Neural Computing and Applications (2022) 34:2869–2882
https://doi.org/10.1007/s00521-021-06565-8

ORIGINAL ARTICLE

Optimization of vector convolutional deep neural network using binary real cumulative incarnation for detection of distributed denial of service attacks

N. G. Bhuvaneswari Amma[1] · S. Selvakumar[2]

Received: 29 September 2020 / Accepted: 20 September 2021 / Published online: 3 October 2021

Abstract
In today's technological world, distributed denial of service (DDoS) attacks threaten Internet users by flooding huge network traffic to make critical Internet services unavailable to genuine users. Therefore, design of DDoS attack detection system is on urge to mitigate these attacks for protecting the critical services. Nowadays, deep learning techniques are extensively used to detect these attacks. The existing deep feature learning approaches face the lacuna of designing an appropriate deep neural network structure for detection of DDoS attacks which leads to poor performance in terms of accuracy and false alarm. In this article, a tuned vector convolutional deep neural network (TVCDNN) is proposed by optimizing the structure and parameters of the deep neural network using binary and real cumulative incarnation (CuI), respectively. The CuI is a genetic-based optimization technique which optimizes the tuning process by providing values generated from best-fit parents. The TVCDNN is tested with publicly available benchmark network traffic datasets and compared with existing classifiers and optimization techniques. It is evident that the proposed optimization approach yields promising results compared to the existing optimization techniques. Further, the proposed approach achieves significant improvement in performance over the state-of-the-art attack detection systems.

Keywords Convolutional neural network · Cumulative incarnation · Deep learning · DDoS attacks · Neural network tuning · Optimization

1 Introduction

Denial of service (DoS) attacks exploit today's Internet infrastructure to a huge extent, and identification of these attacks is challenging to Internet Service Providers. These attacks consume huge network traffic and server resources thereby denying services to legitimate users [1]. Nowadays, these attacks are distributive in nature, and hence, these attacks are named as distributed denial of service (DDoS) attacks. The intention of the attackers is to overwhelm the resources with useless packets for denying legitimate users to access the target system or services. These attacks are launched in the higher layers such as network layer, transport layer, and application layer [2]. The reason for these attacks is the enormous amount of attack tools available in the Internet. Even a novice can launch such attacks with the available tools. The DDoS attack packets do not show any specific characteristics that distinguish malicious traffic from legitimate traffic. According to Kaspersky Lab DDoS Intelligence Report, DDoS attacks are the most dominant threats registered in 79 countries in most of the organizations. Further, the number and complexity of these attacks are on the rise which necessitate identification of these attacks [3].

In order to identify the class of the attack, the network traffic patterns have to be learned using machine learning approaches. The existing DDoS attack detection systems

✉ N. G. Bhuvaneswari Amma
bhuvaneswari@iiitu.ac.in

S. Selvakumar
ssk@nitt.edu; director@iiitu.ac.in

1 School of Computing, Indian Institute of Information Technology, Una, Himachal Pradesh 177 005, India

2 Department of Computer Science and Engineering, National Institute of Technology, Tiruchirappalli and Indian Institute of Information Technology, Una, Himachal Pradesh 177 005, India

that are based on machine learning approaches exhibited poor performance with respect to accuracy and false alarm due to their inability to learn the features in different levels of abstraction [4]. In order to overcome this issue, nowadays, deep learning has been used for pattern recognition and classification [5]. The deep neural networks learn the features layer by layer in a better way to identify unknown attacks [6]. The deep learning techniques used for attack detection include convolutional neural network (CNN), recurrent neural network (RNN), long short-term memory (LSTM), auto-encoder, transfer learning, etc. The CNN automatically and adaptively learns the features using convolutional, pooling, and fully connected layers [2]. The RNN learns by remembering the information based on time [7]. The LSTM is a RNN in which it learns by remembering the information using gates such as input, output, and forget [8]. The auto-encoder is an unsupervised learning technique that learns the data by ignoring the noise in the data [9, 10]. The deep learning techniques can be utilized for transfer learning in which a model trained for solving one problem can be reused for similar problem solving, i.e. it is an improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned [11]. Among these techniques, researchers are mostly attracted towards CNN as it provides automatic feature extraction efficacy and handling of non-linear data in robust manner. The CNN is an extension of the conventional feedforward neural networks. Initially, this technique has been studied for processing two-dimensional or three-dimensional images. Nowadays, this technique is being applied for intrusion detection with one-dimensional network traffic data [12]. The processing is similar to that of higher dimensional but differs in the selection of filters and strides [13].

The success of utilizing CNN for attack detection depends on proper setting of the structure and parameters/connected weights of deep neural network [14]. The number of hidden layers and the neurons in those layers constitute the structure of the neural network. The parameters of the neural network are the connected weights, and random generation of these weights may get the network stuck in the local optima. The issues in the existing deep learning-based DDoS attacks identification systems are the choice of hidden layers and neurons in the hidden layers using trial and error method leading to more training time and inefficient performance [13]. Further, the random choice of weights gets stuck in the local optima leading to poor performance [15]. The weights in the structure of the deep neural network are tuned using optimization algorithms to yield promising results [16].

Evolutionary optimization algorithms such as genetic algorithms (GA) [15], differential evolution (DE) [17], particle swarm optimization (PSO) [16] and ant colony optimization (ACO) are popularly used to find optimal solutions but suffered from long run time and converge early leading to degradation in performance. This problem is effectively captured using CuI optimization [18].

The important challenge in the construction of DDoS attack detection system is the selection and design of an appropriate network structure with suitable parameters for identifying multiple classes. To overcome the challenges in the design of structure and parameters of deep neural network, a method which finds the suitable structure and best set of weights is proposed.

The following propositions are the key contributions of this article:

1. Binary cumulative incarnation (BCuI) approach and its algorithm to tune the structure of the vector convolutional deep neural network (VCDNN).
2. Real cumulative incarnation (RCuI) approach and its algorithm to tune the weights of the VCDNN. The CuI approach proposed by us in [18] has been utilized for tuning the weights of VCDNN; but the structure of the VCDNN is different as compared to the structure of the neural network in [18].
3. TVCDNN-based DDoS attacks identification algorithm is designed to identify the class of the network traffic.

The rest of the article is organized as follows: Sect. 2 discusses the related works in five aspects, viz. DDoS attacks, DDoS attacks detection system, neural network-based classification, deep neural network-based attack detection, and neural network tuning using optimization techniques. Section 3 discusses the proposed TVCDNN approach for DDoS attacks detection. Section 4 discusses the experimental results of the proposed TVCDNN approach for identifying the type of DDoS attacks. Section 5 concludes the article with future research directions.

2 Related works

This section discusses the literature related to the proposed approach.

