Genetic Algorithm based hyperparameter tuned CNN for identifying IoT intrusions

Alexander. R[1], and Pradeep Mohan Kumar. K[2*]
[1,2] Department of Computing Technologies, SRM Institute of Science and Technology,
Kattankulathur 603203, Tamil Nadu, India
[e-mail: alexander845r@gmail.com, pradeepk@srmist.edu.in]
[*]Corresponding author: Pradeep Mohan Kumar. K

Received October 4, 2023; revised December 21, 2023; accepted February 26, 2024;
published March 31, 2024

Abstract

In recent years, the number of devices being connected to the internet has grown enormously, as has the intrusive behavior in the network. Thus, it is important for intrusion detection systems to report all intrusive behavior. Using deep learning and machine learning algorithms, intrusion detection systems are able to perform well in identifying attacks. However, the concern with these deep learning algorithms is their inability to identify a suitable network based on traffic volume, which requires manual changing of hyperparameters, which consumes a lot of time and effort. So, to address this, this paper offers a solution using the extended compact genetic algorithm for the automatic tuning of the hyperparameters. The novelty in this work comes in the form of modeling the problem of identifying attacks as a multi-objective optimization problem and the usage of linkage learning for solving the optimization problem. The solution is obtained using the feature map-based Convolutional Neural Network that gets encoded into genes, and using the extended compact genetic algorithm the model is optimized for the detection accuracy and latency. The CIC-IDS-2017 and 2018 datasets are used to verify the hypothesis, and the most recent analysis yielded a substantial F1 score of 99.23%. Response time, CPU, and memory consumption evaluations are done to demonstrate the suitability of this model in a fog environment.

Keywords: Bayesian optimization, Extended Compact genetic algorithm(eCGA), Genetic Algorithm, Internet of Things (IoT), Intrusion Detection System (IDS), Probabilistic model building genetic algorithm (PMBGA).

1. Introduction

We are instantly awakened by Alexa or Siri with our curtains open, making our daily routine hassle-free. We've got our microwaves set up to begin cooking. We have actually grown better organized as a result of all the equipment being interconnected, and sure, Industry 4.0 has arrived thanks to Internet of Things (IoT) networks [1]. IoT networks are an attempt to connect numerous devices, including those with computing power, sensors, etc. so that devices can communicate with each other for improved functioning over the internet [2]. Although it may seem theoretically straightforward, this is made possible by the development and integration of many significant technologies of this era, including Big Data, Artificial Intelligence, 5G, fog computing, and others. By 2025, it is anticipated that 21.5 billion devices will be connected using these technologies [3]. Better functionality will undoubtedly result from an increase in the number, but it could also pave the way for security-related assaults. The number of threats aimed at Internet of Things (IoT) devices has significantly increased, according to the SonicWall report [4]. According to the research, intrusions and other encrypted threats have increased worldwide by 132% while IoT malware attacks have increased by 77%. Additionally, it is predicted that this number will rise in the ensuing years.

The protection of this network is of the utmost importance because IoT networks become intertwined with regular human activities. Identifying any form of abnormal behavior in the IoT network is crucial and this is done conventionally using firewalls, gateways, and encryption technology. These methodologies do not hold significant results as they are all mechanisms that can’t respond to the newer forms of attacks instantly. The most essential and practical defense for any network is intrusion detection systems (IDS). Whether a connection is cable or wireless, the intrusion detection system plays a crucial role in protecting the network [5]. There are some differences between IoT intrusion detection systems and wired or wireless network IDS. Due to the heterogeneity of IoT networks, there are some reasons why we cannot deploy the same wired or wireless IDS into the IoT network. The computing power, operating systems, protocols, and battery lives of the devices utilized in IoT networks vary [6]. Since each device implements a security system differently, using strong encryption techniques typically requires a lot of CPU power, which most IoT network devices don't have. This necessitates a particular kind of IDS that can operate in spite of heterogeneity and also recognize and respond to an intrusion attempt like a human.

The sorts of attacks detected in the IoT network vary as well based on the IoT network's architectural variations. The following are some specifics of layer-wise attacks in the IoT network. Attacks on the perception layer include service integrity, eavesdropping, jamming, DoS, and Man-In-The-Middle (MITM) attacks. Selective forwarding, Sybil, sinkholes, wormholes, and acknowledgment are examples of network layer attacks. Sniff, Inject, Hijack, DDoS, and Social Engineering are examples of application layer attacks.

In addition, several design-level assaults are frequently seen in IoT networks. IoT-IDS as it is now can recognize and report attacks at both the layer-level and design-level.

1.1 Motivation

The true problem, though, occurs when the adversary creates an entirely new kind of attack. The existing IoT-IDS is unable to respond appropriately and frequently generates false alarms as a result. This calls for a more robust IoT-IDS that can handle newer traffic types more successfully. Various strategies, including data mining, heuristics, ontology-based systems, and more recent innovations like machine learning and deep learning, have been used to create a robust IDS [7]. Machine learning and deep learning are recognized as appropriate techniques to enable the intrusion system to make proactive judgements and think like a person. Support vector machines [8] and random forests [9] are a couple of the machine learning techniques that are employed. Even while using machine learning algorithms produced impressive results in terms of detection accuracy, there are several issues to be aware of when using ML for IoT-IDS. The main worry is that the IoT network is expandable and that the volume of generated traffic will rise as well. Therefore, using an ensemble or any other hybrid technique, as those employed in wireless networks [10], may not be the best course of action because they can lengthen the time needed for detection [11]. Machine learning algorithms take too long when working with high-dimensional data, making them unsuitable for IoT-IDS.

Deep learning is seen as being a critical component when it comes to handling this significant dimension of IoT data. To boost the IoT performance, IDS's several deep learning techniques are applied. Using various deep learning techniques, such as generative adversarial networks (GANs) [11], autoencoders [12], and deep neural networks (DNNs) [13], some of the trickier intrusion attempts can also be effectively handled. Despite the fact that detection accuracy can be quite high, performance or accuracy is tuned by the hyperparameters [14].

The existing machine and deep learning approaches concentrate mainly on improving the detection accuracy of the intrusion detection system, and the accuracy holds significantly good for many of the models. However, the accuracy obtained cannot remain the same because the traffic volume and type are varied. This generates instability in training, hence affecting accuracy. So, while using deep or machine learning models, it is important to set the ideal parameters that can handle the data, irrespective of its volume and type. If the right parameters are not selected, it will have an impact on factors like training duration, computational expense, structure, and prediction accuracy. Thus, the auto-hyperparameter setting is crucial for the intrusion detection data so that the intrusion detection system can act more like a human. To circumvent this problem, the intrusion detection system, if equipped with automatic hyperparameter tuning, can set an ideal parameter for any type of traffic and volume. This is the focus of this work, and this auto-hyperparameter tuning using a genetic algorithm can help the model automatically set the hyperparameter values, irrespective of volume and type.

1.2 Our major contributions

Bayesian or evolutionary algorithms are two methods that can be used to solve the challenge of automatic hyperparameter tuning. The process of optimizing basic neural networks is carried out using some neuro-evolution approaches, including NEAT [15], EPNET [16], and GNARL [17]. One type of evolving network that can optimize the architecture and the hyperparameters for greater performance is DeepNEAT [18]. One of the common issues with the Bayesian approach is the amount of computer resources that are consumed in the process of building different models for different hyperparameters, thus making it unsuitable to run in fog nodes. Some of the algorithms, like catboost [19] and Bayesian optimization [20], tried to address the concern of more computing resources, but parallelism was not achieved. So to address the concerns of having low latency with a higher degree of parallelism, a novel approach using the Bayesian genetic technique is proposed to address the problem of manual ideal hyperparameter setting. The idea comes from the probabilistic model-building genetic algorithm (PMBGA) that operates based on the probability distribution recorded by the Bayesian network. This study expands on DeepNEAT's concept [19], in which the neural arrangement is transformed into a chromosome of the Extended Compact Genetic Algorithm(eCGA), in order to eliminate the issue of manually setting up the hyperparameter values and to provide a superior latency IDS. As a result, the IoT-IDS has evolved with varied degrees of precision and can be used to fog nodes.

The following are the paper's main contributions:

1. Using the Extended Compact Genetic Algorithm, a novel evolutionary deep neural network is suggested for IoT-IDS in the fog environment (eCGA). This will determine which DNN model, in terms of accuracy and latency, should be used in the fog node.
2. The Extended Compact Genetic Algorithm's goal is to minimize the classification error rate (eCGA) with the decrease in the number of parameters. With fewer relevant parameters, the accuracy of the model is increased.
3. A population update that can produce a more accurate model for the following one helps to better direct evolution.

The concept is examined using the CIC-IDS 2017 [21] and CIC-DDoS 2019 datasets, and the proposed methodology is compared to other cutting-edge techniques for higher accuracy and shorter latency.

The remaining portions of the paper are structured as follows:
The various deep learning techniques utilized to improve the IoT-detection IDS's accuracy are discussed in Section 2. The proposed methodology's overall architecture is presented in Section 3. The experimentation process is described in Section 4, the results are examined in Section 5, and the study is finished in Section 6.

