International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING

ISSN:2147-6799 www.ijisae.org Original Research Paper

Botnet Detection in the Internet-of-Things Networks Using Densenet - Binary Moth Flame Optimization

Swapna Thota[1], Dr. D. Menaka[2]

Submitted: 19/01/2024 **Revised**: 28/02/2024 **Accepted**: 05/03/2024

Abstract: DDoS attacks based on the Internet of Things (IoT) have increased in number as a result of its recent growth. In this paper, a method for identifying botnet activity in consumer IoT networks and devices is presented. However, highly unbalanced network traffic data in the training set deteriorates the state-of-the-art ML and DL algorithms' classification capabilities, especially in classes with small sample sizes. This study developed a deep learning-based botnet assault detection algorithm called DenseNet - Binary Moth Flame Optimisation (DenseNet-BMFO). In the meantime, the overall performance of the proposed DenseNet-BMFO and other commonly used algorithms is compared using standard evaluation markers. According to the simulation results, the DenseNet-BMFO approach for identifying IoT network intrusion threats is dependable and efficient. The results of the experiments showed that the suggested methodology produced a 98.25% accuracy rate. The results of the experiment show that the suggested model performs better in botnet detection categorization than the existing methods.

Keywords: *Internet of Things, IoT botnets, IoT botnet detections, DenseNet, Binary Moth Flame Optimization*

1. Introduction

In order to take advantage of the many advantages offered by the Internet of Things (IoT), essential infrastructures including electricity generation [1], communications, healthcare, manufacturing, transportation, water treatment, and agriculture [2] are now interconnected. Because of their greater interconnectedness and ICS use, smart critical infrastructures are more susceptible to cyberattacks by terrorists or "hacktivists." For example, it has been demonstrated that ICS and IoT devices are readily hackable and may be remotely controlled to create IoT-based botnets.. A single weak IoT device can be successfully exploited to compromise the security of the entire IoT-enabled system and disclose confidential data. Because of this, Advanced Persistent Threats (APT) of various botnet attacks find them appealing targets, particularly when they are implemented in sensitive situations.

Network Intrusion Detection Systems (NIDS) have been developed using Machine Learning (ML) approaches to protect connected IoT devices against sophisticated botnet attacks, such as [2]. These NIDS can be placed in key locations across an Internet of things network. In particular, Deep Learning (DL), an advanced machine learning technique, has a special aptitude for automatically extracting features from high-volume, high-speed network traffic produced by heterogeneous, networked Internet of things devices [3]. Due to their high processing and memory needs, NIDS approaches employed in IoT devices. Enough network traffic data is required to ensure effective classification performance in the development of a DL approach for Internet of Things networks to detect botnets [4]. Nevertheless, the curse of dimensionality may result from handling and evaluating high-dimensional network traffic data. Hughes phenomena can also arise from training deep learning models. Processing large amounts of data necessitates a great deal of computational power and storage space. IoT devices lack the memory capacity needed to store the large amounts of network traffic data needed for deep learning. Consequently, a complete DL-based botnet detection system is required that can both accurately detect recent and sophisticated botnet attacks that lessen the high dimensionality of huge network traffic features by using low-dimensional network traffic information.

Because it has the following characteristics, the Bot-IoT dataset [5] is now the most relevant publically available dataset for the detection of botnet assaults in IoT networks IoT network traffic samples (a), complete network information acquired (b), a range of complex IoT botnet assault scenarios (c), accurate ground truth labels (d), and a vast amount of labelled data required for effective supervised DL. Initial feature dimensionality1 for the Bot-IoT dataset is 43, and 1.085 GB of memory are required to store this network traffic data. Thus far, feature dimensionality reduction strategies based on feature selection algorithms have only been applied to the Bot-IoT dataset.