2.1 DDoS attacks

DDoS attacks deny legitimate users from accessing critical services by exploiting the Internet infrastructure. Due to the vulnerabilities existing in the Internet and the availability of more and more attack tools, the complexity and consequences of these attacks get increased. Table 1 shows the types of DDoS attacks categorized in benchmark datasets [19–21]. Table 2 shows the recent DDoS attacks that happened in the world [3]. It is observed that DDoS attacks

Table 1 Types of DDoS attacks

Attack	Description
Back	URL with many back slashes
Neptune	SYN flood attack on one or more ports
Smurf	Packets directed to IP broadcast addresses remotely
Teardrop	Misfragmented UDP packets cause system reboot
Others	Attacks including land, ping of death, process table, and mail bomb

Table 2 Recent DDoS attacks

Attack victim	Month/Year	Incident
US HHS Dept	Mar. 2020	Deprived citizens from accessing official data about COVID-19 pandemic and measures taken against it
German Food Delivery Service	Mar. 2020	Launched DDoS attack and demanded two bitcoins of about $11,000
Facebook	Mar. 2019	Users were unable to log into their accounts
Philippines National Union of Journalists	Feb. 2019	Website disabled several hours with traffic of 468 Gbps
UAlbany	Feb. 2019	Servers down for five min in the University of Albany
US-based Wired Telecommunication Carrier	Mar. 2018	Traffic of 1.7 terabytes per second
GitHub	Feb. 2018	1.3 Tbps of DoS attacks launched in GitHub code repository
Boston Globe	Nov. 2017	Disrupted Boston Globe newspaper's telephones and interrupted editing system
Electroneum Cryptocurrency	Nov. 2017	Suffered DDoS attack on company's website
UK National Lottery	Sep. 2017	Attack launched on lottery's website and its mobile app
DreamHost	Aug. 2017	Attacked DNS infrastructure in offline mode
Melbourne IT	Apr. 2017	Suffered DDoS attack by forcing the victimized Domain Name Register by denying cloud hosting and mailing platforms to their customers
Dyn	Oct. 2016	Websites crippled by DNS errors
ISPs in Mumbai	Jul. 2016	Experienced huge magnitude of 200 Gbps
BBC	Dec. 2015	BBC sites down for three hours
Dutch Government Sites	Feb. 2015	Central Government's major websites crippled more than seven hours

are happening all over the world and prevailing as one of the most serious cyber attacks.

2.2 DDoS attacks detection system

The mechanisms to construct DDoS attack detection systems are classified into statistical, machine learning, and data mining approaches [22, 23]. The statistical method-based detection mechanisms are based on distance measures, and the knowledge of attack traffic is not required for attack detection, but prior assumptions are needed for better and fast detection [24, 25]. The machine learning-based detection methods create a model by learning the features. Neural network is one of the most popularly used machine learning-based methods which provides better generalization capability but suffers from high computational burden and overfitting [26]. These issues motivated us for the creation of TVCDNN in which the structure and parameters are tuned to design a generalized deep learning structure for DDoS attacks identification.

2.3 Neural network-based classification

The survey of existing neural network-based classification methods is given in [22]. Neural processing techniques [27] such as multi-layer perceptron (MLP), support vector machine (SVM), k-nearest neighbour, and deep learning are used for anomaly detection in recent research [4]. The following are the categories of neural network-based learning methods: supervised learning, unsupervised learning, and semi-supervised learning [28–30]. The supervised learning is a type of machine learning that model the relationship between the input features and the target output such that the new data are predicted based on the learned relationships from previous data. The learning happens with labelled dataset. The class label acts as a boundary to separate normal from attack category. The unsupervised learning models relationship with the input patterns alone and is applied when the human expert does not know the type of pattern in the data. The learning happens with unlabelled dataset. The training set contains only normal traffic and anything that does not belong to this kind of traffic is considered as anomalous. The semi-supervised learning lies between supervised and unsupervised learning. The majority of data have no labels and few have labels. The goal of semi-supervised learning is to compute groups of similar examples within the data or to determine the distribution of data within the input space and is best suited for model building.

2.4 Deep neural network-based attack detection

The most widely used neural processing technique is deep learning that learns the data with different abstraction levels [5, 31, 32]. Deep learning approaches such as CNN, auto-encoder, RNN, and LSTM are used for anomaly detection [33]. These methods are capable of adapting to network environments that change dynamically which motivated us to incorporate deep learning in the proposed approach [34]. Among these, the CNNs handle non-linearity in the data in a more robust manner which is the intuition behind proposing a tuned CNN. Generally, CNNs consist of convolutional layer (CL), pooling layer (PL), and fully connected network with input layer, one or more hidden layer (HL), and output layer [35].

2.5 Neural network-tuning using optimization methods

The challenge in the successful design of deep neural network is to have a generalized structure which improves the performance of the system. The tuning of deep neural network structure based on trial and error is time consuming. Therefore, autonomous tuning is required, and optimization algorithms are suitable for the tuning process [36]. The evolutionary optimization techniques such as GA, DE, Evolution Strategies, PSO, ACO, and simulated annealing suffer from high run time and premature convergence [37, 38]. The advantages and drawbacks of these existing neural network tuning based on optimization methods are tabulated in Table 3.

The hurdles in the utilization of optimization techniques for tuning are representation of individuals using encoding schemes and techniques for run time reduction of evolutionary algorithms. This article proposes approaches for tuning the structure and parameters of the deep neural network using BCuI algorithm and RCuI algorithm, respectively.

3 Proposed tuned vector convolutional deep neural network (TVCDNN) approach

The proposed TVCDNN is a deep neural network in which features are extracted using vector convolution method and the structure and parameters of the learning network are tuned using the proposed BCuI- and RCuI-based optimization method. Figure 1 depicts the block diagram of the proposed TVCDNN approach for DDoS attacks detection which detects and identifies the type of traffic. It consists of two modules: TVCDNN learning and TVCDNN-based DDoS attacks identification. The training is done with the TVCDNN network by constructing the VCDNN with structure and parameters tuning. The testing is done in TVCDNN-based DDoS attack identification module using the test data with the learned weights obtained from the TVCDNN learning module. The boxes represented in bold are the working processes and in dotted are the input and output flow of the computations. The network traffic is obtained from the edge router and the result of computations, viz. normal traffic is forwarded to the machines connected to the network switch and the identified class of attacks is forwarded to attack mitigation system.

3.1 Construction of VCDNN

The VCDNN consists of vector convolutional feature extraction (VCFE) and fully connected neural network

Table 3 Pros and cons of existing neural network-tuning techniques

Ref. no	Methods used	Application	Advantages	Drawbacks
[16]	Particle Swarm Optimization	Smart building occupant prediction	Less usage of computational resources	Manual choice of initial neural network parameters setting
[29]	Taguchi Genetic Algorithm	Sunspot number forecast, Associative memory tuning, and XOR problem classification	Robustness, fast convergence, and statistical soundness	Initial neural network parameters set manually
[14]	Co-operative Binary Real PSO	Sunspot number prediction	Low implementation cost, viz. hardware and processing time	Usage of link switches make the neural network more complicated
[30]	Multiobjective optimization	MNIST and CIFAR-10 image classification	High representation ability	Time consuming as layerwise structure tuning was performed