2 Background and Related Work

The main goal of this research is to provide an enhanced-performance IoT-IDS with a high detection rate and minimal delay consumption. The state of the art in studies relevant to this study is discussed in this section, along with the present state of knowledge in those fields. Anyone can interfere with our daily activities and put our lives in risk thanks to the interconnected nature of the globe, like in the case of hacking a car and taking control of the steering wheel. To locate and stop these intruders, precautions must be taken, which is of the utmost importance. The development of artificial intelligence has made it possible to create a robust IoT-IDS, which is what is required today. For an IoT-IDS to be effective, machine learning and deep learning are essential. Thus, we first examine the different methods that have been applied thus far to increase detection accuracy in this literature, and then we examine the different ways that have been utilized thus far for parameter adjustment.

2.1. State-of-the-art machine/deep learning models used in detecting intrusions

In some recent studies using machine learning algorithms, voting, bagging, and stacking ensemble procedures have yielded very high detection accuracy [22]. For the identification of various protocol-based botnet attacks in IoT networks, an ensemble technique has been found that combines the three machine learning algorithms decision tree, Naive Bayes, and artificial neural network [23]. Another ensembled strategy suggested in [24] combines a random forest and an average one-dependence estimator that focuses on eliminating attribute dependencies in order to increase detection accuracy while simultaneously lowering the number of false alarms. This made it possible for the incremental learning process and allowed the relevant features for categorization to be captured. With the REPTree, the ensembled technique used with the NSL-KDD data [25] also attempted to lower the frequency of false alarms. The ensembled approach used here is bagging, which requires less time for model building. On top of that, several investigations are conducted employing the fundamental machine learning techniques. The parameters used to gauge the overall performance of the IDS have been the main source of complaints for most machine learning techniques. It was necessary to create appropriate metrics like scalability and throughput to confirm the performance of the IDS in a very big network because measurements like accuracy, recall, and f1 score were insufficient to determine the performance of an IDS in such a huge network. Additionally, the majority of machine learning algorithms contributed to the overfitting issue, leading to the employment of more intricate modelling and deep learning techniques in the development of effective intrusion detection systems.

Due to their improved performance with bigger dimensions, deep learning techniques have become more and more popular. The identification of attacks in IoT-IDS uses supervised, unsupervised, and hybrid deep learning methods. Deep neural network (DNN), convolutional neural network (CNN), and recurrent neural network (RNN) methods are frequently used in supervised settings [26–31]. The methods employed in [26-32] are all direct implementations of supervised learning algorithms; hence, they are not separately explained. Nevertheless, they are all utilized to execute botnet classifications for IoT networks or protocol-based attacks. Deep belief networks, autoencoders, and restricted Boltzmann machines are employed when the unsupervised learning technique is taken into consideration. The performance of the supervised learning setup is enhanced in the work [32] by the introduction of autoencoders. The goal is to train the autoencoders to learn the latent representation of the input data by regularizing them. Today, achieving a scalable IoT-IDS is essential, and this is done utilizing the deep learning method described in [33], where a master-slave network is used to spread the computations to the fog nodes. Two types of representation—local and global—are identified because of the master-slave connection and are learned utilizing the local gated recurrent unit (Local GRU) and multi-head attention mechanism. According to the findings, this strategy is ideally suited to manage the high volume of traffic in IoT-IDS.

Employing hybrid techniques to achieve reduced latency using generative adversarial networks (GAN) is suggested in [11]. In this instance, a specialized GAN dubbed FID-GAN is utilized, which seeks to give computation resources on the fog and internally exploits the idea of federated learning. An encoder was also employed to hasten the loss. A memory remembering technique LSTM for handling the IoT traffic data is suggested in [34]. It is not a new concept to use evolutionary algorithms in intrusion detection systems, and there is a substantial body of research addressing algorithms like Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Artificial Bee Colony Optimization (ABC), Firefly Algorithm (FA), Bat Algorithm (BA), and Flower Pollination Algorithm (FPA). However, the majority of work using evolutionary algorithms aims to collect a minimal set of features to improve the model's classification accuracy. The best characteristics are chosen for the classification task using a method called multi-objective particle swarm optimization with a Levy flight randomization component (MOPSO-Lévy), which is suggested in [35]. The K-nearest neighbors (KNN) algorithm is the classifier employed here. The combined CNN + LSTM architecture proposed in [48] holds the highest detection accuracy of 99.84% for binary and 99.80% for multi-class classification while using the X-IIoTID dataset.

For the best feature set, similar techniques are utilized in [38]. In order to improve classification accuracy for the intrusion detection system, the idea of applying evolutionary techniques is therefore heavily utilized in the feature selection process. Some of the work related to this is proposed in [36,37]. The data imbalance is addressed using the SMOTE technique in [49] so that synthetic data are generated for balancing the attacks.

2.2. State of the art for Auto Parameter Tuning

The idea of employing evolutionary algorithms to optimize the models is still in its infancy. [36] proposes a novel approach to employing the evolutionary algorithm to evolve the classifier. The CNN is a classifier that was developed using an evolutionary approach. This concept is an expansion of the one put forth in [37], which suggests a technique called CoDeepNEAT for creating optimized neural networks. An algorithmic approach for performing hyperparameter tuning is proposed in [50], where the arithmetic optimization algorithm is enhanced to improve the candidate solution's performance. The AutoML technique is used along with the ensembled strategy for the ideal selection of hyperparameters for the supervised setup using the voting mechanism proposed in [51]. Though these approaches propose a strategy for hyperparameter tuning, their major limitation is their inability to handle varied forms and volumes of data, which makes them require human assistance at some stage.

So, this paper suggests an Extended Compact Genetic Algorithm deep neural network to develop the IoT-IDS, similar to the strategy used in [36], where the DNN hyperparameters are updated to a chromosome that takes the Bayesian framework for optimization into account. The measures that identify the linkage learning process are the model's complexity and performance. With the linkage learning process, a higher number of parameters results in a better detection value, which is desirable; nevertheless, a lesser number of parameters in DNN has an effect on performance. The objective is to optimize these metrics so that the parameters and detection rate of the ideal DNN setup are improved. A special fusion of genetic and probabilistic methods, probabilistic model-building genetic algorithms are used for this optimization. In order to provide improved latency, we then supply the fog environment with a set of DNN with the ideal parameters.

3 The Proposed Methodology

Since this methodology combines several techniques to produce a neural network with the best hyperparameters, it is presented here along with all the parts used to construct such a system. The expanded compact genetic algorithm and the CNN's evolution plan are given after the general structure of the feature-map-based CNN is described. A generic architecture is also described because the proposed method's latency reduction needs to be confirmed in the fog network.

3.1 Feature-Map exploitation-based CNN

Understanding the CNN's overall architecture is essential for choosing the optimum hyperparameter and realizing the necessity of a CNN based on feature-map exploitation. Let's thus examine the fundamental design of CNN as well as the justifications for selecting feature-map-exploitation-based CNN for the encoding process. In the area of computer vision, the convolutional neural network (CNN) left a notable impression. Yann Lecun made the initial suggestion, and it was given the name LeNet [39]. The main idea behind CNN is the integration of feature extraction and classification, which makes it appropriate for both text classification and vision issues. **Fig. 1**, shows the general architecture of CNN.

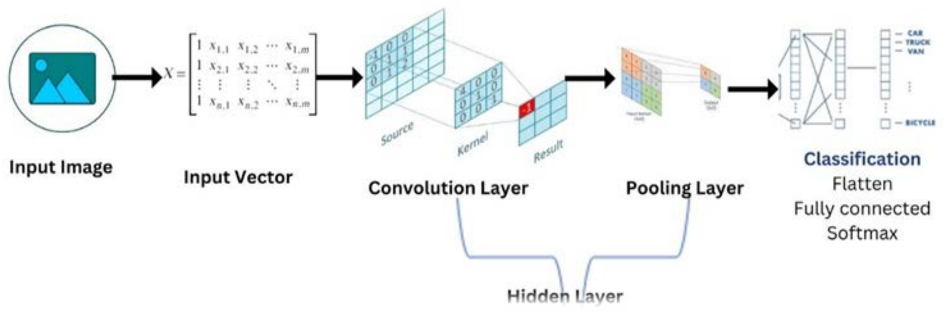

Fig. 1. General Architecture of CNN.

The convolutional neural network (CNN) is made up of three basic components, as illustrated in **Fig. 1**, an input layer, a hidden layer that contains the convolution and pooling layers, and the classification layer, which is often a fully connected layer.

Input Layer: Data's feature vector is transferred to the input layer. The input layer often receives input in the form of pictures. The CIC-IDS data is transformed into an image and provided as an input to CNN, as indicated in [40].

Convolutional Layer: The significant features of the input data must be filtered out by this layer. This is accomplished with the kernel's assistance. The set of learnable parameters is represented by the kernel. Convolution simply refers to a linear operation that is the dot product of the kernel and the input's bounding region. To create a feature map, the kernel is multiplexed in accordance with the stride value. As a result, the dimension of the feature map produced by the convolution process is decreased. The number of convolution layers in a CNN is also flexible, as is the kernel size. In terms of restricted interaction, which lowers the computation's memory demand, parameter sharing, and changes in the input directly affecting the output, this configuration of CNN's convolutional layer makes it superior.

