

# A tree growth based forward feature selection algorithm for intrusion detection system on convolutional neural network

Mathiyalagan Ramasamy<sup>1</sup>, Pamela Vinita Eric<sup>2</sup>

<sup>1</sup>Department of Computer Science and Engineering, New Horizon College of Engineering, Bengaluru, India

<sup>2</sup>Department of Computer Science and Engineering, Presidency University, Bengaluru, India

## Article Info

### Article history:

Received Apr 29, 2022

Revised Sep 10, 2022

Accepted Oct 12, 2022

### Keywords:

DARPA'98 IDS datasets

Deep neural network

Improved deep convolutional neural network

Intrusion detection system

## ABSTRACT

With the rapid advancement of networking technologies, security system has become increasingly important to academics from several sectors. Intrusion detection (ID) provides a valuable protection by reducing the human resources required to keep an eye on intruders, improving the efficiency of detecting the various attacks in networks. Machine learning and deep learning are two key areas that have recently received a lot of attention, with a focus on improving the precision of detection classifiers. Using defense advance research project agency (DARPA'98) datasets, a number of academics and research have developed intrusion detection systems. This paper discusses various approaches developed by different researchers, including scale-hybrid-IDS-AlertNet (SHIA), forward feature selection algorithm (FFSA), modified- mutual information feature selection (MMIFS), deep neural network (DNN), and the holes that remain to be filled, highlighting areas where these procedures can be improved, also are addressed and the proposed approach improved deep convolutional neural network (IDCNN) is compared with existing approach.

*This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.*



## Corresponding Author:

Mathiyalagan Ramasamy

Department of Computer Science Engineering, New Horizon College of Engineering

Bengaluru, India

E-mail: mathi.prajval@gmail.com

## 1. INTRODUCTION

Computer networks play an important part in today's society, but they are subject to invasions. As a result, the best available ways to defend the systems are required [1]. Any action performed to compromise the system's integrity, confidentiality, or availability is referred to as an intrusion. A few intrusion prevention solutions would be used as a key security feature; however, they are incapable to cover entire demand for preventing security breaches. Because of design and programming flaws, as well as numerous penetration tactics, there will always be exploitable weaknesses in systems as they get more complicated [2]. As a result, intrusion detection system (IDS) is required as a second stage to protect the system against such flaws. The IDS is useful not only for detecting existing intrusions, but also for preventing future intrusions [3]-[5]. The DARPA'98 intrusion detection datasets are frequently used to assess IDS efficacy [6]. They are grouped into four categories (DoS, U2R, R2L, and probe), which are further separated into 22 different types of attacks. In a denial-of-service (DoS) assault, intruders prevent authorized clients such as smurf, neptune, back, teardrop, pod, and land from using the service. In a probe attack, the intruders collect information about the host.

Large amounts of data that imitate real network data augment the temporal complexity of training and validation in intrusion detection systems, which is known as the dimensionality curse. Large data also consumes resources and can lead to decreased detection stability. It is practical to delete data that does not contribute to detection before processing. This leads to the creation of an effective feature extraction and

reduction policy that not only helps to cut training time but also improves accuracy and protects against unforeseen threats. The training set is categorized into different groups with fewer numbers, and also simple subsets, using fuzzy clustering. This enables the artificial neural networks (ANN) to examine each subset considerably more quickly, with greater and better accuracy. The experimental findings reveal that the developed FC-ANN achieves an average accuracy of 96.71% [7].

Used the grid-search (GS) and support vector machine (SVM) algorithm to devise and detect anomalous behaviours [8]. The goal is to improve the detection rate while lowering the false alarm rate. The paper employs the knowledge discovery and data mining tools competition (KDD CUP 1999) algorithm to test the program's functionality. The results demonstrate that this strategy boosted the rate of detection and decreased the number of false alarms [9].

Devised IDS utilising support vector machine and provided certain feature selection strategies [10]. They investigated features using two feature selection methods. The forward feature and the linear correlation coefficient were used as techniques. The proposed approach selection algorithm, forward feature selection algorithm (FFSA) and the suggested modified-mutual information feature selection algorithm (MMIFS).

For better performance, created a network IDS. The first step is to model intrusion detection using deep neural network (DNN). The system is built using many source technologies, including tcpdump for packet capture, bro for traffic analysis, and tensorflow for machine learning. However, the results of repeated operations with DARPA'98 datasets and the resulting network flow test reveal that the proposed IDS-DNN system improves accuracy as well as attack detection [10]-[13].

The previous problem in IDS system is degradation of performance when number of data increases in the network. Deep learning is not suggested for using in small data processing environment. Deep learning requires large to massive data generation environment for performing efficient learning operation [14]-[16]. In spite of these advantages, false positive data prediction degrades the accuracy of deep learning due to growing size of data. Huge data volume and input space reduce the quality of classifying intrusion in the network. This misclassifying data makes to increase in false positive results and degrades the performance of IDS. So, it is need of feature selection algorithm to identify the relevant features and classification is finally done on that feature. This sequence helps to improve the classification performance in IDS scenario. Feature engineering is dominant and interesting field in machine learning platform [17]-[20]. Feature selection is classified as filter, wrapper and hybrid models. Some feature selection model uses intelligent algorithms for improving performance of the system. In this paper we use wrapper based intelligent techniques for improving intrusion detection.