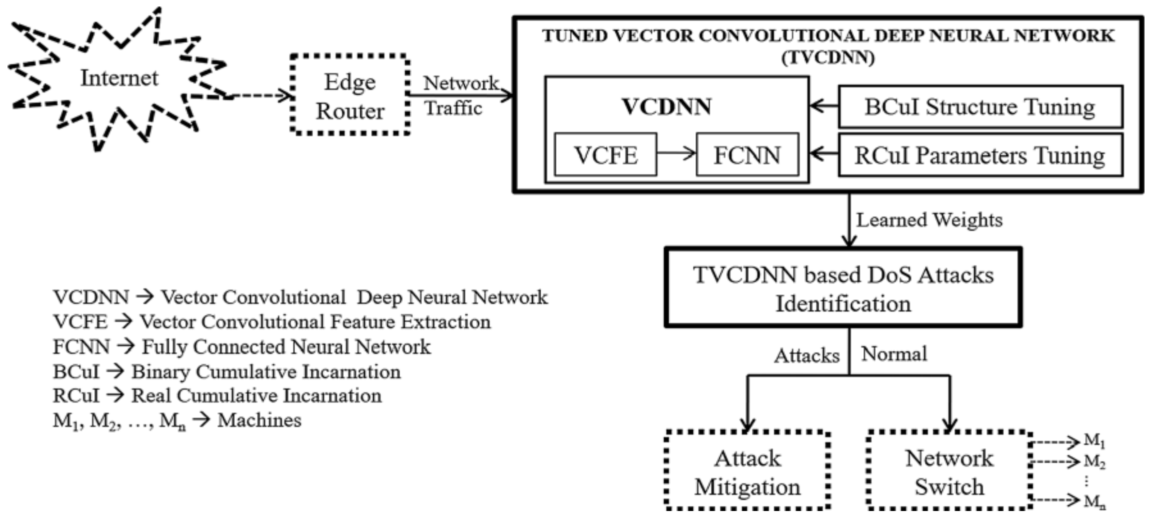

Internet
Edge Router
Network Traffic
TUNED VECTOR CONVOLUTIONAL DEEP NEURAL NETWORK (TVCDNN)
VCDNN
VCFE
FCNN
BCuI Structure Tuning
RCuI Parameters Tuning
Learned Weights
TVCDNN based DoS Attacks Identification
Attacks
Normal
Attack Mitigation
Network Switch
M1
M2
Mn
VCDNN → Vector Convolutional Deep Neural Network
VCFE → Vector Convolutional Feature Extraction
FCNN → Fully Connected Neural Network
BCuI → Binary Cumulative Incarnation
RCuI → Real Cumulative Incarnation
M1, M2, …, Mn → Machines

Fig. 1 Block diagram of proposed TVCDNN approach

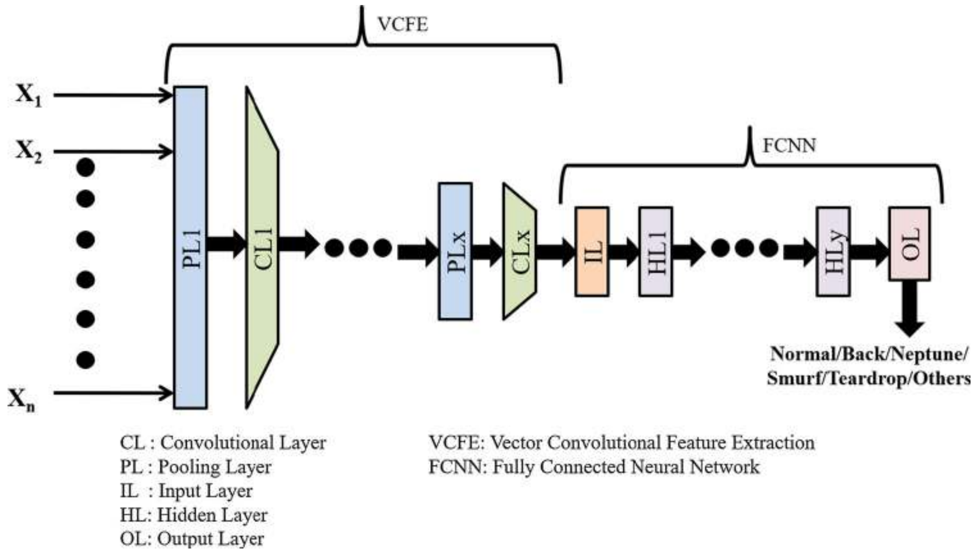


Fig. 2 Structure of vector convolutional deep neural network

(FCNN). Figure 2 depicts the structure of VCDNN. The VCFE comprises of x number of CL and PL. The FCNN comprises of input layer, more than one, i.e., y HLs, and output layer. The CL generates transformed convolution, TrC_i, which is given as input to the PL. The transformed convolution of input vector, X_i, with the filter, Fil, is computed as follows:

$$\mathrm{TrC}_i = X_i \times \mathrm{Fil} \tag{1}$$

where i is the number of records in the dataset.

The PL down samples the feature vector of CL. The max pooling is applied, and the transformed pooling, $TrP()$, is computed as follows:

$$\mathrm{TrP}\left(\mathrm{TrC}_{ij}, \mathrm{TrC}_{ik}\right) = \max\left(\mathrm{TrC}_{ij}, \mathrm{TrC}_{ik}\right) \tag{2}$$

where j is the position of the feature and k = j + 1.

The extracted feature vector is passed to the FCNN training module. The input layer of the FCNN passes the extracted feature vector to the first HL. The computation in the HL is performed by finding the sum of products of feature vector with the weights. It is computed as follows:

$$O_{Hj} = \sum O_{Hj-1} \times W_{ij} \tag{3}$$

where O_{Hj} is the intermediate output of the HL, and W_{ij} is the weights from input layer of FCNN to first HL.

The output of HL is computed using the activation function, Rectified Linear Unit (ReLU), and is as follows:

$$f\left(O_{Hj}\right) = \max\left(O_{Hj}, 0\right) \tag{4}$$

The computations in all the hidden layers are similar to (3) and (4). The computation in the output layer is similar to (3) with the addition of bias term, B, and is computed as follows:

$$O_{Hy} = \sum O_{Hy-1} \times W_{yo} + B \tag{5}$$

where W_{yo} is the weights from last HL of FCNN to the output layer.

The output layer produces the output, Out_i, using SoftMax function, and is computed as follows:

$$\mathrm{Out}_i = \frac{\mathrm{Exp}\left(I_{ji}^{O}\right)}{\sum_{i=0}^{k} \mathrm{Exp}\left(I_{ji}^{O}\right)} \tag{6}$$

The performance of learned network is evaluated using cross-entropy, CE, as follows:

$$\mathrm{CE}(\mathrm{Tar}_i,\ \mathrm{Out}_i) = -\sum_{i=1}^{n} \mathrm{Tar}_i \log(\mathrm{Out}_i) \tag{7}$$

where Tar_i is the target label of each network traffic record in the training dataset.