[1]Research Scholar, [2]Associate Professor
[1,2]Noorul Islam Centre for Higher Education, Kanyakumari, TamilNadu, India
[1]swapnathota@gmail.com, [2]menaka@niuniv.com

Proposed DenseNet - Binary Moth Flame Optimisation framework:

In this study, we present a DenseNet - Binary Moth Flame Optimisation framework (DenseNet-BMFO) that leverages DenseNet 201 and Binary Moth Flame Optimisation for effective IoT network botnet detection. The main contributions of this study are as follows:

(i) Generating a publicly available, labelled dataset with attack vectors, botnet activity, and regular traffic. DenseNet-BMFO In order to categorise network traffic samples into ten botnet attack classes in addition to the regular class, models are trained, verified, and tested using the Bot-IoT dataset.

(ii) The suggestion for reducing feature dimensionality is Binary Moth Flame Optimisation. This method reduces the dimensionality of large-scale IoT network traffic data while producing a low-dimensional latent-space feature representation at the hidden layer without compromising important intrinsic network information.

(iii) A DenseNet201 is suggested for the categorization of network traffic. This technique scans IoT network traffic to separate benign traffic from botnet attack traffic;

(iv) A number of tests are conducted using the Bot-IoT dataset to confirm that DenseNet-BMFO works well in scenarios involving binary and multi-class classification.

(v) To ensure effective In IoT networks, feature dimensionality reduction and botnet attack detection the effectiveness of cutting-edge optimization techniques was examined and contrasted.

The remaining portions of the paper are separated into the following categories: In Section II, we look at pertinent state-of-the-art methods for classifying network traffic and reducing feature dimensionality. The remaining portions of the paper are separated into the following categories: In Section II, we look at pertinent state-of-the-art methods for classifying network traffic and reducing feature dimensionality. Section III provides an explanation of the suggested botnet detection technique for Internet of Things networks, known as DenseNet - Binary Moth Flame Optimisation (DenseNet-BMFO). In Section IV, numerous tests are conducted to confirm DenseNet-BMFO's efficacy. Section V presents and discusses the findings of the experiments. Finally, Section VI concludes the work.

2. Related Work

While there are several datasets for network intrusion detection, they suffer from a number of issues, such as missing ground truth, limited assault [7]. The limited amount of benign traffic samples in the DEFCON-8 dataset limits its application [6]. The labels were not supplied, and the network traffic data are provided in packet form without any extracted features [8]. Only denial-of-service attacks are included in the attack scenarios that are available in the UNIBS dataset. The CAIDA databases contain no ground truth information about the attack samples. The LBNL dataset's network traffic samples lacked labelling, and the features included in the packet files could not be extracted [8]. UNSW-NB15 dataset's attack samples were produced in a simulated setting [9]. Furthermore, these datasets' ground truth is not provided to improve the labelling procedure. The botnet scenarios utilised in the majority of datasets are not described in great detail. Furthermore, IoT network traffic data was absent from relevant databases [9].The Bot-IoT dataset is the most relevant publicly available dataset for network-based botnet attack detection in IoT networks. [10].

Numerous Internet of Things (IoT) gadgets were included in the testbed setup, such as a motion-activated light system, smart fridge, weather station, remote-operated garage door, and smart thermostat [11]. The four IoT botnet scenarios that these attack traffic samples fall under are information theft, reconnaissance, DoS, and DDoS. The main method for reducing the dimensionality of features is to apply a method for transforming a high-dimensional feature set—linear or non-linear. One popular linear transformation technique is principal component analysis (PCA) [12, 13], DL approaches, as well as spectral methods [14]. At the hidden layer, an autoencoder, an unsupervised deep learning method, creates a latent-space representation of the input data [15]. In order to decrease the feature dimensionality in the majority of well-known network intrusion datasets, various autoencoder designs have been suggested [16].

This research does not include feature dimensionality reduction approaches that do not cover all The Bot-IoT dataset contains four botnet assault scenarios nor do they include benign network traffic traces, in order to ensure a fair comparison. For example, the DoS attack scenario was not taken into account by Soe et al. [17, 18]. Furthermore, the method's efficacy in identifying innocuous network data was not disclosed [19]. The performance of the suggested strategy was not assessed by the authors of a related paper [20, 21].