Pooling Layer: Similar to convolution, the pooling layer is in charge of lowering the computing cost by bringing down the input's dimensions. The input will therefore have reduced dimensions when it enters the pooling layer after passing through the convolutional layer; this layer then either does maximum pooling or average pooling. The ability of maximum pooling to prevent total noise activations as well as the fact that it goes through dimensionality reduction for noise removal make it a popular choice.

Fully Connected Layer: This is referred to as the "classification layer," which ideally seeks to construct a non-linear function to determine the relationship between the input features and the output class. This non-linear function is chosen for the learning process based on the categorization type. After the input has been flattened, it is supplied into the feed-forward network, which is made up of several neurons and does backpropagation. After going through several epochs, the learning significantly rises.

The number of distinctive qualities that aid in the identification of attacks needs to be given more focus in our work. So, a modified version of CNN is utilized, one that can automatically extract features and learns them in a hierarchical fashion. In order to comprehend the contribution of a particular input to the convolutional layer, a feature map visualization is performed. The procedures used to identify the traits that help identify classes are described in **Algorithm 1**.

Algorithm 1: Feature Map Exploitation

```
• Input: Image
• Output: Feature Maps
• Define a visualization model
    o For every feature in an image
        ▪ Do
            • Conversion of image to an array
            • Perform the Normalization operation by doing rescaling for the array
        ▪ Iterate
        ▪ Plot the convolutional and pooling layer
    o End for
```

The most dominating elements that facilitate categorization are found through the development of the visualization model for the feature maps. Additionally, creating a model with more characteristics can cause the network to be overfit. As a result, feature selection is essential for solving accurate classification problems, and feature map exploitation-based CNN is used to do this.

Another important thing to be considered for our work is the idea of hyperparameters because this entire work concentrates on auto setting of the hyperparameters so as to obtain an optimal neural network automatically. Some of the hyperparameters that aids in the learning process of the model are as follows:

Learning Rule: As the name suggests, we outline the reasoning behind how the neural network should process data. Learning takes place iteratively in a neural network with continuously updated weights and biases and a prescribed mathematical logic. Hebbian, Perceptron, Delta, Correlation, and Outstar are a few of the well-known learning rules.

Batch Size: The quantity of samples chosen for training in a single iteration is referred to as the batch size. In order for the model to learn and optimize as rapidly as possible, picking the appropriate batch size is important. Again, there is no hard and fast rule for selecting the batch size, and we typically find the best batch size by trial and error.

Dropout Rate: This hyperparameter, as its name suggests, specifies the likelihood that a node should be dropped. Given that it avoids the overfitting issue, this is a crucial part of neural network training. A whole new neural configuration is created from the parent network by defining the dropout probability.

Activation Function: There should be a system that determines whether a specific input node is important for training or not since it is crucial for the neural network to learn just the relevant input throughout the learning process. The "activation function" is a mathematical notation used to make this determination that transforms the input. The activation function's purpose is to enable a node to get the output for a set of input values.

Learning rate: The "learning rate" refers to how frequently the weights of the neural network are updated. The weights are updated significantly if a higher learning rate is selected.

A neural arrangement can be described with hundreds of additional hyperparameters, such as mini-batch, in addition to the ones just mentioned. However, the most popular hyperparameters that, when improved, can considerably boost the network's performance are covered. Despite the fact that these rules are frequently created by hand, in our study we put the emphasis on allowing our feature-map exploitation-based CNN to evolve and then let it choose the type of optimum learning rule, rate, dropout rate, etc. Suppose if the input is defined by the weight(W), height (H) and the number of channels(C), the output is obtained with the filter(F) and stride(S). Thus, the output feature is obtained using the Equation (1) is

$$\left[\frac{W-F}{S+1}\right] \times \left[\frac{H-F}{S+1}\right] \times layerfilters \tag{1}$$

3.2 Evolutionary Strategy- Motivation

When examining the CNN feature map's structure, it becomes evident that selecting the appropriate amount of convolution layers, pooling layers, and fully linked layers was challenging. It is also a laborious effort to select the appropriate hyperparameter values for the feature-map exploitation-based CNN unless there is prior experience working with a variety of use cases. There are several justifications for why we must choose the ideal hyperparameter value. Detection accuracy is improved by a deep neural network's capacity for learning. However, as it is deepened more, it typically overfits the data, which will directly affect the training time and processing resources.

Two essential components of the IoT-IDS are increased compute power and accurate detection. Since the values of the hyperparameters have a significant impact on the computation resource and detection accuracy, this needs to be automatically configured to optimal so that, when it is deployed in a fog environment, it can create a lower latency. This is the primary driver behind the CNN-based feature-map exploitation conversion to a chromosome.

The process of building the evolution model follows an encoding procedure in which all the components of the neural network have to be converted in such a way to be converted as a chromosome. Since our problem involves the usage of real numbers, we perform a value encoding to decide on the number of genes in our chromosome. **Fig. 2**, shows the gene details of the chromosome.

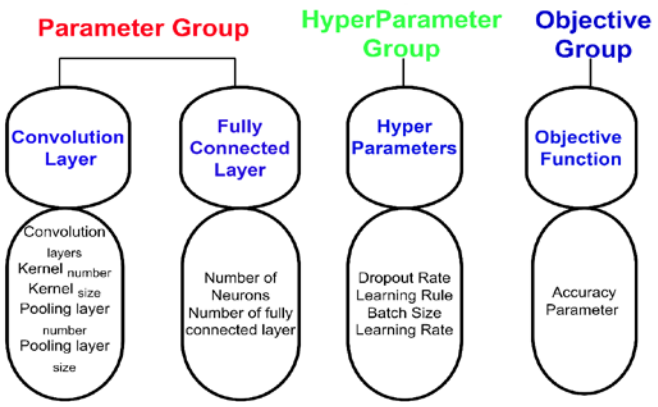

Fig. 2. Structure of the Genes in the Chromosome

The probabilistic model genetic algorithm's overall chromosome structure thus consists of 13 genes and the concept of evolving the CNN model to the chromosome is borrowed from this work [36]. No significant changes have been made because the goal of our research was to adapt the evolutionary algorithm to deal with large-scale, multidimensional data in the IoT environment by using the Bayesian approach. The gene structure thus resembles the one described in [36], with the addition of the feature map CNN being the only variation made at this level. The simplified notations and real value encoding used for the evolution process are displayed in **Table 1**. Our dataset contains multiclass scenarios, hence a SoftMax function was used as the activation function.

Table 1. Gene Encoded values [36]

Genes of the Chromosome	Encoded Values	Notation
Convolution $_{\text{layers}}$	1,2,3	Con_L
Kernel $_{\text{number}}$	2,4,8,16,32,64,128,256	K_N
Kernel $_{\text{size}}$	2,3,4,5,6,7,8,9	K_{size}
Pooling layer $_{\text{number}}$	0- No pooling, 1- Pooling	P_{Decide}
Pooling layer $_{\text{size}}$	2,3,4,5,6,7,8,9	P_{size}
Number of Neurons	8,16.32,64,128,256,512,1024	N_{number}
No. fully connected layers	3	$F_{connected}$
Dropout Rate	0,0.25,0.5	D_{rate}
Learning Rule	Adam, Adamax	L_{rule}
Batch Size	32,64,128,256	B_{size}
Learning Rate	0.1,0.01,0.001,0.0001	L_{rate}
Accuracy	Needs to be Optimized	
Parameter	Needs to be Optimized	

3.3 Probabilistic Model Building Genetic Algorithm-Extended Compact Genetic Algorithm(eCGA)

Having defined the gene's initial population and the objective function that needs to be optimized, it is very clear that our objectives are multifunctional. As a result, it is critical to optimize the feature map CNN for multiple objectives. It is also known that when the model complexity increases, there is a possibility of a decrease in detection accuracy because of the vanishing gradient or the overfitting issue, whereas a decrease in model complexity increases detection accuracy. So, these two conflicting objective functions need to be defined as an objective function that can be handled by the genetic algorithm. There are only two approaches available to do this:

1. The value of each objective function can be calculated by combining all the objective functions into one and using a weighted summation.
2. The best answer is chosen from a set of optimal solutions using a meta-heuristic technique.

Since most of the meta heuristic approaches uses the evolutionary algorithms, the second technique is used for optimizing both the objective functions simultaneously. In particular, the approach used here is similar to the VEGA [41]. **Algorithm 2** explains the genetic algorithm approach for optimizing the multi objective problem. The procedure adopted here is the entire population is made as two and two solutions are obtained. For every solution, depending on the objective function value a fitness value is given. Based on these values, the best solutions are then chosen for the cross over and mutation.