3.2 Structure tuning using binary cumulative incarnation

The structure of the VCDNN network is tuned using the proposed binary CuI approach. The intuition behind using CuI approach for tuning the structure is that the CuI optimization provides best-fit weights for the subsequent generations. The tuning of CLs, PLs, HLs, and neurons in all the layers is the structure of the deep neural network, and the following sections describe the cumulative incarnation, initial generation of populations, and better offspring generation using BCuI approach.

```
Algorithm 1: BCuI Algorithm

Input: Population Size $p_b$, Number of generations
Output: New population, $p_{bnew}$

1: Generate initial population of chromosomes
2: Compute the fitness value of each chromosome
3: Rank chromosomes based on fitness value
4: Select top 50% of the chromosomes, $top_{bc}$
5: Generate another 50% of the chromosomes, $new_{bc}$ using (9)
6: $p_{bnew} = top_{bc}||new_{bc}$
7: return $Pop_{bnew}$
```

3.2.1 Cumulative incarnation

Figure 3 depicts the flow diagram of CuI approach. The CuI is based on GA [15]. In each iteration, the population is new and different as the new 50% of the population, viz. the second generation, is obtained by calculating the cumulative sum of chromosomes which are best fitted in the previous generation. The termination criterion is either based on the number of generations or by fixing a threshold for the objective function. The objective function of the proposed approach is to maximize the survival of fittest function which is computed as follows:

$$f_{\mathrm{obj}} = {}^{1}\!/_{\mathrm{CE}} \tag{8}$$

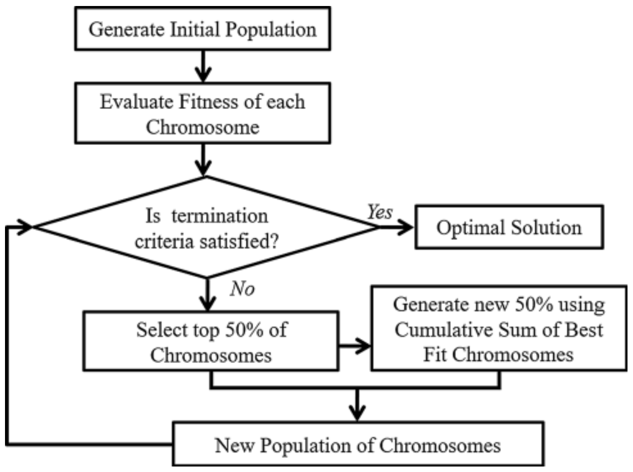

Generate Initial Population
Evaluate Fitness of each Chromosome
Is termination criteria satisfied?
Yes
Optimal Solution
No
Select top 50% of Chromosomes
Generate new 50% using Cumulative Sum of Best Fit Chromosomes
New Population of Chromosomes

Fig. 3 Flow diagram of cumulative incarnation approach

3.2.2 Initial binary population generation

The binary coding technique is used to solve the problem of tuning the CL, PL, HL, and the number of neurons in each layer. In the binary coding scheme, each chromosome is encoded as a vector of binary numbers of size m, where m is the maximum value of each parameter of the structure. The initialization procedure generates p chromosomes, where p is the size of population.

3.2.3 Better binary offspring generation

The better offspring is generated by performing the cumulative binary OR operation on the best fit chromosomes as follows:

$$\begin{aligned} D_l &= \text{OR}(C_1, C_2, \ldots, C_{l+1}) \\ D_p &= \text{OR}(D_1, D_2, \ldots, D_l) \end{aligned} \tag{9}$$

where $OR()$ is a function that returns the value 1 if any or all of its operands is 1 and l ranges from 1 to $p-1$, where p is the size of the population. For example, consider two chromosomes, C_1 and C_2, where $C_1 = (x_1, x_2, \cdots, x_n)$ and $C_2 = (y_1, y_2, \cdots, y_n)$. The new offspring, D_1, is generated by computing $C_1 OR C_2$.

The BCuI algorithm is depicted in Algorithm 1. This algorithm takes the population size, p_b and number of generations as inputs and generates new population of chromosomes, p_{bnew} as output. Here, b represents binary, bc represents chromosomes in binary, and $bnew$ represents new binary offspring. The partial illustration (due to limitation in space) of the generation of number of HLs and neurons in each HL is depicted in Fig. 4. The presence of HL is represented as 1, and the absence of HL is represented as 0. It is seen from the illustration that 3 HLs are generated, each HL with 15, 11, and 8 neurons, respectively. The same method is applicable for generating CL and PL as well.

3.3 Parameters tuning using real cumulative incarnation

The tuning of filters used in CL and weights in HL is one and the same and the following sections describe the initial generation of populations and better offspring generation using RCuI approach.

3.3.1 Initial real population generation

The real coding technique is used to solve the problem of tuning the filters in CL and network weights in fully connected neural network. In the real coding scheme, each chromosome is encoded as a vector of real numbers of size m which includes both positive and negative numbers, where m is the maximum value of each parameter. The initialization procedure generates p chromosomes, where p is the size of population. Each chromosome is divided into

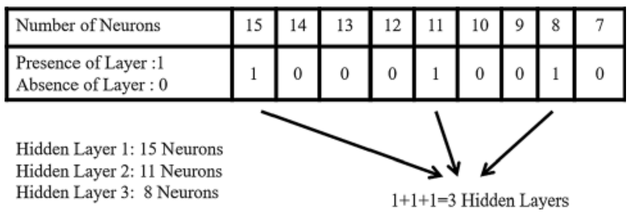


Fig. 4 Illustration of hidden layers and hidden neurons generation

```
Algorithm 2: RCuI Algorithm
Input: Population Size $p_r$, Number of generations
Output: New population, $p_{rnew}$

1: Generate initial population of chromosomes
2: Compute the fitness value of each chromosome
3: Rank chromosomes based on fitness value
4: Select top 50% of the chromosomes, $top_{rc}$
5: Generate another 50% of the chromosomes, $new_{rc}$ using (10)
6: $p_{rnew} = top_{rc} || new_{rc}$
7: return $Pop_{rnew}$
```

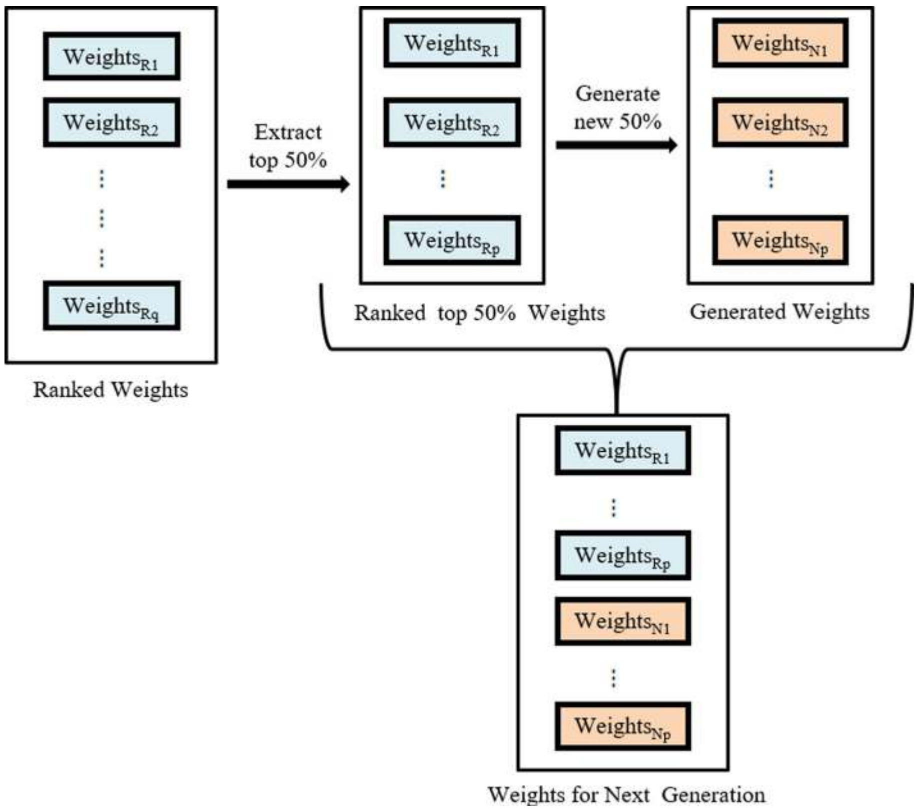


Fig. 5 Illustration of cumulative incarnation of weights

genes, and each gene represents a weight needed for training the VCDNN. Both positive and negative weights are generated and used for experimentation.