## 2. BACKGROUND OF VARIOUS MACHINE LEARNING CLASSIFIERS

### 2.1. Support vector machine classifier

The SVM gains more usage among machine learning application due to its simplest processing style. Recently, cloud computing, big data, distributed environment uses SVM in all its data optimization and classification models. Supervised learning technique is the base for SVM, which classify various data in network. Linear problems as well as nonlinear problems are easily handled by SVM classifier. SVM first generates Hyper plane one to many among high dimensional data space. Hyper plane separates the data with optimal splitting strategy using type of class data belongs to and perform best classification [21]-[24].

### 2.2. K-nearest neighbor

K-nearest neighbor (KNN) is a most traditional and well-known classification model in machine learning platform. It computes the Euclidean distance for classifying the data in space [25]-[27]. Let p and q be data in space S, then the distance of p and q,  $d(p; q)$ , is derived in below expression:

$$d(p, q) = \sqrt{\sum_{s=1}^n (P_s - q_s)^2} \quad (1)$$

total number of data in space is represented as 'n'. The KNN uses Euclidean distance for selecting data in space. When the distance is minimum, data is nearest and optimal for selection.

### 2.3. Naïve Bayes

Naïve Bayes (NB) is old classifier based on Bayes' theorem [28]-[30]. Features in the dataset are assumed as independent (naïve). Let A be an object with K number of features, which is to be classified with vector representation  $X(x_1, \dots, x_n)$ . NB works as given (17) for class M(k).

$$p\left(\frac{m_k}{X}\right) = \frac{p\left(\frac{X}{m_k}\right)p(m_k)}{p(X)} \quad (2)$$

$$y = \operatorname{argmax} p(m_k) \prod_{s=1}^n p(X_s | m_k) \quad (3)$$

X is decided by (3).

#### 2.4. Feed forward deep neural networks

Deep learning is widely accepted and proved in more complex solutions. Neural networks are various types which compute using artificial neurons. These ANN are reflection of biological neurons in the brain. In feed forward neural network, ANN forwards the received information at input. The real time problems with enhancing learning and approximation in nonlinear circumstances are used in feed forward ANN [24]. Here rectified linear unit is used as an activation function.

#### 2.5. Decision tree with random forest

Most complex data mining strategy uses decision tree (DT) feature extraction techniques. DT algorithm produces predictive copy in the tree shape based on process [31]. There are three main nodes such as root (one node), internal (many nodes) and category (child nodes). Top-down classification is occurring in classification of DT till optimal leaf node is found. Multiple decision tree execution is termed as random forest classification. Table 1 describes the survey of various feature selection model.

Table 1. Survey on feature selection in IDS

Reference	Feature selection	Dataset	Technique
[15]	Greywolf optimizer-based feature selection	UNSW-NB15	Machine learning
[16]	Nature-inspired optimization framework J48 (C4.5)	UNSW-NB15	Machine learning
[17]	EvoPy-FS optimization framework	UNSW-NB15	Machine learning
[18]	Firefly algorithm and support vector machine	NSL-KDD	Machine learning
[19]	J48 (C4.5 decision tree) classifier	NSL-KDD	Machine learning
[20]	Deep boltzmann machines	NSL-KDD	Deep learning
[21]	SVM, extreme learning machine (ELM) and random forest (RF)	NSL-KDD	Machine learning
[22]	Deep auto-encoder based	NSL-KDD and KDDCup 99	Deep learning
[23]	Recurrent neural networks	NSL-KDD and KDDCup 99	Deep learning
[24]	GA-LR wrapper approach	KDD Cup 99	Evolutionary based algorithms
[25]	Ant colony optimization (ACO)	KDD Cup 99	Machine learning
[26]	Flooding DoS attacks	KDD Cup 99	Machine learning

An independent abuse identification method that combines a self-taught approach with the advantages of learning algorithms. Frameworks and MAPE-K while completing the plan activities within the MAPE-K platform, employed sparse auto encoder for the unsupervised technique method to discover relevant attributes. The KDD Cup'99 and NSL-KDD databases were followed in the research. The fundamental flaw is that the user to root attack and root to Local assault module has a low discovery rate [32].

A 2 phase strategy approach on deep layered auto encoder that is both efficient and effective. With probability values, the dataset was first categorized into the assault and usual classes. These probabilities achieved then were submitted to the final judgment step for regular and multi-class classification attack categorization as an extra feature. The suggested model's performance is analyzed using the KDD Cup'99 and UNSWNB15 databases. A separate technique was used for these datasets to alleviate challenges presented by a class imbalance in the datasets. Downsampling was used during the KDD Cup'99 to eradicate replacement recordings. SMOTE was used to upsample the dataset to equalize the distribution of entries in UNSWNB15. The DR performance of attack classes with training samples improved considerably as a result of this processing of the database [33].

Three IDS models: completely connected systems, KyotoHoneypot, Variational AE, and Seq2Seq. IDS2017, MAWILab, NSL-KDD, and UNSW-NB15, recordings were all used to test these models. Found that when evaluating different methods in terms of detection rate among all datasets, the Seq2Seq trained model with two recurrent neural networks (RNNs) fared the best [34]. Developed an ID model instance-based aggressive variational AE with regularisation and DNN (SAVER-DNN). The model's performance was evaluated using benchmark information from NSL-KDD and UNSW-NB15. The representation usefulness in recognizing lower occurrence and fresh threats has been confirmed by experimental evidence [35].