Algorithm 2: Multi-objective feature Map based CNN procedure

```
Input:
    o  N: Population {$P_1, P_2, P_3, P_4, P_5$ ........... $P_N$}
    o  K: Subpopulation (N/K) {$P_1, P_2, P_3, P_4, P_5$ ........... $P_K$} {$P_1, P_2, P_3, P_4, P_5$ ........... $P_K$}
    o  Z: Fitness Value
Output:
    o  Initial population is populated randomly $P_4$ and the iteration is at i=0
           ▪  if ($P_4$ = stopping condition)
                  •  random sort (population)
    o  for all the solutions in K: Subpopulation
```

```
            ▪ for i= 1+(K-1) $P_K$ ...........K $P_K$
                  • set fitness value
                  • Based on fitness
                  • Select K solution for all the subpopulation
                  • Perform crossover and mutation for subpopulation
            ▪ i=i+1
            ▪ end for
      o else
            ▪ continue
      o end for
      o end
```

The next step is to optimize the given genetic algorithm using CNN after establishing a multi-objective optimization function as a genetic algorithm. The extended genetic compact algorithm (eCGA), which is combined with this multi-objective function conversion, is then used to optimize the neural network. The choice of eCGA is made in this work because the encoding of eCGA is using the Bayesian network, and if the traditional eCGA, which uses probabilistic polling for subset generation, is modified to be deterministic, the computation resources will be reduced and it can process faster.
Now that these theoretical facts have been taken into account, Algorithm 3 describes how to use the eCGA to produce the best feature map for the CNN. Because linkage learning occurs by selecting a better probability distribution, this algorithm is capable of both evolutionary and probabilistic optimization. The value that indicates a good distribution in this situation is the minimal description length. Even higher-order relationships can be simulated with eCGA because it is based on multivariate factorizations. With the help of the model complexity and the compressed population complexity, this technique achieves linkage learning, a method for identifying the major dependencies between the variables that make up the problem. For a cluster that is first initialized randomly, it assumes that all are equal and seeks to obtain an optimal probability distribution that minimizes the probability model and population. A good distribution has the smallest description length. Marginal distributions combine to generate the marginal product model known as eGCA.

Algorithm 3: Extended Genetic Compact Algorithm(eCGA) based CNN (eCGA-CNN)

```
  Input:
      o N<-Number of Population {$P_1, P_2, P_3, P_4, P_5$ ........... $P_N$}
      o T<-Neighbourhood Size
      o $H_{max}$ <- Maximum generation
      o Weight vectors with uniform spread N {$w_1, w_2, w_3, w_4$.................. $w_t$ }
      o Training and Testing set
  Output:
      o The final Population P
      o Identify the neighbourhood value of T for all subproblems (Algorithm 1)
      o Generate the population
      o For each individual in the population build a feature-map CNN
            ▪ Initialize the clusters randomly
            ▪ Nelder Mead Simplex search
            ▪ Compute energy of each cluster
                  o if (cluster = converged)
                        ▪ end
                  o else
                        ▪ Build minimum description model
```

```
                    ▪ Create new cluster
    o Iterate
    o for i = 1 to $H_{max}$ do
        ▪ for j= 1 to N do
            • Reproduction: select random values from the cluster and get the
              new solution
            • Evaluate CNN
            • Update old cluster with new cluster
        ▪ End for
    o End for
    o End for
```

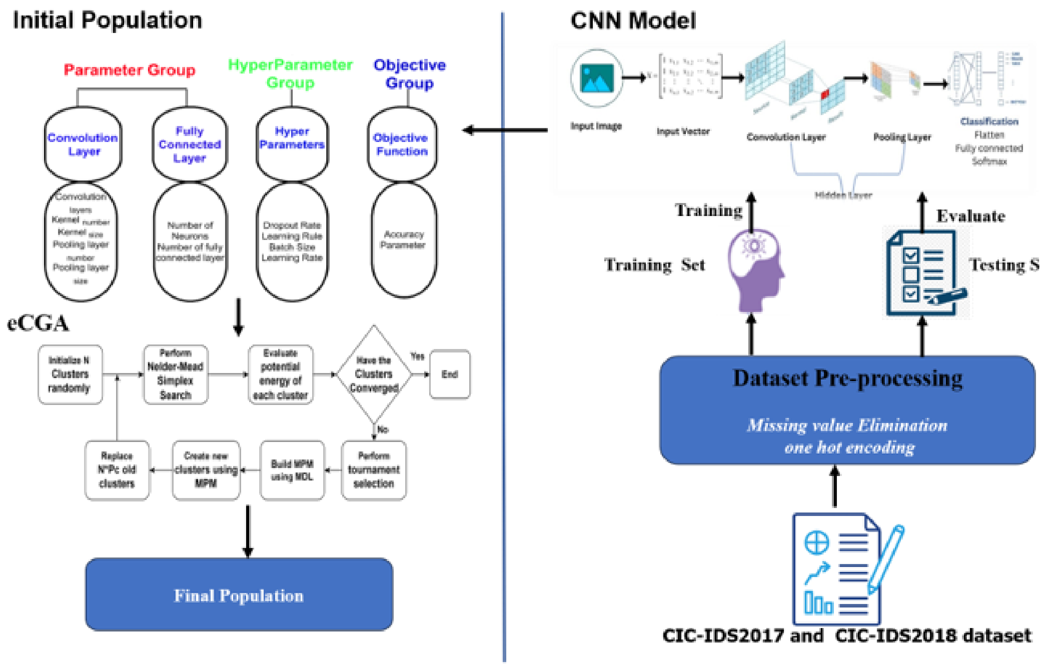
Initial Population
Parameter Group
HyperParameter Group
Objective Group
Convolution Layer
Fully Connected Layer
Hyper Parameters
Objective Function
Convolution layers Kernel number Kernel size Pooling layer number Pooling layer size
Number of Neurons Number of fully connected layer
Dropout Rate Learning Rule Batch Size Learning Rate
Accuracy Parameter
eCGA
Initialize N Clusters randomly
Perform Nelder-Mead Simplex Search
Evaluate potential energy of each cluster
Have the Clusters Converged
Yes
End
No
Perform tournament selection
Build MPM using MDL
Create new clusters using MPM
Replace N*Pc old clusters
Final Population
CNN Model
Input Image
Input Vector
Convolution Layer
Pooling Layer
Classification
Flatten
Fully connected
Softmax
Hidden Layer
Training
Evaluate
Training Set
Testing Set
Dataset Pre-processing
Missing value Elimination
one hot encoding
CIC-IDS2017 and CIC-IDS2018 dataset

Fig. 3. System Model of eCGA based CNN

The overall system diagram including the training phase and evolutionary phase is showcased in **Fig. 3**. The general steps in this process include one hot encoding preprocessing, selecting the best features for the model, and feeding the training and testing sets to the CNN to create genes with the ability to automatically tune the hyperparameters using the eCGA algorithm.

3.4 Achieving Low latency in Fog Environment

A low latency detection service is also being worked on as part of the development of an IoT-IDS, in addition to increasing detection accuracy. In a typical network intrusion detection system, the IDS is often used on a potent server where the computational power and resources are immense. But since the devices and their computing capability are so varied in an IoT context, this is useless. According to IDC, the majority of data handled globally will be processed in a fog environment, necessitating the use of a security detective mechanism in a fog environment. This is important since low latency is guaranteed by fog computing, which means our suggested IoT-IDS will likewise operate with low latency.

As a result of the suggested linkage learning model's high computing requirements, the entire model training procedure cannot be implemented in a fog environment. The main problem is that it could require a lot of training time, which prevents it from meeting the low latency criteria. A cloud server might be utilized to conduct the training process. After that, the learned models are applied to fog devices that can provide real-time detection services with minimal latency.

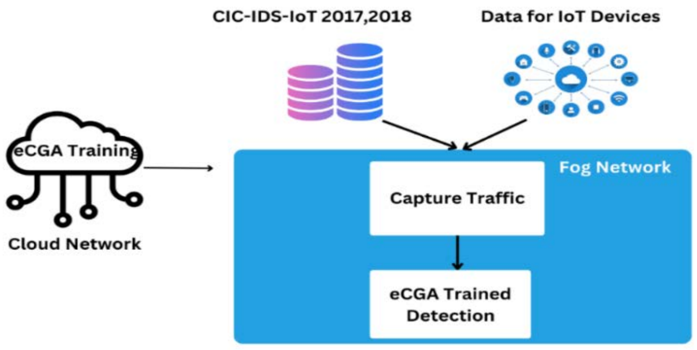

Fig. 4. Incorporation of the proposed hyperparameter tuned model into the Fog Network

Fig. 4, shows the way the fog network uses the trained model for detecting intrusive behavior. As illustrated in the figure, traffic generated by various IoT devices and received in the fog network is handled by multiple subcomponents to determine the traffic's location. The proposed methodology is specifically deployed inside the detection module, where the regular IDS runs and takes care of the classification of the traffic.

4. Experimental configuration

This section gives a quick overview of the technology and setting that were used for the studies. The datasets utilized in the trials are then supplied, along with instructions for pre-processing them. In order to assess the performance of the proposed system, various performance measures are finally introduced.

4.1 Experimental Setup

With help from Keras, Tensorflow, PyGAD, Numpy, and Pandas, the methodology is implemented using the Python 3.11.2 programming environment. The Ubuntu operating system and AMD Ryzen 5000 Series 7 hardware were used to conduct this experiment. The Windows 10 operating system, an Intel Core i5 CPU, 4GB of RAM, and the iFogSim tools are used to virtualize the fog node.