To sum up, the innovative methods used in the linked work focused on selecting specific features from the network traffic data of the Bot-IoT dataset. However, this approach might affect how efficiently botnet assaults are identified in IoT networks because the classifiers won't have access to specific relevant network data during training, validation, and testing. As a result, the feature selection strategy in IoT networks may lead to a high false alarm rate and low accuracy in botnet attack detection. But while dimensionality of large-scale IoT network traffic data is reduced and a low-dimensional latent space feature representation is created at the hidden layer, BMF maintains significant intrinsic network information.

3. Proposed Method for Botnet Attack Detection in Iot Networks

This section outlines the research methods that use deep learning and optimisation models to anticipate the existence of IoT botnets. This section includes details about BMFO, DenseNet201, preprocessing of data as well as data on network traffic. Three algorithms from DenseNet - Binary Moth Flame Optimisation (DenseNet-BMFO) were used to identify a botnet that was utilising the Internet of Things. The next chapters provide an explanation of these models. The model's general structure is depicted in Figure.1

3.1 Dataset

For an 11-class classification, the Bot-IoT dataset includes three label categories and 43 network traffic metrics. Out of the 43 features, only 37 were determined to be useful in identifying botnet assaults in Internet of Things networks. A scenario involving the classification of 11 classes is analysed in detail to identify botnet assaults. Because they had less samples than the majority classes in this study, the classes DDU, DDT, DT, SS, OSF, and DUDH, DDH, DE, DH Norm, and KE are regarded as minority.

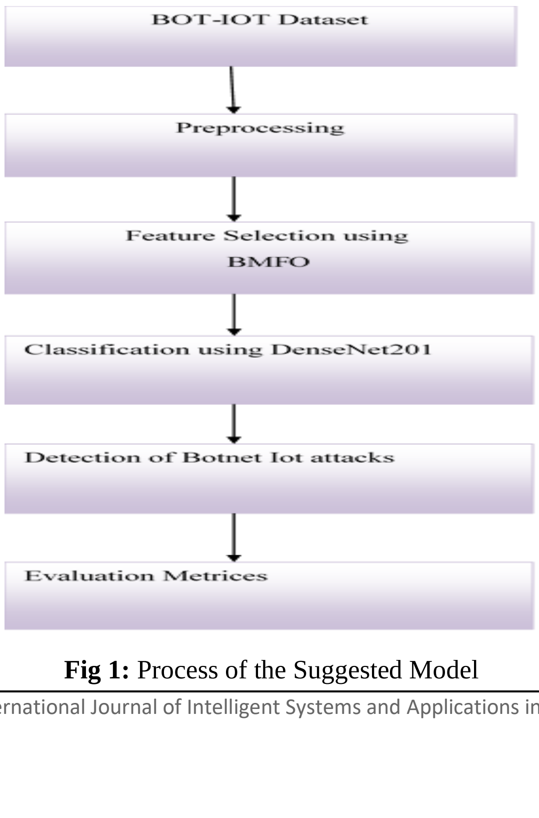

BOT-IOT Dataset
Preprocessing
Feature Selection using BMFO
Classification using DenseNet201
Detection of Botnet Iot attacks
Evaluation Metrices

Fig 1: Process of the Suggested Model

3.2 Data Preprocessing

Examples of data preprocessing include feature normalisation, data splitting, label encoding, and reshaping. All of the traffic highlights in the network estimations were first normalised to a size of 0 and 1, using the min-max normalisation technique described in the Condition below.

$$x_{norm=}\frac{x - x_{min}}{x_{max} - x_{min}}$$

The minimum and minimum values of x are addressed by x_min and x_max, respectively, if x is a feature vector of network traffic. One more component in the feature arrangement deals with a unit time step. This changes the aspect of the list of capabilities from X R(Pq) to X R(Pq1), where the total number of tests (p) and characteristics (q) are represented, correspondingly. The numerals 0 through 10 were used to address the numerical values of the 11 classes. To train and evaluate, the whole dataset was randomly divided into test sets (20%), training sets (60%) and validation sets (20%).

