance performance in real-time detection environments.

11. References

- [1] G. Dayanandam, T. V. Rao, D. Bujji Babu, and S. Nalini Durga, "DDoS Attacks—Analysis and Prevention," *Lecture Notes in Networks and Systems*, vol. 32, pp. 1–10, 2019, doi: 10.1007/978-981-10-8201-6_1.
- [2] R. F. Fouladi, C. E. Kayatas, and E. Anarim, "Frequency-based DDoS attack detection approach using naive bayes classification," 2016 39th International Conference on Telecommunications and Signal Processing, TSP 2016, pp. 104–107, Nov. 2016, doi: 10.1109/TSP.2016.7760838.
- [3] C. J. Hsieh and T. Y. Chan, "Detection of DDoS attacks based on neural network using Apache Spark," 2016 International Conference on Applied System Innovation, IEEE ICASI 2016, Aug. 2016, doi: 10.1109/ICASI.2016.7539833.
- [4] Z. Ren, X. Liu, R. Ye, and T. Zhang, "Security and privacy on internet of things," *Proceedings of 2017 IEEE 7th International Conference on Electronics Information and Emergency Communication, ICEIEC 2017*, pp. 140–144, Oct. 2017, doi: 10.1109/ICEIEC.2017.8076530.
- [5] G. Ramadhan, Y. Kurniawan, and Chang-Soo Kim, "Design of TCP SYN Flood DDoS attack detection using artificial immune systems," pp. 72–76, Feb. 2017, doi: 10.1109/ICSNGT.2016.7849626.
- [6] M. Leon, T. Markovic, and S. Punnekkat, "Comparative Evaluation of Machine Learning Algorithms for Network Intrusion Detection and Attack Classification," *Proceedings of the International Joint Conference on Neural Networks*, vol. 2022-July, 2022, doi: 10.1109/IJCNN55064.2022.9892293.
- [7] R. Vishwakarma and A. K. Jain, "A survey of DDoS attacking techniques and defence mechanisms in the IoT network," *Telecommun Syst*, vol. 73, no. 1, pp. 3–25, Jan. 2020, doi: 10.1007/S11235-019-00599-Z/METRICS.
- [8] F. Kargl, J. Maier, and M. Weber, "Protecting web servers from distributed denial of service attacks," *Proceedings of the 10th International Conference on World Wide Web, WWW 2001*, pp. 514–524, Apr. 2001, doi: 10.1145/371920.372148.
- [9] M. Sachdeva, G. Singh, K. Kumar, and K. Singh, "DDoS Incidents and their Impact: A Review.," *Int. Arab J. Inf. Technol.*, vol. 7, no. 1, pp. 14–20, 2010.
- [10] J. E. Dayhoff and J. M. DeLeo, "Artificial neural networks: opening the black box," *Cancer: Interdisciplinary International Journal of the American Cancer Society*, vol. 91, no. S8, pp. 1615–1635, 2001.
- [11] A. D. Dongare, R. R. Kharde, and A. D. Kachare, "Introduction to artificial neural network," *International Journal of Engineering and Innovative Technology (IJEIT)*, vol. 2, no. 1, pp. 189–194, 2012.
- [12] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, 2018.
- [13] D. Peraković, M. Periša, ... I. C.-2016 24th, and undefined 2016, "Artificial neuron network implementation in detection and classification of DDoS traffic," *ieeexplore.ieee.org* D Peraković, M Periša, I Cvitić, S Husnjak 2016 24th telecommunications forum (TELFOR), 2016•ieeexplore.ieee.org, doi: 10.1109/TELFOR.2016.7818791.
- [14] T. Radivilova, L. Kirichenko, D. V. Ageyev, D. Ageyev, and V. Bulakh, "Classification methods of machine learning to detect DDoS attacks," *ieeexplore.ieee.org* T Radivilova, L Kirichenko, D Ageyev, V Bulakh 2019 10th IEEE International Conference on Intelligent Data, 2019•ieeexplore.ieee.org, doi: 10.1109/IDAACS.2019.8924406.
- [15] O. Ali, P. C.-2018 9th I. A. U. Computing, and undefined 2018, "Towards DoS/DDoS attack