4.2 Dataset Description

Though there is a wide variety of intrusion datasets are available for the public, to carry out this work specific two variations of the IoT-IDS dataset CIC-IDS2017 and CIC-IDS2018 dataset. The Canadian Institute for Cybersecurity dataset 2017 and 2018 is chosen because of its closeness towards the real IoT traffic data. This dataset is built using the five days normal and attack patterns observed in the Canadian institute of Cybersecurity. The features present in this dataset is 78 and the overall distribution of this dataset is shown in **Fig. 5**, along with the percentage of each attack is presented in **Fig. 6, 7, 8,** and **9**.

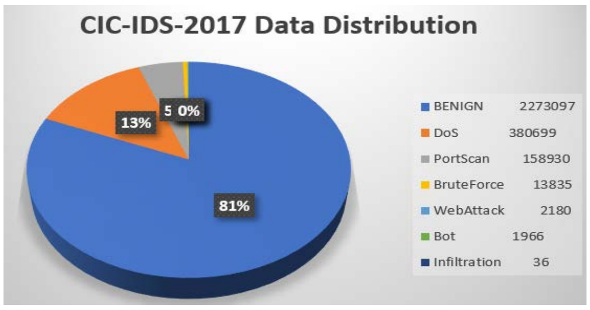

Fig. 5. Class Distribution plot for the CIC-IDS 2017 Dataset showing the number of instances for each class.

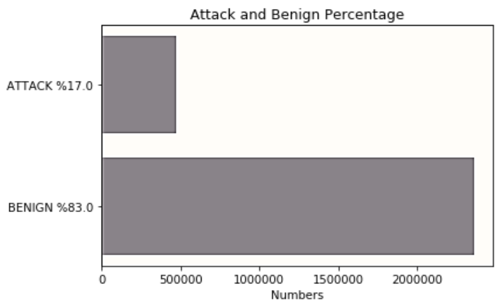

Fig. 6. Distribution percentage of the major class labels Benign and Attack

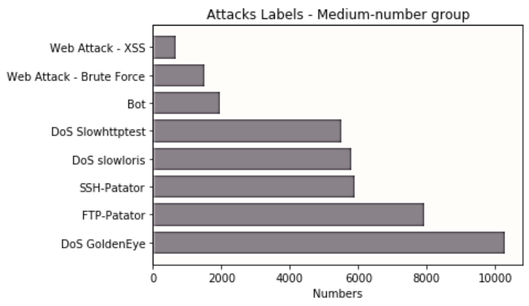

Fig. 7. Number of attacks in each of the attack class

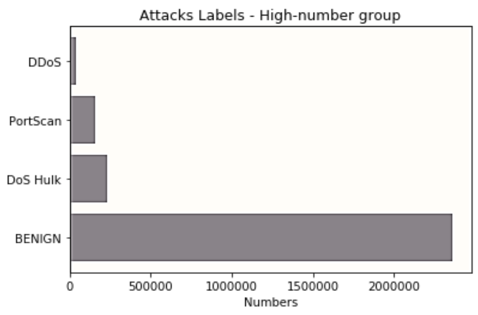

Fig. 8. Group of Attacks that are holding the highest number of attack instances

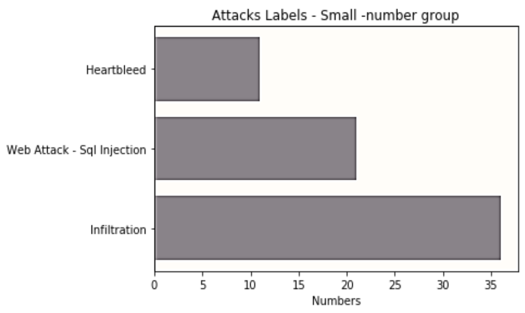

Fig. 9. Group of Attacks that are holding the smallest number of attack instances

More classes are included in the CIC-IDS 2018 dataset, which is organized daily. Data from event logs and network traffic are collected every day. This has 80 features, and **Fig. 10**, depicts the class distribution map of the same.

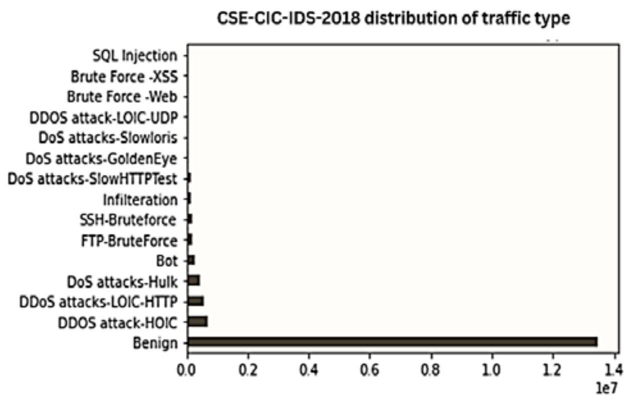

Fig. 10. Class Distribution plot for the CIC-IDS 2018 Dataset

Table 2, includes the same information in tabular style along with the number of each traffic source and its percentage. These simulated datasets were chosen because of their authenticity; even though the dataset demonstrates class imbalance, it can be useful to do a meta-analysis using them. Also, they are more similar to actual IoT traffic statistics.

Table 2. Multi Class Label count values and the percentage of distribution

Class	Label	Percentage
Benign	13390249	0.829776
DDOS attack-HOIC	686012	0.042511
DDoS attacks-LOIC-HTTP	576191	0.035706
DoS attacks-Hulk	461912	0.028624
Bot	286191	0.017735
FTP-BruteForce	193354	0.011982
SSH-Bruteforce	187589	0.011625
Infilteration	160639	0.009955
DoS attacks-SlowHTTPTest	139890	0.008669
DoS attacks-GoldenEye	41508	0.002572
DoS attacks-Slowloris	10990	0.000681
DDOS attack-LOIC-UDP	1730	0.000107
Brute Force-Web	611	0.000038
Brute Force-XSS	230	0.000014
SQL Injection	87	0.000005

4.3 Dataset Pre-processing

In terms of data preparation, only the most elementary pre-processing removing null values and non-numeric characters is carried out for the two datasets. 2867 records from the CIC IDS 2017 dataset were found to be useless and removed. Also, according to [42], there were 15 classes present that were reduced to 7 classes. The infinite values in the CSECICIDS2018 dataset were all converted to null values. Large correlations between a number of features were also present in the IDS2018 dataset; the feature map CNN will feature engineer these features. Some of the strongly associated traits are displayed in **Table 3**. **Fig. 11**, shows the feature map retrieved features.

This job is performed in a neural network; hence it is necessary to convert the categorical form to a numeric representation. Hence, the conversion only uses one hot encoding. Then both the datasets are split into 60:20:20 for training, testing, and validation.

Table 3. Strong Correlated features requiring feature map reduction

	Variable_1	Variable_2	Correlation
0	tot_fwd_pkts	subflow_fwd_pkts	1.000000
1	fwd_psh_flags	syn_flag_cnt	1.000000
2	totlen_fwd_pkts	subflow_fwd_byts	1.000000
3	fwd_pkt_len_mean	fwd_seg_size_avg	1.000000
4	tot_bwd_pkts	subflow_bwd_pkts	1.000000
5	fwd_urg_flags	cwe_flag_count	1.000000
6	bwd_pkt_len_mean	bwd_seg_size_avg	1.000000
7	totlen_bwd_pkts	subflow_bwd_byts	1.000000
8	flow_iat_min	fwd_iat_min	0.999996
9	flow_iat_max	fwd_iat_max	0.999994
10	rst_flag_cnt	ece_flag_cnt	0.999988
11	flow_duration	fwd_iat_tot	0.999986
12	flow_iat_std	fwd_iat_std	0.999981
13	flow_iat_mean	fwd_iat_mean	0.999963
14	subflow_fwd_pkts	fwd_act_data_pkts	0.999189
15	tot_fwd_pkts	fwd_act_data_pkts	0.999189
16	bwd_header_len	subflow_bwd_pkts	0.997798
17	tot_bwd_pkts	bwd_header_len	0.997798

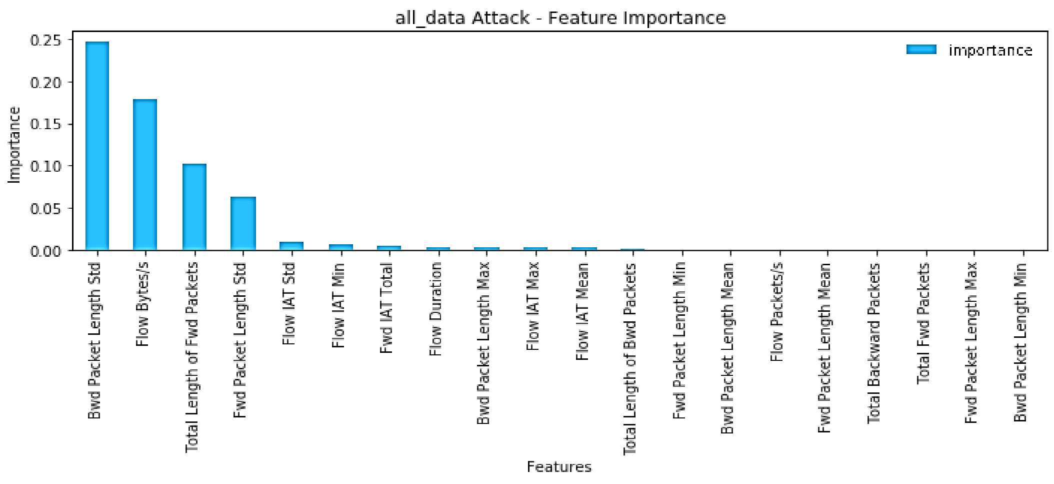

Fig. 11. Feature Map extracted features

5. Result Analysis

The results are used to analyze the proposed system in two different ways.