Developed a multistage stage comprising the 1D convolution operation and 2 heaps of fully linked levels, using the principle of AE. To recreate the data once more in the unsupervised step, two auto encoders were trained and evaluated using normal and attack streams. These recently rebuilt items are employed in the direct step to make new enhanced datasets that were inserted in 1D. The outcome of that convolution operation was flattened and supplied convolution level, a database in conclusion classification used this softmax level. Experiments were passed away on the KDD Cup'99, CICIDS2017 UNSWNB15 database, their optional method outcome other DL models in comparison. They haven't proved how this model works for alternative groups. The next weakness was that it doesn't disclose any details on the attack's attributes [36].

Intrusion detection system solutions are not able to cope pace with the advancements in the field of high-speed networking. Before IDS products that are now used in the gigabit network can provide comprehensive protection against attacks, significant improvements are necessary despite their extensive documentation, intrusion detection solutions on the market can only identify 50% of all threats. As a result, the present focus of this thesis is to create a network-based IDS that can identify attacks in a big, high-speed, high-volume venture network. Research by Antunes *et al.* [37] offer a scenario in which an attacker attacks a specific system without authorization while avoiding harming other networks with the necessity to look for, and send hidden and sensitive data. Experts can carry out such an assault to conceal the entire attack and avoid detection. Large corporate IDS detection is also impossible since a single connection is targeted. The much more critical aspect of the assault is that existing systems are incapable of detecting such attacks, and the hacker can erase all traces of his or her activity before exiting the system. In the case of a complete invader, with exception of worms, imitation does not make it more difficult to recognize their destination by diminishing its functionality. Despite being able to identify denial of service intrusions, the IDS should be able to detect attacks as well as the intruder's motives. These attacks are network-specific, and the attacker's goal is to gain money. As a result, the severity of attacks has risen, as has the focus on a particular network. As a result, these attacks are difficult to detect with basic IDS, necessitating the use of improved IDS to identify severe attacks. Let's look at the history of network security to see how we might identify harsh attacks and learn more about security.

### 3. IMPROVED DEEP CONVOLUTIONAL NEURAL NETWORK

#### 3.1. Data preprocessing

Step 1: data normalization

Normalizing the data is the first step to reduce the noise. Removed noise from dataset helps to process without interruption. PCA is computed after normalization. Data is normalized by taking away data which is greater than mean value from the certain column. Consider there are two dimensions P and Q, all P become p and all Q become q. Then the mean computed will be zero for this condition.

Step 2: covariance matrix computation.

We take dataset with 2D information and matrix will be as:

$$Cov\ matrix = \begin{bmatrix} var[p1] & cov[p1,p2] \\ cov[p2,p1] & var[p2] \end{bmatrix} \quad (4)$$

where: Variance [p1]=cov[p1,p1]

Variance [p2]=cov[p2,p2]

Step 3: calculating the eigenvalues and eigenvectors.

Here we compute eigen vectors and eigen values for covariance matrix. The square matrix can be easily computed to eigen values and vectors. Eigenvalues is  $\lambda$  for given matrix and solution is obtained by (5);

$$D(\lambda I - A) = 0 \quad (5)$$

where: I : identity matrix of A matrix with equal dimensions

D: matrix determinant

$\lambda$  : eigen value for eigen vector M get from following equation

$$M(\lambda I - A) = 0 \quad (6)$$

Step 4: the components are chosen and feature vectors are formed.

Here dimensionality reduction is done in features. Highest value is considered as principle component of the input dataset. It computes based on choosing value of eigen. Dimensions are reduced by choosing appropriate first x eigenvalues and rest is ignored. Note that if eigen value is less then we will loss huge needed feature. So, eigen value is selected based on covariance computed.

### 3.2. Tree growth based FFSA

#### 3.2.1. Binary TGA algorithm

For feature selection we use binary tree growth algorithm (TGA) [1]. Binary TGA is explained in following section. TGA is accomplished for continuous optimization problem with good performance. Major concern is, TG algorithm cannot be applied directly to feature selection process. Agarwal *et al.* [26] suggested novel enhanced TGA called improved TGA with grey wolf optimizer strategy. But grey wolf does not efficiently compute TGA for IDS.

In our proposed model we use fire fly optimizer with improved binary-TGA. Fire fly fitness value embeds with improved binary-TGA for optimal detection solution. In first phase binary TGA for feature selection is computed. Second phase linear mechanism is used for  $\theta$  computation. Third phase we embed firefly fitness with binary TGA to increase the performance of exploration and exploitation categories in solution. Feature selection is considered to be a NP hard problem due to large growing size of feature. TGA works perfectly by proving superiority in continuous problem. Various advantages TG algorithm encourage us to use in feature selection problem. TGA has to be modified with n dimensional Boolean space by using value 0'S OR 1'S. The transform function is used and utilized in TGA for working as binary function on in TGA. For changing continuous to discrete, all values in dataset are changed as continuous to discrete value with following (7).

$$s_i = \begin{cases} 1 & \text{if } \frac{s_i}{\sum_{i=1}^n s_i} \geq \frac{1}{n} \\ 0 & \text{if } \frac{s_i}{\sum_{i=1}^n s_i} < \frac{1}{n} \end{cases} \quad (7)$$

Whereas n represents dimensions and it is a total feature in the dataset. Solution is obtained by (8).