- detection using artificial neural networks,” *ieeexplore.ieee.org* O Ali, P Cotae 2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile, 2018•*ieeexplore.ieee.org*, Accessed: May 05, 2024. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/8796637/>
- [16] H. Bharadwaj, V. K.-P. computer science, and undefined 2018, “Comparative study of neural networks in path planning for catering robots,” Elsevier H Bharadwaj, V Kumar *Procedia computer science*, 2018•Elsevier, Accessed: May 05, 2024. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1877050918309967>
- [17] R. Fletcher, C. R.-T. computer journal, and undefined 1964, “Function minimization by conjugate gradients,” *academic.oup.com* R Fletcher, CM Reeves *The computer journal*, 1964•*academic.oup.com*, 2024, Accessed: May 05, 2024. [Online]. Available: <https://academic.oup.com/comjnl/article-abstract/7/2/149/335311>
- [18] C. Arthur, V. Temeng, Y. Z.-G. M. Journal, and undefined 2020, “Performance evaluation of training algorithms in backpropagation neural network approach to blast-induced ground vibration prediction,” *ajol.info* CK Arthur, VA Temeng, YY Ziggah *Ghana Mining Journal*, 2020•*ajol.info*, vol. 20, no. 1, pp. 20–33, 2020, doi: 10.4314/gm.v20i1.3.
- [19] M. M.-N. networks and undefined 1993, “A scaled conjugate gradient algorithm for fast supervised learning,” Elsevier MF Møller *Neural networks*, 1993•Elsevier, Accessed: May 05, 2024. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0893608005800565>
- [20] B. Gupta, P. Chaudhary, X. Chang, N. N.-C. & Electrical, and undefined 2022, “Smart defense against distributed Denial of service attack in IoT networks using supervised learning classifiers,” Elsevier BB Gupta, P Chaudhary, X Chang, N Nedjah *Computers & Electrical Engineering*, 2022•Elsevier, Accessed: May 05, 2024. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0045790622000404>
- [21] I. Cvitić, D. Peraković, M. Periša, and B. Gupta, “Ensemble machine learning approach for classification of IoT devices in smart home,” *International Journal of Machine Learning and Cybernetics*, vol. 12, no. 11, pp. 3179–3202, Nov. 2021, doi: 10.1007/S13042-020-01241-0.
- [22] M. Ge, X. Fu, N. Syed, Z. Baig, G. Teo, and A. Robles-Kelly, “Deep learning-based intrusion detection for IoT networks,” *ieeexplore.ieee.org* M Ge, X Fu, N Syed, Z Baig, G Teo, A Robles-Kelly 2019 IEEE 24th pacific rim international symposium on dependable, 2019•*ieeexplore.ieee.org*, doi: 10.1109/PRDC47002.2019.00056.
- [23] G. Ajeetha and G. Madhu Priya, “Machine Learning Based DDoS Attack Detection,” 2019 *Innovations in Power and Advanced Computing Technologies, i-PACT 2019*, Mar. 2019, doi: 10.1109/I-PACT44901.2019.8959961.
- [24] B. NG, S. S.- Neurocomputing, and undefined 2019, “Deep radial intelligence with cumulative incarnation approach for detecting denial of service attacks,” Elsevier BA NG, S Selvakumar *Neurocomputing*, 2019•Elsevier, Accessed: May 05, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231219302711?casa_token=alPgRQWTbUAAAAA:4AloApnd0ZUPNx2rt5HJU0MXU6D-GfcRejkeJi9lyWf-yvf_fsLSpM08pLEoApwXWLraVu9TU_M
- [25] O. Elejla, B. Belaton, M. Anbar, ... B. A.-... S. and T., and undefined 2019, “Comparison of classification algorithms on ICMPv6-based DDoS attacks detection,” Springer OE Elejla, B Belaton, M Anbar, B Alabsi, AK Al-Ani *Computational Science and Technology: 5th ICCST 2018*, Kota Kinabalu, Malaysia, 2019•Springer, vol. 481, pp. 29–30, 2018, doi: 10.1007/978-981-13-2622-6_34.
- [26] S. Wankhede, D. K.-2018 F. International, and undefined 2018, “DoS attack detection using machine learning and neural network,” *ieeexplore.ieee.org* S Wankhede, D Kshirsagar 2018 Fourth International Conference on Computing Communication, 2018•*ieeexplore.ieee.org*, Accessed: May 05, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8697702/?casa_token=TxldOs-hZkAAAAA:Y5Kf4VHVNNIc-c6KZp7BAX9o8tR_OM9zLeVKMdrStsWlXL F3FCMYZOa32YdeujidAYoMffXXwk
- [27] C. She, W. Wen, Z. Lin, and K. Zheng, “Application-layer ddos detection based on a one-class support vector machine,”

- [28] X. Yuan, C. Li, X. L.-2017 I. international conference on, and undefined 2017, “DeepDefense: identifying DDoS attack via deep learning,” ieeexplore.ieee.orgX Yuan, C Li, X Li2017 IEEE international conference on smart computing (SMARTCOMP), 2017•ieeexplore.ieee.org, Accessed: May 05, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7946998/?casa_token=4Uj2PG36d6YAAAAA:SZPJb2JVKC0jhcpIgzA7ARrU5lkooR8ii6u_x4MKc5IsiGQU94EPXU6t4HYFbBbWxoBqttt2Aq4
- [29] E. Hodo, X. Bellekens, ... A. H.-2016 I., and undefined 2016, “Threat analysis of IoT networks using artificial neural network intrusion detection system,” ieeexplore.ieee.orgE Hodo, X Bellekens, A Hamilton, PL Dubouilh, E Iorkyase, C Tachtatzis, R Atkinson2016 International Symposium on Networks, Computers and, 2016•ieeexplore.ieee.org, Accessed: May 05, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7746067/?casa_token=A3K4e5Lf0V0AAAAA:R2eVomz3VUMItIHLDaKpDcTu_94JDSQJ2r9rLppnhphM_jJRBmEnSGPBy8IpG7RKWFABlv2XISY