- To demonstrate the suggested system's superiority over other deep learning methods in terms of detection accuracy.
- Some of the best-evolved model performances can be found when objective values are included and when the performance of the model changes as the hyperparameters evolve.

Some of the quantifiable values are displayed in this section to demonstrate how well the suggested model works.

5.1 Population Analysis

This population study is done to highlight the subset of present generational solutions. It is crucial to comprehend how evenly the individuals were dispersed throughout the first generation and in the last generation in order to say that the proposed methodology succeeded successfully. The fact that the individuals should be evenly dispersed suggests there hasn't been any premature convergence The CIC-IDS-2017 and 2018 beginning and final generation populations are plotted for the multiclass problem taking this into account. The results are given in **Fig. 12** and **13**. As observed in **Fig. 12** and **13**, the initial population shows the distribution irregularly exhibiting a high detection error rate. This value is significantly decreased in the final generation. A similar type of distribution is observed for the CIC-IDS-2018 data as well.

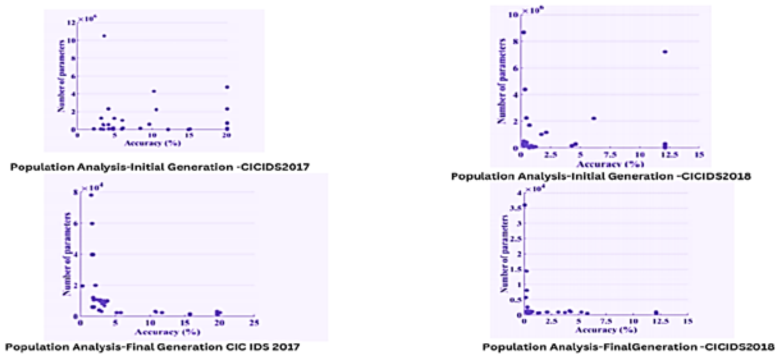

Fig. 12. Distribution of Individuals in the initial and final generation CIC-IDS-2017 and 2018

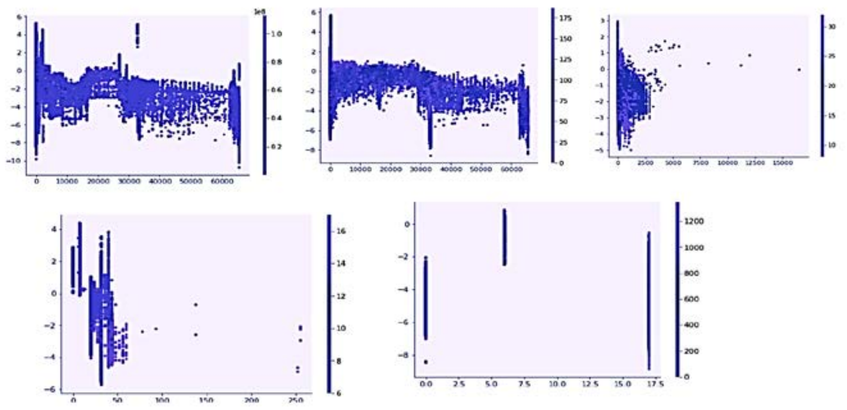

Fig. 13. Distribution of Individuals in the initial and final generation CIC-IDS-2018

The number of parameters and error rate have increased since the distribution is noticeably greater. The extreme values for both datasets are displayed in **Table 4**, for a quantitative examination.

Table 4. Quantitative values for extreme cases

Dataset	**Factors**	**Initial Generation**		**Final Generation**	
CIC-IDS-2017		parameters	Error rate	Parameters	Error rate
	Number of Parameters	1320	16.21	1520	0.38
	Error rate (Minimum)	58,525	0.15	21,534	0.01
	Average	1,09,785	8.32	15,343	3.21
CIC-IDS-2018	Number of Parameters	1682	14.21	289	11.26
	Error rate (Minimum)	78,352	0.11	1072	0.24
	Average	99,875	7.23	882	4.98

As shown in **Table 4**, the final generation error rate and the number of parameters is significantly lower that shows with the proposed approach the detection performance and the complexity are greatly increased. The average number of parameters at both the initial and final generation shows a decreased error rate when the number of parameters is significantly reduced. The point that needs to be noted here is the effect of the number of parameters on the detection accuracy. **Fig. 14**, shows the distribution plot of the error.

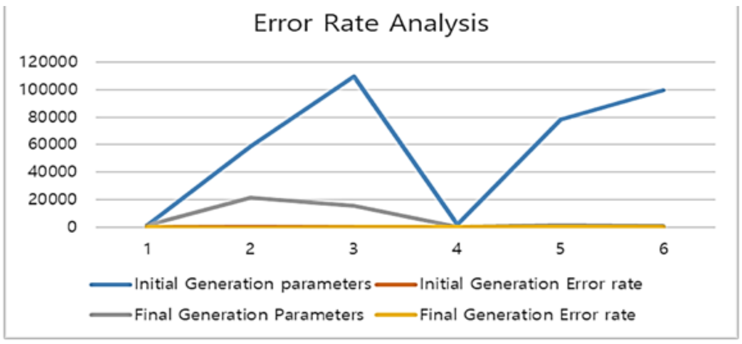

Fig. 14. Distribution plot for the error generated without genetic algorithm and with genetic algorithm

From **Fig. 14**, it is clear that the error rate is significantly lower at the final generation of parameters. The grey line indicating the final generation parameters is lower than that of the blue line that is showing the initial generation parameters and the similar results are observed for the final generation error rate as well. This visualization is done based on the values that we have obtained from **Table 4**.

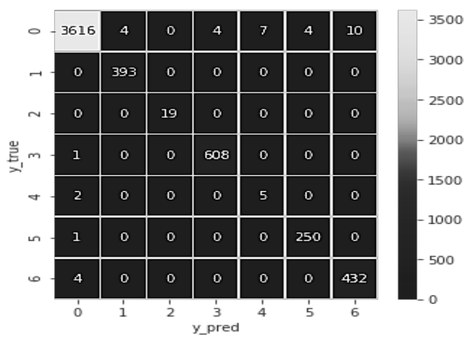

Fig. 15. Confusion Matrix of the CIC-IDS-2017 dataset

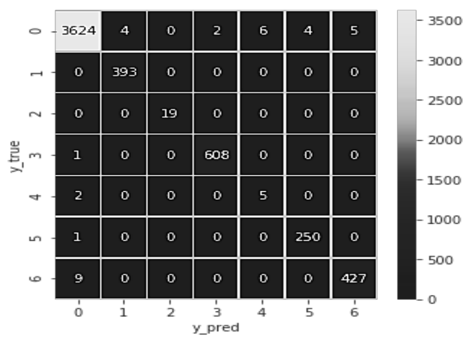

Fig. 16. Confusion Matrix of the CIC-IDS-2018 dataset

The confusion matrix for the best-performed individual is shown in **Fig. 15, 16** and **Table 5, 6** for CIC IDS 2017 and 2018 data. The detection accuracy outperformed in both the dataset with a value above 99%.

Table 5. Quantitative values for best performed case in CIC IDS2017

	precision	recall	f1-score	support
0	1.00	0.99	0.99	3645
1	0.99	1.00	0.99	393
2	1.00	1.00	1.00	19
3	0.99	1.00	1.00	609
4	0.42	0.71	0.53	7
5	0.98	1.00	0.99	251
6	0.98	0.99	0.98	436
accuracy			0.99	5360
macro avg	0.91	0.96	0.93	5360
weighted avg	0.99	0.99	0.99	5360

Table 6. Quantitative values for best performed case in CIC IDS2018

	precision	recall	f1-score	support
0	1.00	0.99	1.00	3645
1	0.99	1.00	0.99	393
2	1.00	1.00	1.00	19
3	1.00	1.00	1.00	609
4	0.45	0.71	0.56	7
5	0.98	1.00	0.99	251
6	0.99	0.98	0.98	436
accuracy			0.99	5360
macro avg	0.92	0.95	0.93	5360
weighted avg	0.99	0.99	0.99	5360

5.2 Meta-Analysis

To show the superiority of the proposed system to that of the other deep learning approaches that used manual hyperparameter tuning is compared along the detection accuracy values. **Table 7**, makes it abundantly evident that the proposed methodology outperformed in terms of optimization as well as detection accuracy, with noteworthy outcomes.