$$s = (s_1, s_2 \dots \dots \dots, s_n), s_i \in \{0,1\}, i = 1,2, \dots, n \quad (8)$$

Where as  $s_i$  represents selected feature in ith position. If the  $s(i)=1$  then it means feature is selected, else it is not selected. Example considers the selected value as  $S=(1, 0, 1, 1, 1, 0\dots)$ , the features first, third, fourth, fifth are selected. The fitness function must be designed, such that it must provide more accurate feature in feature selection as well as classification model. In this model we use fitness model for evaluating the subset selection with high classification accuracy.

$$Fitnessvalue = \alpha E_r(F) + \beta \frac{|P|}{|T|} \quad (9)$$

Classification error rate is computed as  $E_r(F)$ ,  $|p|$  denotes number of selected features,  $|T|$  denotes total features in the dataset,  $\alpha$  and  $\beta$  are two parameters to evaluate criteria 0 or 1. Here  $\alpha \in [0,1]$ ,  $\beta = 1 - \alpha$  used from [3]. SVMs are learning techniques that employ a hypothesis space of linear functions in a high-dimensional higher dimensional space. SVMs are built on the concept of linear separability, sometimes known as a hyper-plane classifier. Consider that there are N training pieces of information.  $\{(x_1, y_1), (x_2, y_2), (x_3, y_3), \dots, (x_n, y_n)\}$ , where  $x_i$  belong to  $R^n$  and  $y_i \in \{+1, -1\}$ . Consider a hyper-plane defined by  $(w, b)$ , where  $w$  is a weight vector and  $b$  is a bias, see Figure 1. We can classify a new object  $x$  with  $f(x) = \sin(w \cdot x + b) = \sin(\sum_{i=1}^n a_i y_i (x_i \cdot x) + b)$ . Remember that training matrices  $x_i$  are a linear combination; there's a Lagrangian integrator I for each repetition. The Lagrangian integrators indices, I represent the significance of each interactive medium. When the biggest marginal high energy is found, only areas near it will also have  $I > 0$ . All of those are essential procedures. Figure 1 shows that  $i=0$  for all other locations. This means only points near to the hyperplane constitute the assumption, optimised using  $\min_x \|w\|_2 / 2$  subject to  $y_i(x_i \cdot w + b) \geq 1$ .

These key pieces of information act as support vectors. Figure 2 displays two classes together with their margins, or borders. Support vectors are solid objects whereas nonsupport vectors are vacant. The margins are solely influenced by training examples; removing or adding empty entities won't modify them. However, any modification to the solid objects, such as their addition or removal, might alter the margins. The results of include items in the margin region are shown in Figure 1. As can be seen, adding or removing items that are too far from the boundaries, such predictor variables 1 or 2, has no impact on the edges. However, eliminating or make additions that were close to the intervals, such piece of proof 2 and /or 1, resulted in more margins. Under this configuration, a grouping trees is regularly created.

One of hierarchical trees employing this method is seen growing in Figure 3. The bold nodes are regression coefficients. Nodes 1, 2, 3, 5, 6, and 9 may grow since they're support vector nodes. Since nodes 4,

8, 7, and 10 are not vector support networks, we halt their growth in the interim. Growing tree increases information set's points for a more accurate performance. Developing a tree to a specific size/level using the set values (+3, +4, +5, +6, +7, 3, 4, 5, 6, 7). The dash (--) nodes indicate group without support vectors. The gradient boosting links are shown by the prominent nodes. We add +4 and +7's offspring to the training set. The SVM nodes 5 & 7 are same. The updated database is (3, 4, 10, 11, 6, 12, 13), (+3, +11, +12, +5, +6, +14, +15). Notice that +4 and +7 are omitted since their offspring are in the training set. Extending vertices to the boundary area has already been fixed, and the make better (dotted line) is much more precise. From doing this, the classifier is correctly adjusted, increasing the accuracy level. RNN excellent correctness in prototype identification of keysection of IDs. Compared to feedforward and Elman recurring neural network models, RNN fared better.

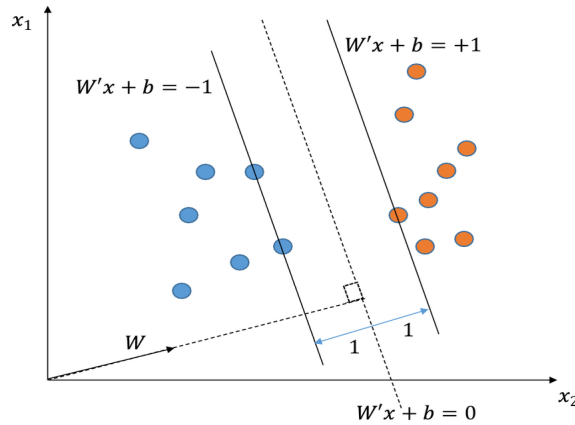


Figure 1. SVM's linear separating

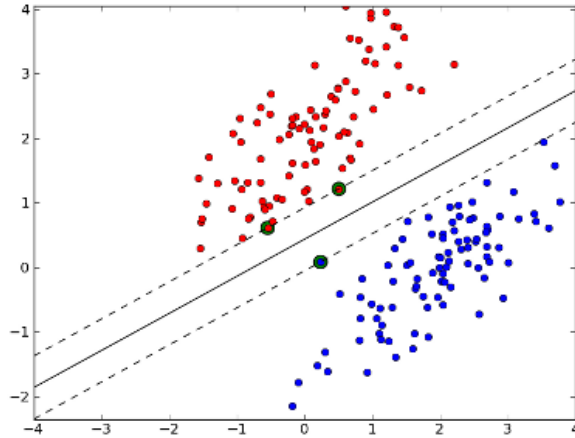


Figure 2. Putting the support vector positions and the non-support vector positions in separate categories