3.3 Optimal Feature Selection

The suggested BMFO method is applied on the feature attributes of the primary datasets. The total sum of features is given by the notation Fk m, where n is a positive integer between 1 and M. Optimal feature selection is indicated by the notation Fk (m*) opt, where k=1,2,...,M and M* is the total sum of optimum feature selections

3.4 Binary Moth-Flame Optimization (BMFO)

To solve the part decision problem, the nonstop MFO method is converted to binary using the S, V, and U-formed move capabilities. The various trading ability classes are fully illustrated in Fragment 4.1, along with flowcharts and pseudo-code that explain how to use them to determine various B-MFO kinds.

3.5 BMFO Using S-Shaped Transfer Function

Common function S2 is the sigmoid (S-shaped) capability shown in Condition (1). It was originally designed for the (B-MFO) design.

$$K_{fs}\big(j_l(f+1)\big) = 1/(1+exp^{j_l}) \qquad (1)$$

$$y_i\,(f+1) = 1/(1+exp^{j_l}) \qquad (2)$$

The current and previous positions are used to resolve each search specialist's updated positions in accordance with condition (2). The speed is used in move capabilities in a few parallel metaheuristic algorithms, such as BMOF, to calculate the likelihood that the region will change.

$$Bf_y\left(y_{ij}(t+1)\right) = |tanh\big(y_i(t)\big)| \qquad (3)$$

$$y_i\,(f+1) = y_i(t) If\; gyf_w\big(y_{ij}(t+1)\big) \qquad (4)$$

$$y_i\ (f+1) = y_i(t) If\ g{>}yf_w\big(y_{ij}(t+1)\big \qquad (5)$$

In cycle t, the position of the ith search specialist in dimension d is assigned by yi (f+1), and the supplement of yij (t+1) is displayed by yij (t+1).Furthermore, R represents an opportunity number that falls between 0 and 1. In the event that the speed is high, the pursuit specialists alter to supplement. However, the TFv instructs the hunt experts to stay in their momentum regions if the speed is low.

3.4 U-Shaped Transfer Function with BMFO

The two control parameters of this function individually govern the basin's width and inclination.

$$MF_{U=}\big(y_{ij}(t+1)\big) = \alpha\big|\big(y_{ij}(t)\big)\alpha = 1 \qquad (6)$$

$$y_{ij}(t+1) = \{y_i(t)\ if\ g{<}yf_w\ \big(\big(y_{ij}(t+1\big)\ y_i(t)\ if\ g{>}yf_w\big(y_{ij}(t+1\big)$$

(7)

3.5 BMFO for Feature Selection Problem Solving

Finding the best subset of the significant and important highlights is the aim of the feature selection issue, which aims to produce an extra accurate information model. Using an exchange capability is highlighted in the element determination problem.

The fitness function that is used to express the objectives displays the classification error as CE. The importance of feature reduction and the quality of the classification are seen in (8).

$$Fitness = \eta. FE\alpha +^{\frac{M_{Sf}}{N_{tf}}} \qquad (8)$$

3.4 Classification Process

At the end of the network, in a layer with as many neurons as the total number of output classes, they are implemented as a classifier. The SoftMax function is used to convert the probability that an input belongs to a specific class into decimal values in multi-class classification problems. The most likely class is determined by looking at the target projected group of the inputs. The SoftMax activation function for a K-class classification issue has the following mathematical expression: (9).

$$\sigma(Z)j = \frac{e^{zj}}{\Sigma_j^l \ \ =1 e^{zj}} \qquad (9)$$

(Z) j is the probability that the input is a member of j, zj is the output along the jth dimension, and j is a class sum from 1 to l.

3.4.1 Transfer Learning Method

Even with cutting-edge technology, these models still need to be taught. One way to do this is through transfer learning, in which a newly constructed and trained model serves as the basis for a new assignment. There are two common approaches you might follow when it comes to transfer learning: Make a method template. Retraining a model is the second strategy. The first stage in building a model approach is to choose a task that is comparable to the current work and includes a large amount of data. Selecting a pre-trained model is the first step in the pre-trained ideal approach.

Transfer learning is implemented by discarding the completely associated layers at the top of the model that were in charge of classifying the dataset and using the previous levels as feature extractors after the replica's pre-trained weights have been determined. The model can be trained to comprehend the functions from the feature extractor to the output classes utilising the fresh dataset by incorporating classification layers. Pre-trained CNNs using DenseNet201 were employed in this investigation.