Table 7. State-of-the-art comparative analysis with detection accuracy for CIC-IDS-2017

Method	Accuracy	Precision	Recall	F1
GRU [43]	98.53	77.59	74.23	75.66
LSTM [44]	98.89	89.22	74.79	77.05
AE-ANN [45]	98.13	54.20	53.31	54.74
SDAE-SVM [46]	95.38	52.01	44.43	45.25
SAE-DNN [47]	97.66	68.85	59.96	60.65
Proposed	**99.24**	**99.24**	**99.24**	**99.23**

5.3 Latency Analysis

The pretrained eCGA-based model must be able to execute in the fog node for the claim of lower latency to be valid, and the amount of time it takes to react to attacks must also be confirmed. These fog nodes are simulated using the iFogSim simulation environment, and the pretrained models are made to operate in those nodes, in order to test this. The CPU and memory consumption for the fog node when these nodes attempted to do this ranged from 10% to 20%, demonstrating the fog nodes' ability to support this methodology, which will result in lower latency. **Fig. 17** and **18**, shows the comparison of latency for cloud and fog nodes.

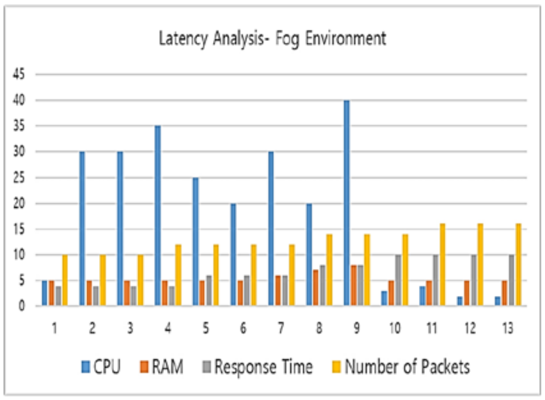

Fig. 17. Latency analysis for the Fog environment in terms of the QoS parameters

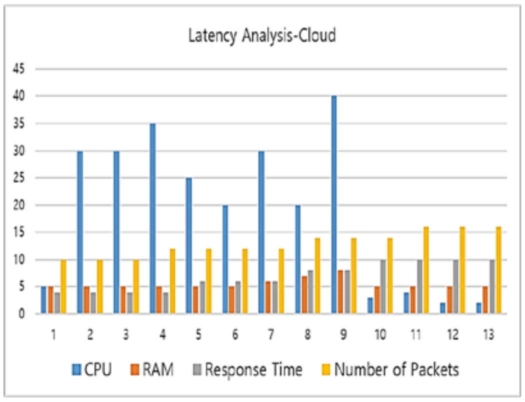

Fig. 18. Latency analysis for the Cloud environment in terms of the QoS parameters

Thus, with the plotting of the population at the initial and final stages of population denotes a substantial reduction in the error rate and this clearly showed the relationship between the number of parameters and the error. With the reduction in error, the detection accuracy is improved and that is compared to the state-of-the-art for its superiority and the latency analysis exhibits the suitability of this approach at the fog node.

6. Conclusion

The method proposed in this study better automates the hyperparameter selection by utilizing the CNN feature map and an enhanced genetic compact algorithm. For executing the categorization of IoT-IDS network traffic, the genetic algorithm is used to achieve the goal of obtaining a superior neural network with automatic adjustments. IoT traffic is categorized using the feature map CNN, which is subsequently encoded as a chromosome for hyperparameter tweaking. With the CIC-IDS 2017 and 2018 datasets, a greater detection capacity is attained with this arrangement. For improved solution selection, the fitness value derived from the objective functions is used. The best performers within the population are selected to provide quantitative values for detection accuracy. By the results, we were able to show that the suggested procedure is effective at identifying the optimal choice. In terms of result analysis, the error rate is lower following population analysis, the latency is confirmed by contrasting the cloud and fog settings, and in terms of response time, it is clear that the fog requires less time than the cloud.

Despite the automated hyperparameter adjustment capabilities of the proposed model, this system is evaluated on balanced data due to sample availability. These model parameters tuning findings must be confirmed on a significantly unbalanced dataset, though. Adversarial synthetic data production is required to include the data's imbalance. The next goal of this study is to verify the integration of Jacobian saliency approaches with SMOTE and GAN networks to produce adversarial samples instantaneously and create such a massive amount of diverse data.

References

[1] Khan. R, Khan. S.U, Zaheer. R, and Khan. S, "Future internet: the internet of things architecture, possible applications and key challenges," in *Proc. of 2012 10th international conference on frontiers of information technology*, pp. 257-260, Dec. 2012. Article (CrossRef Link)

[2] Perera. C and Vasilakos. A.V, "A knowledge-based resource discovery for Internet of Things," *Knowledge-Based Systems*, 109, pp. 122-136, 2016. Article (CrossRef Link)

[3] Al-Fuqaha. A, Guizani. M, Mohammadi. M, Aledhari. M, and Ayyash. M, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE communications surveys & tutorials*, 17(4), pp. 2347-2376, 2015. Article (CrossRef Link)

[4] Mid-Year Update: SonicWall Cyber Threat Report, 2022. [Online]. Available: https://www.sonicwall.com/resources/white-papers/mid-year-2022-sonicwall-cyber-threat-report/

[5] Forestiero. A, "Metaheuristic algorithm for anomaly detection in Internet of Things leveraging on a neural-driven multiagent system," *Knowledge-Based Systems*, 228, p. 107241, 2021. Article (CrossRef Link)

[6] Hassan. M, Huda. S, Sharmeen. S, Abawajy. J, and Fortino. G, "An adaptive trust boundary protection for IIoT networks using deep-learning feature-extraction-based semisupervised model," *IEEE Transactions on Industrial Informatics*, 17(4), pp. 2860-2870, 2021. Article (CrossRef Link)

[7] Vijayakumar. D.S and Ganapathy. S, "Machine Learning Approach to Combat False Alarms in Wireless Intrusion Detection System," *Comput. Inf. Sci*., 11(3), pp. 67-81, 2018. Article (CrossRef Link)

[8] Hindy. H, Bayne. E, Bures. M, Atkinson. R, Tachtatzis. C, and Bellekens. X, "Machine learning based IoT intrusion detection system: An MQTT case study (MQTT-IoT-IDS2020 dataset)," in *Proc. of 12th International Networking Conference: INC*, pp. 73-84, 2021. Article (CrossRef Link)

[9] Al-Omari. M, Rawashdeh. M, Qutaishat. F, Alshira'H. M, and Ababneh. N, "An intelligent tree-based intrusion detection model for cyber security," *Journal of Network and Systems Management*, 29, pp. 1-18, 2021. Article (CrossRef Link)

[10] Vijayakumar. D.S and Ganapathy. S, "Multistage ensembled classifier for wireless intrusion detection system," *Wireless Personal Communications*, 122(1), pp. 645-668, 2022. Article (CrossRef Link)

[11] de Araujo-Filho. P.F, Kaddoum. G, Campelo. D.R, Santos. A.G, Macêdo. D, and Zanchettin. C, "Intrusion detection for cyber–physical systems using generative adversarial networks in fog environment," *IEEE Internet of Things Journal*, 8(8), pp. 6247-6256, 2021. Article (CrossRef Link)

[12] Chen. Y, Ashizawa. N, Yeo. C.K, Yanai. N, and Yean. S, "Multi-scale self-organizing map assisted deep autoencoding Gaussian mixture model for unsupervised intrusion detection," *Knowledge-Based Systems*, 224, pp. 107086, 2021. Article (CrossRef Link)

[13] Ravi. N and Shalinie. S.M, "Semisupervised-learning-based security to detect and mitigate intrusions in IoT network," *IEEE Internet of Things Journal*, 7(11), pp. 11041-11052, 2020. Article (CrossRef Link)

[14] Suganuma. M, Shirakawa. S, and Nagao. T, "A genetic programming approach to designing convolutional neural network architectures," in *Proc. of the genetic and evolutionary computation conference*, pp. 497-504, 2017. Article (CrossRef Link)

[15] Stanley. K.O and Miikkulainen. R, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, 10(2), pp. 99-127, 2002. Article (CrossRef Link)

[16] Yao. X and Liu. Y, "A new evolutionary system for evolving artificial neural networks," *IEEE transactions on neural networks*, 8(3), pp. 694-713, 1997. Article (CrossRef Link)

[17] Angeline. P.J, Saunders. G.M, and Pollack. J.B, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE transactions on Neural Networks*, 5(1), pp. 54-65, 1994. Article (CrossRef Link)

[18] Miikkulainen. R, Liang. J, Meyerson. E, Rawal. A, Fink. D, Francon. O, Raju. B, Shahrzad. H, Navruzyan. A, Duffy. N, and Hodjat. B, "Evolving deep neural networks," in *Artificial intelligence in the age of neural networks and brain computing*, Academic Press, 2019, pp. 293-312. Article (CrossRef Link)

[19] Nayak. J, Naik. B, Dash. P.B, Vimal. S, and Kadry. S, "Hybrid Bayesian optimization hyper tuned catboost approach for malicious access and anomaly detection in IoT nomaly framework," *Sustainable Computing: Informatics and Systems*, 36, p. 100805, 2022. Article (CrossRef Link)