3.3.2 Better real offspring generation

The better offspring is generated by performing the cumulative sum operation on the best fit chromosomes as follows:

$$\begin{aligned} Dr_l &= \mu(Cr_1, Cr_2, \ldots, Cr_{l+1}) \\ Dr_p &= \mu(Dr_1, Dr_2, \ldots, Dr_l) \end{aligned} \tag{10}$$

where $\mu()$ is a function that computes the mean of the chromosomes. Consider two chromosomes, Cr_1 and Cr_2, where $Cr_1 = (xr_1, xr_2, \cdots, xr_n)$ and $Cr_2 = (yr_1, yr_2, \cdots, yr_n)$. The new offspring Dr_1 is generated as $\mu(Cr_1, Cr_2)$.

Algorithm 2 depicts the RCuI algorithm which takes the population size, p_r and number of generations as inputs and generates new population, p_{rnew} as output. Here, r represents real, rc represents chromosomes in real, and $rnew$ represents new real offspring.

Figure 5 illustrates the generation of weights using cumulative incarnation [18]. The objective function computed using (8) denoted as Weights$_{Ri}$ $(1 \leq i \leq q)$ is used to rank the VCDNN weights. The top ranked 50% of weights from the previous generation, R_1 to R_p, are selected for elitism in the next generation. The incarnated 50% of weights are calculated from these best fit weights, R_1 to R_p. The $(p+1)^{th}$ individual of weights, Weights$_{N1}$, is the mean of weights R_1 and R_2. The $(p+2)^{th}$ individual of weights is the mean of R_1, R_2, and R_3. Likewise, the $(q-1)^{th}$ individual of weights is the mean of R_1, R_2, $\cdots$, R_{p-1}, and R_p. The q^{th} individual of

Algorithm 3: TVCDNN based DDoS Attacks Identification Algorithm

```
Input: Test data, TVCDNN structure, and learned weights
Output: $Normal/Back/Neptune/Smurf/Teardrop/Others$

1: Let $k$ be the number of CL and PL fine tuned using BCuI
2: for $i = 1$ to $k$ do
3:     Compute transformed convolution, $TrC_i$ using (1)
4:     Compute transformed pooling, $TrP_i$ using (2)
5: end for
6: Let $h$ be the number of hidden layers fine tuned using BCuI
7: for $j = 1$ to $h$ do
8:     Compute $O_{Hj} = \sum O_{Hj-1} \times W_{ij}$ {$W_{ij}$ – learned weights tuned using RCuI}
9:     Compute ReLU activation function, $f(O_{Hj}) = max(O_{Hj}, 0)$
10: end for
11: Perform computation in output layer using (5)
12: Compute $Out_i$ using (6)
13: Convert output to vector
14: Perform mask operation with vector [1 1 1 1 1 1]
15: if $Out_1$ is 1 then
16:     Test data is identified as 'Normal', return $Normal$
17: else if $Out_2$ is 1 then
18:     Test data is identified as 'Back', return $Back$
19: else if $Out_3$ is 1 then
20:     Test data is identified as 'Neptune', return $Neptune$
21: else if $Out_4$ is 1 then
22:     Test data is identified as 'Smurf', return $Smurf$
23: else if $Out_5$ is 1 then
24:     Test data is identified as 'Teardrop', return $Teardrop$
25: else
26:     Test data is identified as 'Others', return $Others$
27: end if
```

weights, Weights_{Np}, is the mean of individual $p+1$ to individual $q-1$. This method of weight generation produces optimized weights leading to better results and hence used for optimizing the tuning process.

3.4 TVCDNN-based DDoS attacks identification

Algorithm 3 depicts the TVCDNN-based DDoS attacks identification algorithm which takes test data, TVCDNN structure, and learned weights as inputs and outputs any one of the identified class labels. The VCDNN is constructed for the test data with the tuned CL, PL, HL, and number of neurons in each layer, and the learned weights using RCuI are assigned to the network. The features are extracted using VCFE, and the class label of the given test data is obtained from FCNN.

4 Experimental results

The proposed approach was implemented in MATLAB R2020b on Intel®Core TM2 Quad CPU Q9650@3.00 GHz processor with GPU NVIDIA and 16 GB RAM under Windows 10. The benchmark datasets such as KDD Cup [20] and NSL KDD [21] datasets were used to evaluate the performance of the proposed TVCDNN for DDoS attacks identification. The statistics of the benchmark network traffic datasets are tabulated in Table 4. The NSL KDD dataset is the refined version of the KDD Cup dataset. The training dataset of KDD Cup consists of redundant and duplicate records. In order to overcome the training results biased to some classes, these records have been removed in the refined version [20]. Table 5 shows the parameter initialization ranges. The maximum number of HLs is half of the input features and being a deep neural network, and the minimum is chosen as more than one HL.

4.1 Performance analysis

The performance analysis of the proposed approach is based on the following measures:

- True Negative (TN): Normal traffic is classified as Normal
- False Positive (FP): Normal traffic is classified as Attack
- False Negative (FN): Attack traffic is classified as Normal

Table 4 Statistics of datasets

Dataset		Normal	Back	Neptune	Smurf	Teardrop	Others
KDD Cup	Training	97,278	2203	107,201	280,790	979	285
	Testing	60,593	1098	58,001	164,091	12	5855
NSL KDD	Training	13,449	196	8282	529	188	39
	Testing	2152	359	1579	627	12	1026

Table 5 Parameter initialization ranges

Parameter	Minimum	Maximum
Number of CL and PL	1	5
Number of Filters	2	5
Number of HL	2	14
Number of Neurons in HL	7	20

- True Positive (TP): Attack traffic is classified as Attack

The obtained confusion matrices for the two benchmark datasets, viz. KDD Cup and NSL KDD on applying the proposed approach are tabulated in Tables 6 and 7, respectively. The optimized structure of the TVCDNN consists of three levels of CL and PL with the filter of size 3. The number of hidden layers in the FCNN are four, and the number of neurons in each hidden layer are 16, 12, 9, and 7 for the first, second, third, and fourth hidden layers, respectively.