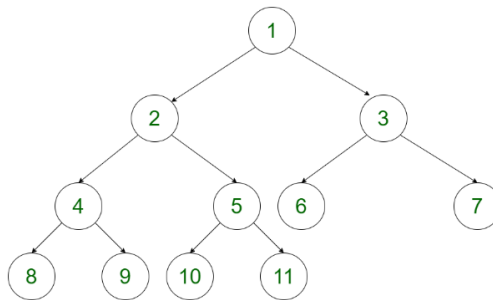


Figure 3. Constructing hierarchies that used proposed method

### 3.2.2. Utilizing the enhanced inertia weight-based dragons flyer enhancer, feature choosing (IIW-DFO)

The International maritime organization and oceans is imitating the activities of the gonfly during migrating or during foraging. Swarming comes in two flavours: dynamic and static. Narrow groups of dragonfly chase other swarms in a limited zone, causing significant local activity changes. In a dynamical swarm, many dragonfly fly in one direction for a great range. Research and exploitation of steady and transient activity are key to swarm-based methods. To enhance domestic and international searching, tweak the five parameters. Following data methods, it has been determined that:

#### a. Partition

The aversion to additional agents in the district to distinguish themselves. It is computed in the same way as in (5.1).

$$S_i = -\sum S_i = -\sum_{j=1}^N X - X_j \quad (10)$$

where  $X$  is the position,  $N$  is no. of neighbors of the dragonfly,  $S$  is partition movement of the  $i^{\text{th}}$  individual,  $X_j$  is position of the  $j^{\text{th}}$  neighbor.

#### b. Position

Synchronizing of an individual's rapidity to that of a neighboring individual. It is the agent's configuration of velocity with the neighboring dragonflies' velocity vector. It is computed in the same way as in (11):

$$A_i = \frac{\sum_{j=1}^N V_j}{N} \quad (11)$$

where,  $A_i$  is alignment motion for  $i^{\text{th}}$  item and  $V_j$  is velocity of the  $j^{\text{th}}$  neighborhood.

#### c. Cohesion

Measuring a person's distance from the neighborhood's center. It's written like (12):

$$C_i = \frac{\sum_{j=1}^N x_j}{N} - X \quad (12)$$

where,  $C_i$  is cohesion of the  $i^{\text{th}}$  term and  $N$  is neighborhood of size.

#### d. Repulsion towards provisions

The motion of a dragonfly more toward the lure of food is depicted in (13).

$$F_i = X^+ - X \quad (13)$$

where  $F_i$  is attraction of food of  $i^{\text{th}}$  terms,  $X^+$  is location of source of food.

#### e. Enemies of distraction: dragonflies avoid adversaries, as shown in (14):

$$E_i = X^- + X \quad (14)$$

where  $E_i$  is distraction,  $X^-$  is enemy position. Position vector  $X$ , have been used to maintain the location of the particular dragonfly. The phase vector is much like the PSO computation velocity vector Dhanabal *et al.* [34], which is specified as Yin *et al.* [22] in the code (15).

$$\Delta X_{t+1} = (S_i + A_i + C_i + F_i + E_i) + \omega \Delta X_t \quad (15)$$

where  $\omega$  is weight of inertia and  $t$  is iteration of counter. The proportional gain of the step direction has been enhanced in this suggested study, as in (16).

$$\omega = \omega_{\max} - \frac{\omega_{\max} - \omega_{\min}}{t_{\max}} \cdot t \quad (16)$$

where,  $\omega_{\max}$  and  $\omega_{\min}$  are initial and finish ranges,  $t_{\max}$  is iteration of max and  $t$ -iterations. As the number of repetitions increases, the weight value decreases, leaving the universal retrieval accuracy. After step vectors calculation, important while designing is changed (17).

$$X_{t+1} = X_t + \Delta X_{t+1} \quad (17)$$

### 3.2.3. IIW-DFO of pseudo code

The 5 synaptic weights, velocity distribution, and positional matrices get increased inertia strength up until the things that change is reached [37]. If the dragons fly has any neighbors, the location is also changed. Customized attributes are put into such a deep learning-based classification scheme called DBCNN for vulnerability scanning (2020).

```

Input: Number of dragonflies & stepped vector Xi, i=(1,2,...n)
1: repeat the process in as instances as feasible (t maximum)
2: The goal ranges are established.
3: Refresh the provisions supply & adversary.
4: 5 weighted components S, A, C, F, and E using (10). (15)
5: Update your neighbors' perimeter
6: dragonfly has a companion
7: Update angular rateSydne et al., (2020) using it (16)
8: Increase the inertial weighting using the inertial weighting (17)
9: Update the positioning and velocity with it (18)
10: if it not,
11: The location variable is updated using levy flight Hammad et la., (2020).
12: If all else not pass, called it per daytime.
13: The newedlocation of the dragonfly were determined by altering borders.
14: End all
Outcome: bring back the chosen features and functionality.