3.3.1 DenseNet201

Every layer is connected to every other layer by a feed-forward mechanism established by the Dense Convolutional Network (DenseNet) architecture. DenseNet has a very thin layer 12 filters per layer, and the overall collective information of the network is composed of only a small number of feature maps. DenseNet has the advantage of having fewer parameters because of its adaptability, feature deployment, encouragement of feature reuse, and lightness on the gradient problem. DenseNet-201 is the name of a convolutional neural network that has 201 layers. Using the ImageNet database, it loads a pre-trained network that has been trained on over a million images. The network will classify images that represent 1000 different object categories, including pens, mice, keyboards, and various animals. Consequently, comprehensive feature representations for various image kinds are now included in the network. 224 × 224 resolution photos are needed for the network. As shown in Figure 2, each layer consists of batch normalisation, ReLU activation combined with a 3x3 filter convolution.

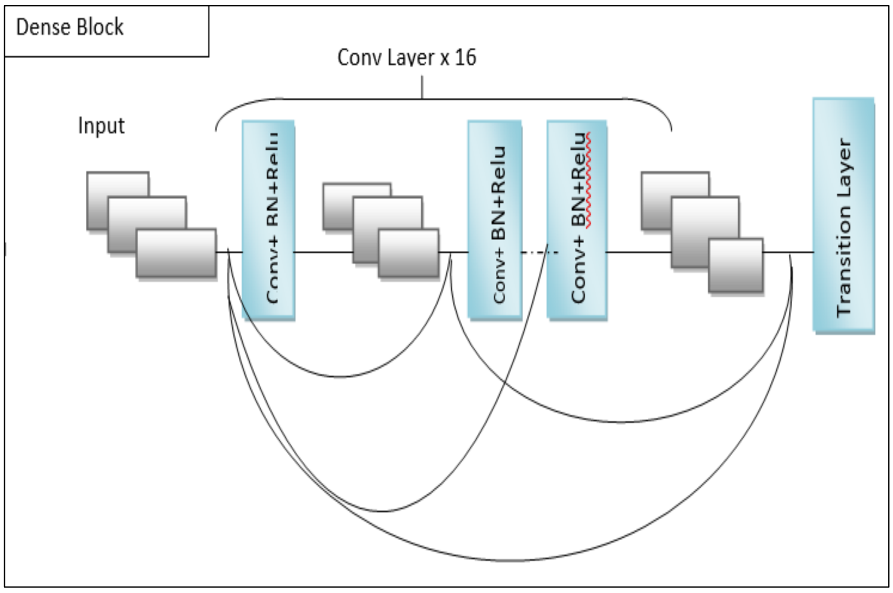


Fig 2. Architecture of DenseNet201

Each block includes a matrix-based input representing a single image pixel to reduce overfitting during training. This matrix has advanced to the batch normalisation stage. When ReLu activation occurs, if the x value is negative, it becomes 0; otherwise, nothing occurs. A matrix image undergoes convolutional processing with a 3x3 filter after a successful ReLu activation step. A matrix value that has already been processed will be the output.

Algorithm: Botnet Identification System

```
1: data Processing
2: ← 25% unitToDrop
3: Parse the data into a preset format.
4: definition of token
5: once again
6: /*Data parsing to format*/
7: Rows do for row ← 1.
8: Text to tokenized integer format conversion
9: Tokenized text index
10: Make a tokenized text index dictionary.
11: Up to 25 max, pad data arrays with 0s
12: Add further tokenized features to the array.
13: conclude for
14: until the dataset is returned
15: Divide training and testing according to unitToDrop
16: Train And Validate with test and training data
17: model ← consecutive ()
18: cell 0
19: engagement → sigmoid
20: defeat ← mae
21: Adam, optimizer
```

```
22: periods → 100
23: Make a new unit for BLSTM/LSTM.
24: Model with an LSTM unit added
25: Make a fresh Dense Layer
26: To the model, add a dense layer.
27: Dense Layer activation should be set.
28: Build the model with Loss and Optimiser
29: Once again
30: *Adjust Model*/
31: epochs do for epoch ← 1.
32: Analyse Loss, Verify Loss
33: Analyse both the validation and overall accuracy.
34: conclude for
35: till every epoch has ended
36: ValLoss, Return Loss, Acc, and ValAcc
```