The performance metrics such as accuracy, precision, and error rate / false alarm are computed for measuring the performance of TVCDNN using (11), (12), and (13) and are tabulated in Table 8.

$$\text{Accuracy} = \frac{\text{TP}}{\text{TP}+\text{FP}} \times 100 \tag{11}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} \times 100 \tag{12}$$

$$\text{ErrorRate} = \frac{\text{TP}}{\text{TP}+\text{FP}} \times 100 \tag{13}$$

Table 6 Confusion matrix of proposed TVCDNN approach for KDD Cup dataset

Traffic	Normal	Back	Neptune	Smurf	Teardrop	Others
Normal	59,186	536	442	137	101	191
Back	7	1002	29	28	14	18
Neptune	3	21	57,921	19	11	26
Smurf	142	256	194	163,089	171	239
Teardrop	0	0	0	0	12	0
Others	14	19	21	8	7	5786

Table 7 Confusion matrix of proposed TVCDNN approach for NSL KDD dataset

Traffic	Normal	Back	Neptune	Smurf	Teardrop	Others
Normal	2146	3	1	0	0	2
Back	0	353	3	0	0	3
Neptune	4	2	1569	0	0	4
Smurf	2	0	0	624	0	1
Teardrop	0	0	0	0	12	0
Others	21	24	39	17	4	921

4.2 Comparison with base classifiers

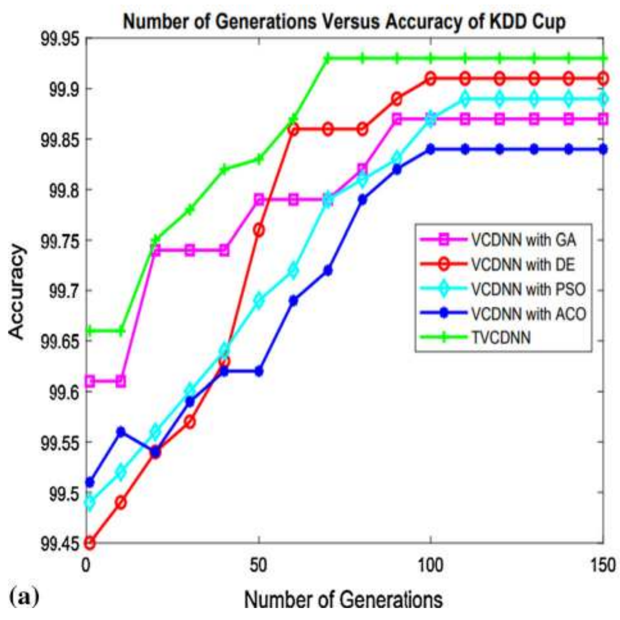
The performance of the proposed TVCDNN is compared with *VCDeepFL* [13] and base classifiers for multi-class classification, namely MLP and SVM. The structure of MLP used for experimentation is $n-10-9-8-7-6$ structure and the activation functions, tanh and sigmoid have been used for hidden layers and output layer, respectively. Table 8 tabulates the performance metrics of the classifiers. It is evident that TVCDNN provides significant improvement over existing multi-class classifiers. The achievement of 100% accuracy is obtained on Teardrop attack class using the proposed approach for both the datasets.

4.3 Comparison with existing optimization algorithms

The proposed TVCDNN approach is compared with existing optimization algorithms, viz. GA, DE, PSO, and ACO. The tuning of VCDNN structure with the proposed BCuI and RCuI optimization is compared with tuning of VCDNN using the existing optimization methods. The initial population size of all the optimization algorithms is 100. The dataset is trained using TVCDNN network up to 150^{th} generation and stopped as the performance was not improved after 100^{th} generation. Figure 6a and b depicts the accuracy of the proposed approach with varying generations for KDD Cup and NSL KDD datasets, respectively. It is observed that the performance of the attack

Table 8 Comparison of proposed approach with existing multi-class classifiers

Approach	Dataset	Metrics	Normal	Back	Neptune	Smurf	Teardrop	Others
TVCDNN	KDD Cup	Accuracy	**97.68**	**91.25**	**99.86**	**99.39**	**100**	**98.82**
		Precision	**99.72**	**54.63**	**98.83**	**99.88**	**75**	**92.40**
		Error Rate	**2.32**	**8.74**	**0.14**	**0.36**	**0**	**1.18**
	NSL KDD	Accuracy	**99.72**	**98.33**	**99.37**	**99.52**	**100**	**89.77**
		Precision	**98.76**	**92.41**	**97.33**	**97.35**	**75**	**98.93**
		Error rate	**0.28**	**1.67**	**0.63**	**0.48**	**0**	**10.23**
VCDeepFL	KDD Cup	Accuracy	97.58	88.89	99.60	99.22	83.33	97.40
		Precision	99.71	68.35	98.99	99.77	2.73	81.27
		Error rate	2.42	11.11	0.40	0.33	16.67	2.60
	NSL KDD	Accuracy	99.63	95.90	97.93	91.07	19.61	97.81
		Precision	99.26	97.77	99.05	99.20	83.33	87.13
		Error rate	0.37	4.10	2.07	8.93	80.39	2.19
MLP	KDD Cup	Accuracy	97.17	82.24	98.26	98.94	58.33	87.60
		Precision	98.08	56.97	98.21	99.49	1.20	82.01
		Error rate	2.83	17.76	1.74	0.74	41.67	12.40
	NSL KDD	Accuracy	99.12	86.5	95.59	90.83	22.92	97.5
		Precision	98.93	96.38	98.80	99.52	91.67	79.82
		Error rate	0.88	13.5	4.41	9.17	0.77	2.5
SVM	KDD Cup	Accuracy	96.76	79.51	98.46	99.18	75	89.91
		Precision	98.66	50.06	98.11	99.60	2.91	80.05
		Error rate	3.24	20.49	1.54	0.53	25	10.09
	NSL KDD	Accuracy	99.16	83.66	95.05	91.16	44.44	96.98
		Precision	99.02	95.54	98.54	98.72	66.67	81.48
		Error rate	0.84	16.34	4.95	8.84	0.56	3.02



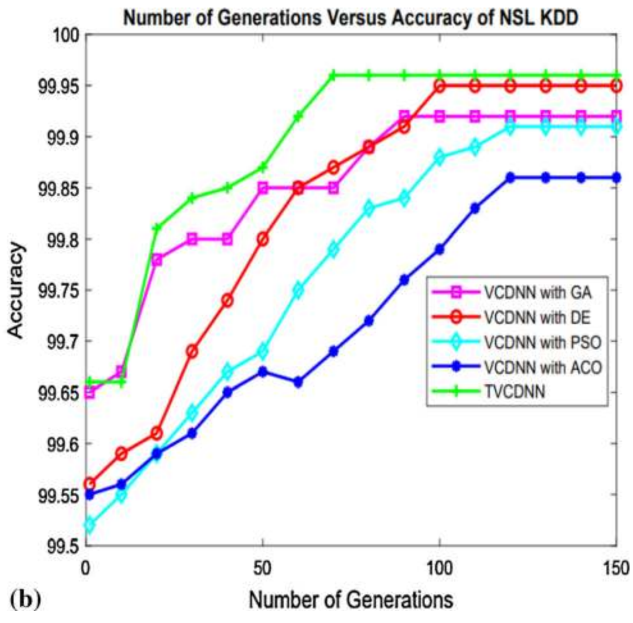


Fig. 6 Accuracy comparisons for varying number of generations **a** KDD Cup **b** NSL KDD

identification system with respect to accuracy was improved with the increase in the number of generations. Figure 7a and b depicts the error rate of the proposed approach with varying generations for KDD Cup and NSL KDD datasets, respectively, and it is seen that the error rate tends to decrease around 70^{th} generations for the proposed