```

## 4. RESULTS AND EXPERIMENTS

MATLAB is used to conduct the experiments. The DARPA'98 dataset is a performance benchmark for intrusion detection methods. It is most well-known IDS with extensive ID data sets [27]. It is classified into five categories, including normal and four forms of invasions (i.e., DoS, Probe, U2R and R2L) [10]. It is made up of both training and test data. There are about 5 million records of connections in the training data, and over 2 million links in the testing data. All of the data is classified as either normal or intrusive. Real data is measured by using TPR as the real data with following calculation:

$$TPR = \frac{P}{P+Q} \quad (18)$$

The intrusion data is identified as an intrusion in the network using proposed TGFF algorithm is calculated as TNR using:

$$TPR = \frac{S}{S+R} \quad (19)$$

The intrusions in the network sometime judged as good feature. This identity is measured as:

$$FPR = \frac{R}{R+S} \quad (20)$$

The normal real feature sometimes judged as intrusion feature. This identity is measured using:

$$FNR = \frac{Q}{P+Q} \quad (21)$$

Nest we calculate accuracy of intrusion detection in the network in percentage. Accuracy is defined as intrusions that are correctly predicted as intrusion feature and real feature is detected as real feature using proposed TGFF algorithm.

$$Accuracy = \frac{TPR+TNR}{TPR+TNR+FPR+FNR} \quad (22)$$

Precision is defined as ratio of number of predictions is considered as correct. It is calculated by P: P+R:

$$Precision = \frac{TPR}{TPR+FPR} \quad (23)$$

Sensitivity of the detection is calculated by representing all true positive evaluations divided by all positive evaluation. It is given by:

$$Sensitivity = \frac{TPR}{TPR+FNR} \quad (24)$$



Accuracy is otherwise tested by using F-measure score. It balances the precision and sensitivity in detecting real intrusion:

$$F - measure = \frac{2 * precision * sensitivity}{precision + sensitivity} \quad (25)$$

Where:

- Accuracy: it's described as the design's overall appropriateness, and it's measured as the ratio of real classification. The accuracy comparison, such as FFSA, DNN, scale-hybrid-IDS-AlertNet (SHIA), and improved deep convolutional neural network (IDCNN), is shown in Figure 4. Systems are measured on the x-axis, and precision is measured on the y-axis. Kalman filtering is a method used in IDCNN-based attacker identification. The accuracy is calculated as like in Figure 4.
- Precision: by examining the false positive and true positive instances. Figure 5 shows a precision comparison of the SHIA, DNN, FFSA, and IDCNN algorithms. Methods are on the x-axis, and precision is on the y-axis. Kernel random vectors are used in IDCNN to compute the relevance vectors and number of samples. The great precision in intrusion detection is achieved by considering the large likelihood samples using IDCNN. According to the experimental data, the IDCNN achieves a precision of 90%, whereas other approaches such as KMIE, multilayer Bayesian, DNN, and LSSVM achieve 78%, 81.5%, 87.89%, and 86.78%, respectively.

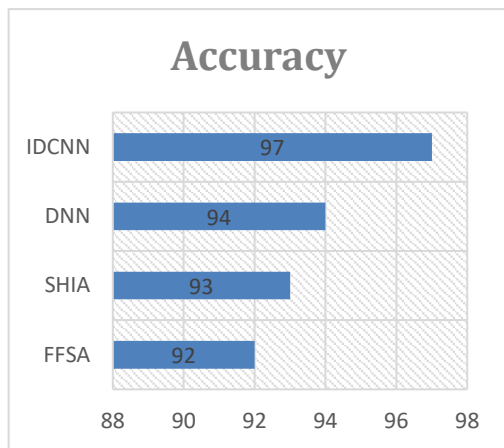


Figure 4. Accuracy

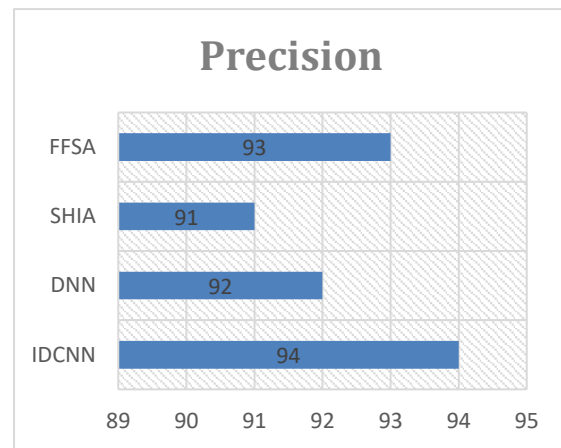


Figure 5. Precision

## 5. CONCLUSION

A difficult area of research is finding network intrusions. There are other capabilities offered by the incursion that allow you to use the network as a real user. The effectiveness of detection in an IDS system is affected by a number of features. IDS is a widely used network security method that protects the integrity and accessibility of critical resources in secured systems. Machine learning is used to improve the efficiency of IDS, despite the fact that there are varieties of supervised and unsupervised research methodologies. In this article, we looked at various methods for detecting intruders, such as SHIA, FFSA, DNN, and improved DCNN. However, based on observations, there is still need to improve current intrusion detection algorithms in order to get appropriate results.