4. Results and Discussion

Where the union of maps from layers 0 to l-1 is represented by the symbol [x 0, x 1... x (l-1)], and x l is the layer's input. By dividing the network's nodes into dense blocks (DB) and transition nodes (TB), the authors were able to modularize their system. Multiple dense layers (DL) made up of 11 Conv and 33 Conv layers make up a dense block. The 11 Conv layer, the 22 average pooling layer, and the batch layer make up the transition blocks that lie in between the dense blocks. DenseNet121 is a variant of the network consisting of four dense blocks, each containing six, twelve, twenty-four, or sixteen dense layers.

4.1. Performance metrices

We have utilised the three commonly-used ROC measures to compare the effectiveness of various categorization methods. A learning table that can be used to depict how a classification model is presented is called a confusion matrix. The "True Positive" (T P) test results are the first to accurately determine whether a botnet is under attack; the "True Negative" (T N) test results are the second to accurately determine whether a botnet is not under attack; the "False Negative" (F N) test results are the third to fourth; and the "False Positive" (FP) test results are the fifth to last. In terms of forecasts. Accuracy for a dataset of scope n is defined as. The confusion matrix values display the proportion of properly and erroneously classified samples. The True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) value categories make up the confusion matrix. The identified lesions are subjected to performance standards for sensitivity, specificity, and accuracy. True positive (TP): The test would be positive if there was an input of a tumour.

True negative (TN): The test would be negative if the input wasn't tumor-related.

False positive (FP): When an input is not a tumour,The outcome is favourable.

False negative (FN): The outcome would be negative if there was a tumour input.

Sensitivity or Recall:

The following criteria are used to assess the ability to correctly identify MIR lesions:

$$\text{Sensitivity} = \frac{TP}{TP+FN} \qquad (10)$$

Specificity:

Specificity, expressed as, can be used to quantify the degree of accuracy with which MIR lesions can be rejected.

$$\text{Specificity} = \frac{TN}{TP+FP} \qquad (11)$$

Accuracy:

The lesion detection performance is calculated using the formula

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \qquad (12)$$

Precision:

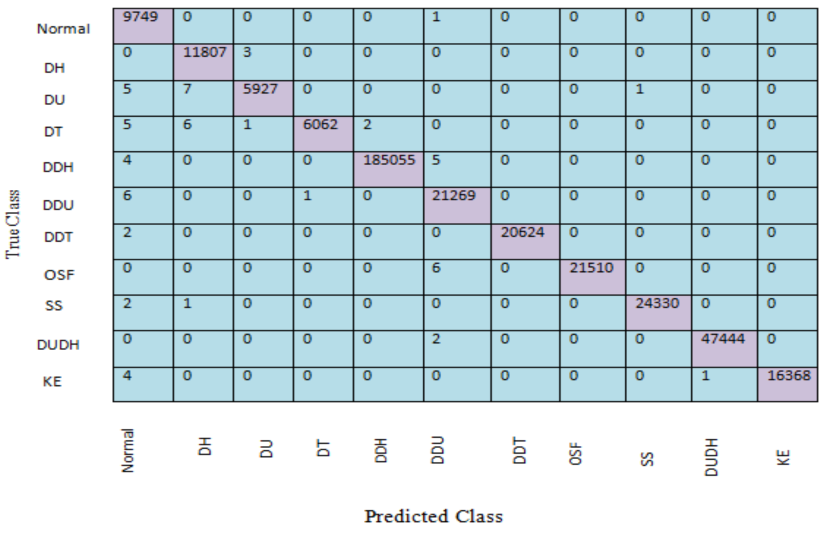
Precision gauges how well a machine learning model can predict positive cases. The association between the overall number of true positive estimations and the false positive estimates that correspond to true positives is demonstrated.

$$Precision = \frac{TP}{(TP+FP)} \qquad (13)$$

F1-Score:

To determine the F1-score, the precision and recall harmonic techniques are applied. The following formula is used to calculate the F1-Score:

$$F1 - Measure = (2 \times P \times R)/(P + R) \qquad (14)$$


True Class
Predicted Class

	Normal	DH	DU	DT	DDH	DDU	DDT	OSF	SS	DUDH	KE
Normal	9749	0	0	0	0	1	0	0	0	0	0
DH	0	11807	3	0	0	0	0	0	0	0	0
DU	5	7	5927	0	0	0	0	0	1	0	0
DT	5	6	1	6062	2	0	0	0	0	0	0
DDH	4	0	0	0	185055	5	0	0	0	0	0
DDU	6	0	0	1	0	21269	0	0	0	0	0
DDT	2	0	0	0	0	0	20624	0	0	0	0
OSF	0	0	0	0	0	6	0	21510	0	0	0
SS	2	1	0	0	0	0	0	0	24330	0	0
DUDH	0	0	0	0	0	2	0	0	0	47444	0
KE	4	0	0	0	0	0	0	0	0	1	16368

Fig 3: Confusion Matrix

Figure 3 illustrates how the primary diagonal elements show examples of effectively classified data. The number of distinct classes in the dataset in a multi-class classification issue is equal to the number of rows and columns in the confusion matrix. For every class, the TP, TN, FP, and FN observations are generated using the confusion matrix. In light of the observations, a number of performance indicators, including accuracy, precision, recall, and F1 score, are assessed. On the first dataset, Table 1 displays the estimated model's performance using the various approaches.

Table 1: Analysis of Suggested Model on Initial Dataset

Methods	**Accuracy (%)**	**Precision (%)**	**Recall**	**F1**
SVM	82.3	83.4	0.8528	0.8234
Efficient Net	84.5	84.2	0.8624	0.8323
ResNet	88.9	88.1	0.8859	0.8614
VGGNet	90.4	87.7	0.8921	0.8742
AlexNet	92.3	89.6	0.8947	0.8852
MobileNet	93.2	90.2	0.8992	0.8941
DenseNet-BMFO	**98.2**	**95.8**	**0.9511**	**0.9526**

The suggested model scored 98.2% in the accuracy analysis, while the current models, including SVM, EfficientNet, and ResNet, scored between 82% and 88%, and VGGNet, AlexNet, and MobileNet, scored between 90% and 93%. The existing methods, including SVM, EfficientNet, ResNet, VGGNet, AlexNet, and MobileNet, achieved roughly 82% to 92% when the models were assessed for precision, recall, and F1-score; in contrast, the suggested model achieved 95.8% accuracy, 95.11% recall, and 95.26% F1-score. Better performance can be attributed to the BMFO model's selection of the best features. Table 2 shows the projected model's performance using the available methods on the second dataset. The projected model's graphical analysis is shown in Figures 4 and 5.