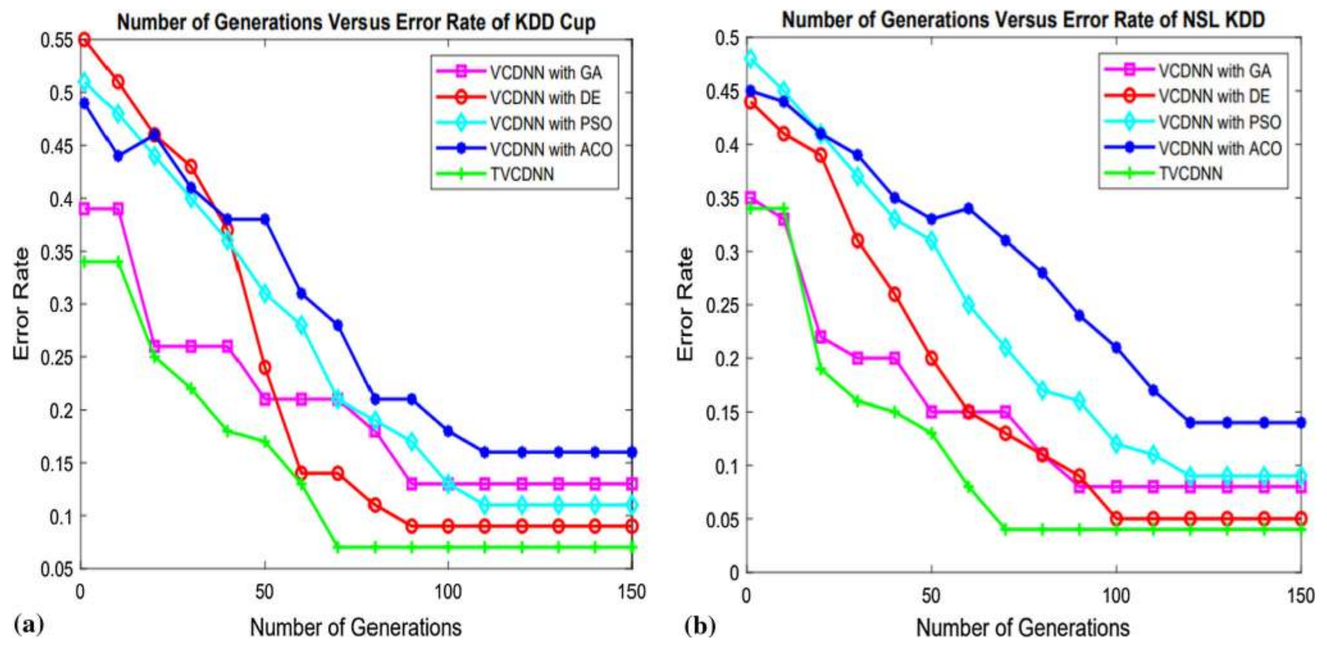


Fig. 7 Error rate comparisons for varying number of generations **a** KDD Cup **b** NSL KDD

approach. The convergence of the TVCDNN approach has been obtained around 80th and 70th generations and gets saturated for KDD Cup and NSL KDD datasets, respectively. The VCDNN with GA approach has been converged while reaching 90th generations for both the datasets and the VCDNN with DE, PSO, and ACO converges beyond 100th generations and gets saturated for both the datasets. It is observed that as the number of generations is less, the learning time is fast. Therefore, it is evident that TVCDNN converges earlier compared to tuning of VCDNN with GA, DE, PSO, and ACO for both the datasets. Further, the learning time is less, and there is gradual convergence using the proposed TVCDNN approach compared to tuning the VCDNN using the existing optimization techniques.

4.4 State-of-the-art comparison

The proposed approach is compared with the state-of-the-art deep neural network approaches for attack detection, and it is observed that the proposed approach yields promising results. Table 9 tabulates the state-of-the-art comparison for the performance metrics, accuracy and error rate with NSL KDD dataset. It is noticed that the performance of proposed approach for identifying other categories of attacks exhibits drastic improvement, and hence, the proposed approach is suitable for identifying unknown attacks. Figure 8a and b depicts the comparison of deep neural network with tuning and without tuning for the performance metrics, viz. accuracy and error rate, respectively, for the benchmark datasets. It is seen that the VCDNN with tuning provides promising results compared to VCDNN without tuning. The NSL KDD dataset provides better results compared to KDD Cup dataset with and without tuning as NSL KDD does not contain redundant and duplicate records. Moreover, the reason for significant improvement with tuning is that both the structure and parameters are tuned using CuI approach which only uses best fit positive and negative weights. The significance of positive and negative weights in tuning VCDNN is to represent an excitatory and inhibitory connection, respectively. Generally, the nodes of the network correspond to the excitatory neurons of the brain. But the learning

Table 9 State-of-art comparison of NSL KDD dataset with respect to accuracy and error rate

Approach	Metrics	Normal	Back	Neptune	Smurf	Teardrop	Others
TVCDNN	Accuracy	**99.72**	**98.33**	**99.37**	**99.52**	**100**	**98.82**
	Error rate	**0.28**	**1.67**	**0.63**	**0.48**	**0**	**1.18**
Reported Results [13]	Accuracy	99.63	95.90	97.93	91.07	19.61	97.81
	Error rate	0.37	4.10	2.07	8.93	80.39	2.19
Reported Results [33]	Accuracy	97.91	36.77	98.05	99.10	**100**	–
	Error Rate	2.09	63.23	1.95	0.9	**0**	–

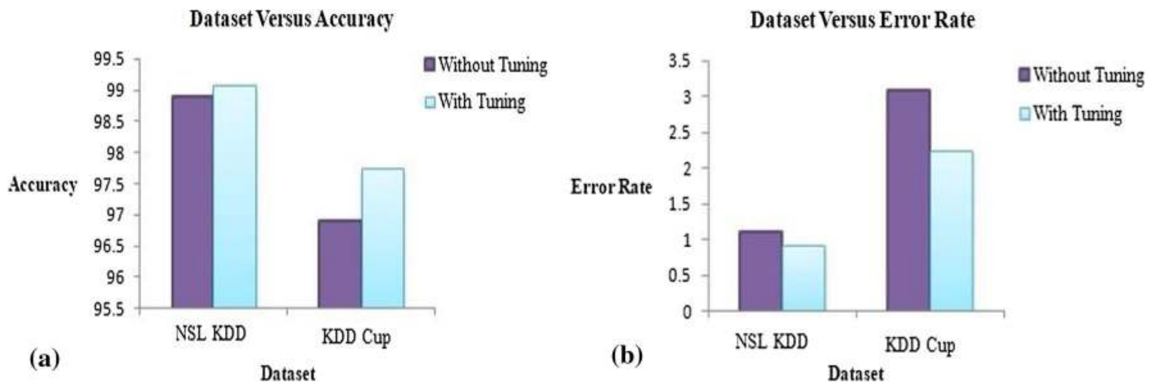


Fig. 8 Dataset versus performance metrics **a** accuracy **b** error rate

process could be slow if only excitatory neurons are used. Hence, the anti-correlations represented by the negative weights are represented by parallel opposing populations of excitatory neurons which are kept separated and anti-correlated by inhibitory neurons that connect between them improved the performance of the tuning process.