## REFERENCES




- [1] A. Iqbal and S. Aftab, "A Feed-Forward and Pattern Recognition ANN Model for Network Intrusion Detection," *International Journal of Computer Network and Information Security*, vol. 11, no. 4, pp. 19–25, Apr. 2019, doi: 10.5815/ijcnis.2019.04.03.
- [2] P. Negandhi, Y. Trivedi, and R. Mangrulkar, "Intrusion Detection System Using Random Forest on the NSL-KDD Dataset," in *Emerging Research in Computing, Information, Communication and Applications, Springer Singapore*, 2019, pp. 519–531, doi: 10.1007/978-981-13-6001-5\_43.
- [3] C. Zhong, Y. Chen, and J. Peng, "Feature Selection Based on a Novel Improved Tree Growth Algorithm," *International Journal of Computational Intelligence Systems*, vol. 13, no. 1, p. 247, 2020, doi: 10.2991/ijcis.d.200219.001.
- [4] J. Zhang, F. Li and F. Ye, "An Ensemble-based Network Intrusion Detection Scheme with Bayesian Deep Learning," *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1-6, doi: 10.1109/ICC40277.2020.9149402.

- [5] C. Koliás, G. Kambourakis, A. Stavrou and S. Gritzalis, "Intrusion Detection in 802.11 Networks: Empirical Evaluation of Threats and a Public Dataset," in *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 184–208, Firstquarter 2016, doi: 10.1109/COMST.2015.2402161.
- [6] R. Mitchell and I.-R. Chen, "A survey of intrusion detection in wireless network applications," *Computer Communications*, vol. 42, pp. 1–23, Apr. 2014, doi: 10.1016/j.comcom.2014.01.012.
- [7] J. Hu, X. Yu, D. Qiu, and H.-H. Chen, "A simple and efficient hidden Markov model scheme for host-based anomaly intrusion detection," *IEEE Network*, vol. 23, no. 1, pp. 42–47, Jan. 2009, doi: 10.1109/mnet.2009.4804323.
- [8] D. A. Effendy, K. Kusriani and S. Sudarmawan, "Classification of intrusion detection system (IDS) based on computer network," *2017 2nd International conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, 2017, pp. 90–94, doi: 10.1109/ICITISEE.2017.8285566.
- [9] E. Viegas, A. O. Santin, A. Franca, R. Jasinski, V. A. Pedroni, and L. S. Oliveira, "Towards an Energy-Efficient Anomaly-Based Intrusion Detection Engine for Embedded Systems," *IEEE Transactions on Computers*, vol. 66, no. 1, pp. 163–177, Jan. 2017, doi: 10.1109/tc.2016.2560839.
- [10] B. S. M. Hosseini, B. Amiri, M. Mirzabagheri, and Y. Shi, "A New Intrusion Detection Approach using PSO based Multiple Criteria Linear Programming," *Procedia Computer Science*, vol. 55, pp. 231–237, 2015, doi: 10.1016/j.procs.2015.07.040.
- [11] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.
- [12] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A Deep Learning Approach to Network Intrusion Detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, Feb. 2018, doi: 10.1109/tetci.2017.2772792.
- [13] I. Lopez-Moreno, J. Gonzalez-Dominguez, D. Martinez, O. Plhot, J. Gonzalez-Rodriguez, and P. J. Moreno, "On the use of deep feedforward neural networks for automatic language identification," *Computer Speech & Language*, vol. 40, pp. 46–59, Nov. 2016, doi: 10.1016/j.csl.2016.03.001.
- [14] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," *arXiv:1502.01852*, Dec. 2015, doi: 10.1109/iccv.2015.123.
- [15] Q. Al-Tashi, H. M. Rais, S. J. Abdulkadir, S. Mirjalili, and H. Alhussian, "A Review of Grey Wolf Optimizer-Based Feature Selection Methods for Classification," in *Algorithms for Intelligent Systems, Springer Singapore*, 2019, pp. 273–286, doi: 10.1007/978-981-32-9990-0\_13.
- [16] M. Madi, F. Jarghon, Y. Fazea, O. Almomani, and A. Saaidah, "Comparative analysis of classification techniques for network fault management," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 28, no. 3, pp. 1442–1457, May 2020, doi: 10.3906/elk-1907-84.
- [17] R. A. Khurma, I. Aljarah, A. Sharieh, and S. Mirjalili, "EvolvoPy-FS: An Open-Source Nature-Inspired Optimization Framework in Python for Feature Selection," in *Algorithms for Intelligent Systems, Springer Singapore*, 2019, pp. 131–173, doi: 10.1007/978-981-32-9990-0\_8.
- [18] W. L. Al-Yaseen, "Improving Intrusion Detection System by Developing Feature Selection Model Based on Firefly Algorithm and Support Vector Machine," *IAENG International Journal of Computer Science*, vol. 46, no. 4, pp. 534–540, Nov. 2019.
- [19] A. Taherkhani, G. Cosma, and T. M. McGinnity, "Deep-FS: A feature selection algorithm for Deep Boltzmann Machines," *Neurocomputing*, vol. 322, pp. 22–37, Dec. 2018, doi: 10.1016/j.neucom.2018.09.040.
- [20] R. Mathiyalagan and P. V. Eric, "Review on intrusion detection system based methods on knowledge discovery data (KDD) dataset," *CR*, vol. 7, no. 10, 2020.
- [21] P. V. Eric and R. Mathiyalagan, "An Efficient Intrusion Detection System Using Improved Bias Based Convolutional Neural Network Classifier," *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 12, no. 6, pp. 2468–2482, Apr. 2021, doi: 10.17762/turcomat.v12i6.5689.
- [22] C. Yin, Y. Zhu, J. Fei, and X. He, "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017, doi: 10.1109/access.2017.2762418.
- [23] S. Aljawarneh, M. B. Yassein, and M. Aljundi, "An enhanced J48 classification algorithm for the anomaly intrusion detection systems," *Cluster Computing*, vol. 22, no. S5, pp. 10549–10565, Sep. 2017, doi: 10.1007/s10586-017-1109-8.
- [24] C. Khammassi and S. Krichen, "A GA-LR wrapper approach for feature selection in network intrusion detection," *Computers & Security*, vol. 70, pp. 255–277, Sep. 2017, doi: 10.1016/j.cose.2017.06.005.
- [25] M. Ramasamy and P. V. Eric, "An improved deep bagging convolutional neural network classifier for efficient intrusion detection system," *Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 1, pp. 405–413, Feb. 2022, doi: 10.11591/eei.v11i1.3252.
- [26] M. Agarwal, D. Pasumarthi, S. Biswas, and S. Nandi, "Machine learning approach for detection of flooding DoS attacks in 802.11 networks and attacker localization," *International Journal of Machine Learning and Cybernetics*, vol. 7, no. 6, pp. 1035–1051, Nov. 2014, doi: 10.1007/s13042-014-0309-2.
- [27] V. L. L. Thing, "IEEE 802.11 Network Anomaly Detection and Attack Classification: A Deep Learning Approach," *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, 2017, pp. 1–6, doi: 10.1109/WCNC.2017.7925567.
- [28] S. Ding and G. Wang, "Research on intrusion detection technology based on deep learning," *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, 2017, pp. 1474–1478, doi: 10.1109/CompComm.2017.8322786.
- [29] I. Ahmad, M. Basher, M. J. Iqbal, and A. Rahim, "Performance Comparison of Support Vector Machine, Random Forest, and Extreme Learning Machine for Intrusion Detection," *IEEE Access*, vol. 6, pp. 33789–33795, 2018, doi: 10.1109/access.2018.2841987.
- [30] W. Mo, C. L. Gutterman, Y. Li, S. Zhu, G. Zussman, and D. C. Kilper, "Deep-Neural-Network-Based Wavelength Selection and Switching in ROADM Systems," *Journal of Optical Communications and Networking*, vol. 10, no. 10, p. D1, May 2018, doi: 10.1364/jocn.10.0000d1.
- [31] D. Papamartzivanos, F. Gómez Mármol and G. Kambourakis, "Introducing Deep Learning Self-Adaptive Misuse Network Intrusion Detection Systems," in *IEEE Access*, vol. 7, pp. 13546–13560, 2019, doi: 10.1109/ACCESS.2019.2893871.
- [32] M. Ramasamy and P. V. Eric, "A Novel Classification and Clustering Algorithms for Intrusion Detection System on Convolutional Neural Network," *Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 5, pp. 2845–2855, 2022, doi: 10.11591/eei.v11i5.4145.
- [33] M. A. Khan, "HCRNNIDS: hybrid convolutional recurrent neural network-based network intrusion detection system," *Processes*, vol. 9, no. 5, pp. 1–14, 2021, doi: 10.3390/pr9050834.
- [34] M. Dhanabal and S. P. Shantharadah, "A study on NSLKDD dataset for intrusion detection system based on classification algorithms," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 6, pp. 446–452, 2015, doi: 10.17148/IJARCC.2015.4696.
- [35] Z. Yang et al., "A systematic literature review of methods and datasets for anomaly-based network intrusion detection," *Computers & Security*, vol. 116, 2022, doi: 10.1016/j.cose.2022.102675.