[20] Masum. M, Shahriar. H, Haddad. H, Faruk. M.J.H, Valero. M, Khan. M.A, Rahman. M.A, Adnan. M.I, Cuzzocrea. A, and Wu. F, "Bayesian hyperparameter optimization for deep neural network-based network intrusion detection," in *Proc. of 2021 IEEE International Conference on Big Data*, pp. 5413-5419, Dec. 2021. Article (CrossRef Link)

[21] Sharafaldin. I, Lashkari. A.H, and Ghorbani. A.A, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. of ICISSp*, 1, pp. 108-116, 2018. Article (CrossRef Link)

[22] Ren. Y, Zhang. L, and Suganthan. P.N, "Ensemble classification and regression-recent developments, applications and future directions," *IEEE Computational intelligence magazine*, 11(1), pp. 41-53, 2016. Article (CrossRef Link)

[23] Jabbar. M.A and Aluvalu. R, "RFAODE: A novel ensemble intrusion detection system," *Procedia computer science*, 115, pp. 226-234, 2017. Article (CrossRef Link)

[24] Gaikwad. D.P and Thool. R.C, "Intrusion detection system using bagging ensemble method of machine learning," in *Proc. of 2015 international conference on computing communication control and automation*, pp. 291-295, Feb. 2015. Article (CrossRef Link)

[25] Moustafa. N, Turnbull. B, and Choo. K.K.R, "An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things," *IEEE Internet of Things Journal*, 6(3), pp. 4815-4830, 2018. Article (CrossRef Link)

[26] McLaughlin. N, del Rincon. J.M, Kang. B, Yerima. S, and Miller. P, "Deep android malware detection," in *Proc. of the Seventh ACM on Conference on Data and Application Security and Privacy*, pp. 301-308, 2017. Article (CrossRef Link)

[27] Maghrebi. H, Portigliatti. T, and Prouff. E, "Breaking cryptographic implementations using deep learning techniques," in *Proc. of Security, Privacy, and Applied Cryptography Engineering: 6th International Conference, SPACE*, pp. 3-26, Dec. 2016. Article (CrossRef Link)

[28] Brun. O, Yin. Y, Gelenbe. E, Kadioglu. Y.M, Augusto-Gonzalez. J, and Ramos. M, "Deep learning with dense random neural networks for detecting attacks against IoT-connected home environments," *Procedia Computer Science*, vol. 134, pp. 458-463, Feb. 2018. Article (CrossRef Link)

[29] Diro. A.A and Chilamkurti. N, "Distributed attack detection scheme using deep learning approach for Internet of Things," *Future Generation Computer Systems*, 82, pp. 761-768, 2018. Article (CrossRef Link)

[30] Torres. P, Catania. C, Garcia. S, and Garino. C.G, "An analysis of recurrent neural networks for botnet detection behavior," in *Proc. of 2016 IEEE biennial congress of Argentina (ARGENCON)*, pp. 1-6, 2016. Article (CrossRef Link)

[31] Atiga. J, Mbarki. N.E, Ejbali. R, and Zaied. M, "Faulty node detection in wireless sensor networks using a recurrent neural network," in *Proc. of Tenth international conference on machine vision (ICMV 2017)*, Vol. 10696, pp. 711-716, 2018. Article (CrossRef Link)

[32] Vu. L, Nguyen. Q.U, Nguyen. D.N, Hoang. D.T, and Dutkiewicz. E, "Learning latent representation for iot anomaly detection," *IEEE Transactions on Cybernetics*, 52(5), pp. 3769-3782, 2022. Article (CrossRef Link)

[33] Abdel-Basset. M, Chang. V, Hawash. H, Chakrabortty. R.K, and Ryan. M, "Deep-IFS: Intrusion detection approach for industrial internet of things traffic in fog environment," *IEEE Transactions on Industrial Informatics*, 17(11), pp. 7704-7715, 2020. Article (CrossRef Link)

[34] Imrana. Y, Xiang. Y, Ali. L, and Abdul-Rauf. Z, “A bidirectional LSTM deep learning approach for intrusion detection,” *Expert Systems with Applications*, 185, p. 115524, 2021. Article (CrossRef Link)
[35] Habib. M, Aljarah. I, and Faris. H, “A modified multi-objective particle swarm optimizer-based Lévy flight: An approach toward intrusion detection in Internet of Things,” *Arabian Journal for Science and Engineering*, 45, pp. 6081-6108, 2020. Article (CrossRef Link)
[36] Chen. Y, Lin. Q, Wei. W, Ji. J, Wong. K.C, and Coello. C.A.C, “Intrusion detection using multi-objective evolutionary convolutional neural network for Internet of Things in Fog computing,” *Knowledge-Based Systems*, 244, p. 108505, 2022. Article (CrossRef Link)
[37] Zhu. Y, Liang. J, Chen. J, and Ming. Z, “An improved NSGA-III algorithm for feature selection used in intrusion detection,” *Knowledge-Based Systems*, 116, pp. 74-85, 2017. Article (CrossRef Link)
[38] Y.H. Yang, H.Z. Huang, Q.N. Shen, Z.H. Wu, and Y. Zhang, “Research on intrusion detection based on incremental GHSOM,” *Chinese J. Comput.* 37(5), pp. 1216–1224, 2014 Article (CrossRef Link)
[39] LeCun. Y and Bengio. Y, “Convolutional networks for images, speech, and time series,” *The handbook of brain theory and neural networks*, 3361(10), p. 1995, 1995. Article (CrossRef Link)
[40] Kim. J, Shin. Y, and Choi. E, “An intrusion detection model based on a convolutional neural network,” *Journal of Multimedia Information System*, 6(4), pp. 165-172, 2019. Article (CrossRef Link)
[41] Kursawe. F, “A variant of evolution strategies for vector optimization,” in *Proc. of Parallel Problem Solving from Nature: 1st Workshop*, pp. 193-197, June. 2005. Article (CrossRef Link)
[42] Panigrahi. R and Borah. S, “A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems,” *International Journal of Engineering & Technology*, 7(3.24), pp. 479-482, 2018. Article (CrossRef Link)
[43] Assis. M.V, Carvalho. L.F, Lloret. J, and Proença Jr. M.L, “A GRU deep learning system against attacks in software defined networks,” *Journal of Network and Computer Applications*, 177, p. 102942, 2021. Article (CrossRef Link)
[44] Vinayakumar. R, Soman. K.P, and Poornachandran. P, “A comparative analysis of deep learning approaches for network intrusion detection systems (N-IDSs): deep learning for N-IDSs,” *International Journal of Digital Crime and Forensics (IJDCF)*, 11(3), pp. 65-89, 2019. Article (CrossRef Link)
[45] Gamage. S and Samarabandu. J, “Deep learning methods in network intrusion detection: A survey and an objective comparison,” *Journal of Network and Computer Applications*, 169, p. 102767, 2020. Article (CrossRef Link)
[46] Lv. Z, Qiao. L, Li. J, and Song. H, “Deep-learning-enabled security issues in the internet of things,” *IEEE Internet of Things Journal*, 8(12), pp. 9531-9538, 2020. Article (CrossRef Link)
[47] Muhammad. G, Hossain. M.S, and Garg. S, “Stacked autoencoder-based intrusion detection system to combat financial fraudulent,” *IEEE Internet of Things Journal*, vol. 10, no. 3, pp. 2071–2078, 2023. Article (CrossRef Link)
[48] Altunay. H.C and Albayrak. Z, “A hybrid CNN+ LSTM based intrusion detection system for industrial IoT networks,” *Engineering Science and Technology, an International Journal*, 38, p. 101322, 2023. Article (CrossRef Link)
[49] Altunay. H.C and Albayrak. Z, “Network Intrusion Detection Approach Based on Convolutional Neural Network,” *Avrupa Bilim ve Teknoloji Dergisi*, (26), pp. 22-29, 2021. Article (CrossRef Link)
[50] Kavitha. S, Uma Maheswari. N, and Venkatesh. R, “Intelligent Intrusion Detection System using Enhanced Arithmetic Optimization Algorithm with Deep Learning Model,” *Tehnički vjesnik*, 30(4), pp. 1217-1224, 2023. Article (CrossRef Link)
[51] Khan. M.A, Iqbal. N, Jamil. H, and Kim. D.H, “An optimized ensemble prediction model using AutoML based on soft voting classifier for network intrusion detection,” *Journal of Network and Computer Applications*, 212, p. 103560, 2023. Article (CrossRef Link)

Alexander R received the B.E. degree in Computer Science and Engineering from Anna University,2006 and M.E. degree in Computer Science and Engineering from Anna University, 2011. He is pursuing Ph.D in the Department of Computing Technologies, SRM Institute of Science and Technology, Kattankulathur, Tamil Nadu, India. His research interests include Federated Learning, Fog Computing, Network Security, Deep Learning, Cloud Computing, Internet of things and Machine learning.

Pradeep Mohan Kumar K received the B.E. degree in Computer Science and Engineering from Anna University, 2005, M.E. degree in Computer Science and Engineering from Annamalai University, 2007 and Ph.D degree in Periyar Maniammai Institute of Science and Technology, India, 2016. He is an Associate Professor in the Department of Computing Technologies, SRM institute of Science and Technology Kattankulathur, Tamil Nadu, India. His research interests include AI/ML, IoT, Cloud Computing, Network Security.