5 Conclusion

In this article, TVCDNN approach is proposed for the detection of DDoS attacks. The objective is to generalize the structure and parameters of VCDNN to identify DDoS attacks with better performances. The structure and parameters of TVCDNN network were tuned using BCuI and RCuI, respectively. The TVCDNN-based DDoS attack detection system identified the category of traffic. The benchmark datasets, viz. KDD Cup and NSL KDD have been used to evaluate the performance of the proposed TVCDNN approach. The experimental results showed that the proposed approach yields significant improvement in accuracy and reduction in error rate compared to the state-of-the-art deep learning approaches. Further, it is observed from the experimental analysis that the proposed TVCDNN approach converges faster than the existing optimization algorithms, such as GA, DE, PSO, and ACO. It is also depicted that tuning the structure and parameters of the VCDNN shows improvement in performance compared to deep neural network without tuning. However, the extension of this work could be on the fly detection of DDoS attacks using deep autonomous learning.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Zargar ST, Joshi J, Tipper D (2013) A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks. IEEE Commun Surv Tutor 15(4):2046
2. Bojovic PD, Basicevic I, Ocovaj S (2019) amd M. Popovic, A practical approach to detection of distributed denial of service attacks using a hybrid detection method, Computers & Electrical Engineering 73:84–96
3. Oleg Kupreev AG, Ekaterina Badovskaya (2019) Kaspersky lab ddos threat report. https://securelist.com/ddos-attacks-in-q1-2019/90792/
4. Saied A, Overill RE, Radzik T (2016) Detection of known and unknown ddos attacks using artificial neural networks. Neurocomputing 172:385
5. Schmidhuber J (2015) Deep learning in neural networks: an overview. Neural Netw 61:85
6. Asad M, Asim M, Javed T, Beg MO, Mujtaba H, Abbas S (2020) DeepDetect: Detection of distributed denial of service attacks using deep learning. Comput J 63(7):983–994
7. Sohi SM, Seifert JP, Ganji F (2021) RNNIDS: Enhancing network intrusion detection systems through deep learning. Comput Secur 102:1–23
8. Amma BN, Selvakumar S, Velusamy RL (2020) SAGRU: A stacked autoencoder-based gated recurrent unit approach to intrusion detection, in Springer Intelligent Data Engineering and Analytics, pp. 41–50
9. Kasim O (2020) An efficient and robust deep learning based network anomaly detection against distributed denial of service attacks. Comput Netw 180:1–10
10. Zhao R, Yin J, Gui G, Adebisi B, Obtsuki T, Gacanin H, Sari H (2021) An efficient intrusion detection method based on dynamic autoencoder. IEEE Wirel Commun Lett 10:1707
11. Ge M, Syed NF, Fu X, Baig Z, Kelly AR (2021) Towards a deep learning driven intrusion detection approach for Internet of Things. Comput Netw 186:1–11
12. Vinaykumar R, Soman KP, Poornachandran P (2017) Applying convolutional neural network for network intrusion detection, in 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI) (IEEE, 2017), pp. 1222–1228
13. Amma NGB, Selvakumar S (2018) VCDeepFL: Vector Convolutional Deep Feature Learning Approach for Identification of Known and Unknown Denial of Service Attacks, in TENCON 2018–2018 IEEE Region 10 Conference (IEEE, 2018), pp. 0640–0645

14. Zhao L, Qian F (2011) Tuning the structure and parameters of a neural network using cooperative binary real particle swarm optimization. Expert Syst Appl 38(5):4972
15. Bhardwaj A, Tiwari A, Bhardwaj H, Bhardwaj A (2016) A genetically optimized neural network model for multi-class classification. Expert Syst Appl 60:211
16. Qolomany B, Maabreh M, Al-Fuqaha A, Gupta A, Benhaddou D (2017) Parameters optimization of deep learning models using Particle swarm optimization, in Wireless Communications and Mobile Computing Conference (IWCMC), 2017 13th International (IEEE, 2017), pp. 1285–1290
17. Liang JJ, Qu BY, Mao X, Niu B, Wang D (2014) Differential evolution based on fitness Euclidean distance ratio for multimodal optimization. Neurocomputing 137:252
18. Amma BN, Selvakumar S (2019) Deep radial intelligence with cumulative incarnation approach for detecting denial of service attacks. Neurocomputing 340:294
19. McHugh J (2000) Testing intrusion detection systems: a critique of the 1998 and 1999 Darpa intrusion detection system evaluations as performed by Lincoln laboratory. ACM Trans Inform Syst Secur TIS-SEC 3(4):262
20. Tavallaee M, Bagheri E, Lu W, Ghorbani AA (2009) A detailed analysis of the KDD CUP 99 data set, in Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on (IEEE, 2009), pp. 1–6
21. Iglesias F, Zseby T (2015) Analysis of network traffic features for anomaly detection. Mach Learn 101(1–3):59
22. Buczak AL, Guven E (2016) A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Commun Surv Tutor 18(2):1153
23. Weller-Fahy DJ, Borghetti BJ, Sodemann AA (2015) A survey of distance and similarity measures used within network intrusion anomaly detection. IEEE Communications Surveys & Tutorials 17(1):70
24. Amma N, Selvakumar S (2020) A statistical class center based triangle area vector method for detection of denial of service attacks, Cluster Computing-The Journal of Networks Software Tools and Applications
25. Amma NGB, Subramanian S (2019) Feature Correlation Map based Statistical Approach for Denial of Service Attacks Detection, in 2019 5th International Conference on Computing Engineering and Design (ICCED) (IEEE, 2019), pp. 1–6
26. Velliangiri S, Premalatha J (2017) Intrusion detection of distributed denial of service attack in cloud, Cluster Computing pp. 1–9
27. Prieto A, Prieto B, Ortigosa EM, Ros E, Pelayo F, Ortega J, Rojas I (2016) Neural networks: an overview of early research, current frameworks and new challenges. Neurocomputing 214:242
28. Idhammad M, Afdel K, Belouch M (2018) Semi-supervised machine learning approach for ddos detection. Appl Intell 48:3193
29. Tsai JT, Chou JH, Liu TK (2006) Tuning the structure and parameters of a neural network by using hybrid taguchi-genetic algorithm. IEEE Trans Neural Netw 17(1):69
30. Liu J, Gong M, Miao Q, Wang X, Li H (2018) Structure learning for deep neural networks based on multiobjective optimization. IEEE Trans Neural Netw Learn Syst 29(6):2450
31. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436
32. Tajbakhsh N, Shin JY, Gurudu SR, Hurst RT, Kendall CB, Gotway MB, Liang J (2016) Convolutional neural networks for medical image analysis: full training or fine tuning? IEEE Trans Med Imag 35(5):1299
33. Shone N, Ngoc TN, Phai VD, Shi Q (2018) A deep learning approach to network intrusion detection. IEEE Trans Emerg Topics Comput Intell 2(1):41
34. Yuan X, Li C, Li X (2017) DeepDefense: Identifying DDoS Attack via Deep Learning, in 2017 IEEE International Conference on Smart Computing (SMARTCOMP) (IEEE, 2017), pp. 1–8
35. Ghiasi-Shirazi K (2019) Generalizing the convolution operator in convolutional neural networks, Neural Processing Letters pp. 1–20
36. Keserwani PK, Govil MC, Pilli ES (2021) An effective NIDS framework based on a comprehensive survey of feature optimization and classification techniques, Neural Computing and Applications pp. 1–21
37. Mavrovouniotis M, Yang S (2015) Training neural networks with ant colony optimization algorithms for pattern classification. Soft Comput 19(6):1511
38. Elbeltagi E, Hegazy T, Grierson D (2005) Comparison among five evolutionary-based optimization algorithms. Adv Eng Inform 19(1):43