- [36] G. Andresini, A. Appice, N. D. Mauro, C. Loglisci and D. Malerba, "Multi-Channel Deep Feature Learning for Intrusion Detection," in *IEEE Access*, vol. 8, pp. 53346-53359, 2020, doi: 10.1109/ACCESS.2020.2980937.
- [37] M. Antunes, L. Oliveira, A. Seguro, J. Veríssimo, R. Salgado, and T. Murteira, "Benchmarking Deep Learning Methods for Behaviour-Based Network Intrusion Detection," *Informatics*, vol. 9, no. 1, pp. 1-18, doi: 10.3390/informatics9010029.

## BIOGRAPHIES OF AUTHORS



**Mathiyalagan Ramasamy**    received the Bachelor of Technology in Information Technology in 2005 from Dr. Navalar Neduncheziyan college of Engineering, Affiliated by Anna University, Tamil Nadu, India, and he received the Master of Engineering in Computer Science and Engineering in 2010 from Oxford college of Engineering, Affiliated by Anna University, Tamil Nadu, India. He is currently working as Assistant Professor in Presidency University, Bengaluru, Karnataka, India. His research interests include intrusion detection system, convolutional neural network, and network security. He can be contacted at email: mathi.prajval@gmail.com.



**Dr. Pamela Vinitha Eric**    is working as a Professor in the Department of Computer Science and Engineering at the Presidency University, Bangalore, India. She received a Doctorate in Philosophy from National Institute of Technology Calicut, India. She has around 25 years of experience in the education sector. Her areas of interest are bioinformatics, data compression, cryptography, and network security. She is actively pursuing research and supervises several research scholars. She can be contacted at email: pamela.vinitha@gmail.com.