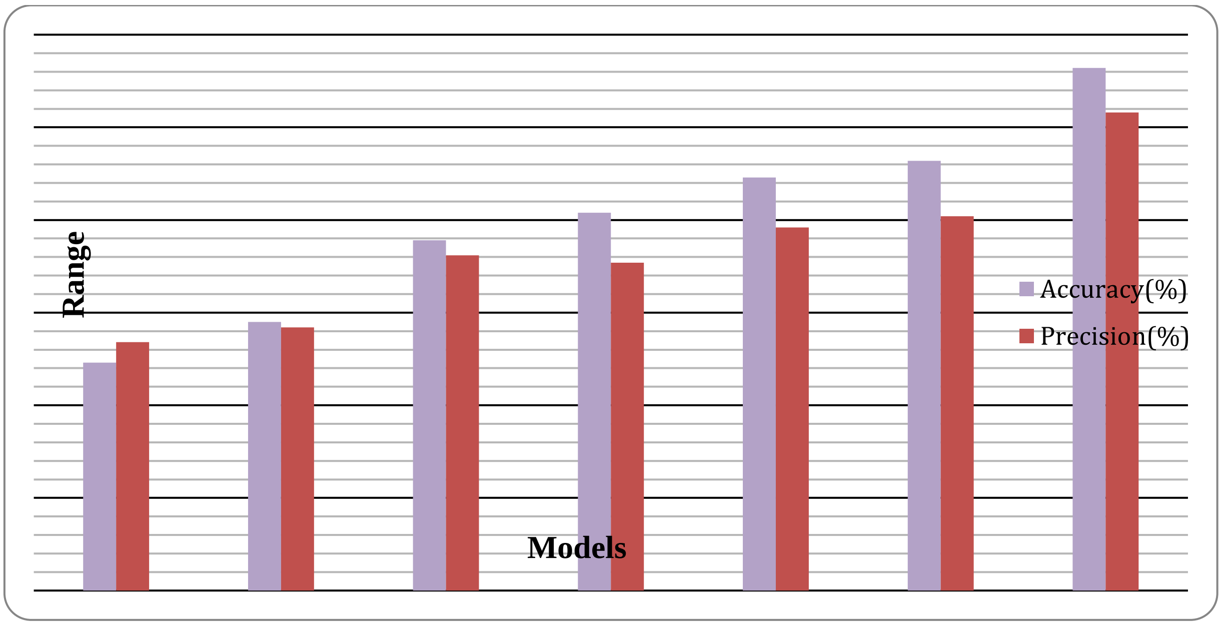


Fig 4: Comparative analysis of the suggested model

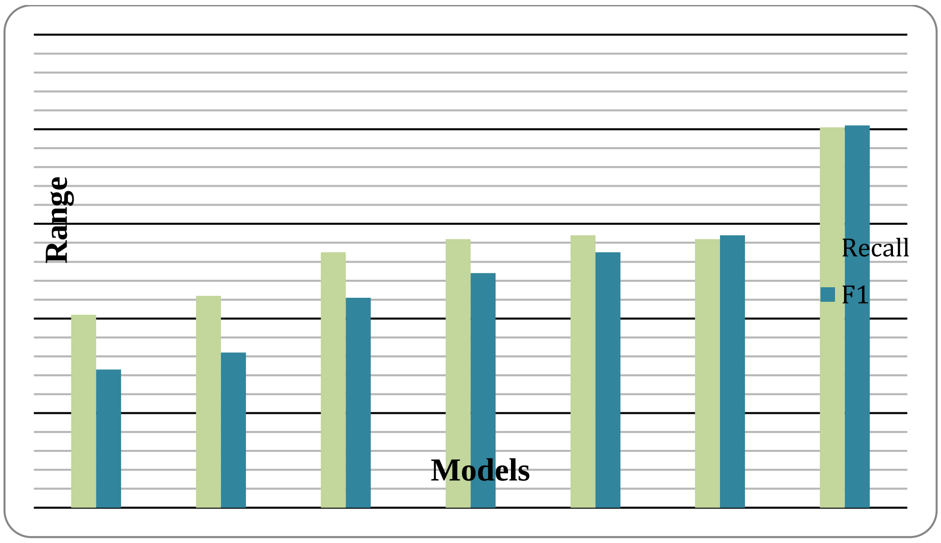


Fig 5: Analysis of Projected Model

Table 2: Analysis of the Proposed Model for the Second Dataset

Methods	Accuracy (%)	Precision (%)	Recall	F1
SVM	82.2	82.5	79.6	80.7
Efficient Net	84.6	84.2	80.7	81.3
ResNet	85.7	86.8	82.0	85.6
VGGNet	88.1	88.7	82.5	87.4
AlexNet	89.6	88.9	85.4	88.3
MobileNet	90.6	89.6	88.4	88.9
DenseNet-BMFO	96.5	95.4	93.2	93.1

SVM scored 80%, EfficientNet scored 81%, ResNet scored 85%, VGGNet scored 87%, AlexNet scored 88%, MobileNet scored 88%, and the suggested model scored 93% in the examination of the F1-score. The suggested model attained 96.5% accuracy, 95.4% precision, and 93.2% recall compared to the other current models' about 82% to 90% accuracy, precision, and recall. We present the graphical analysis in Figures 6 and 7

.

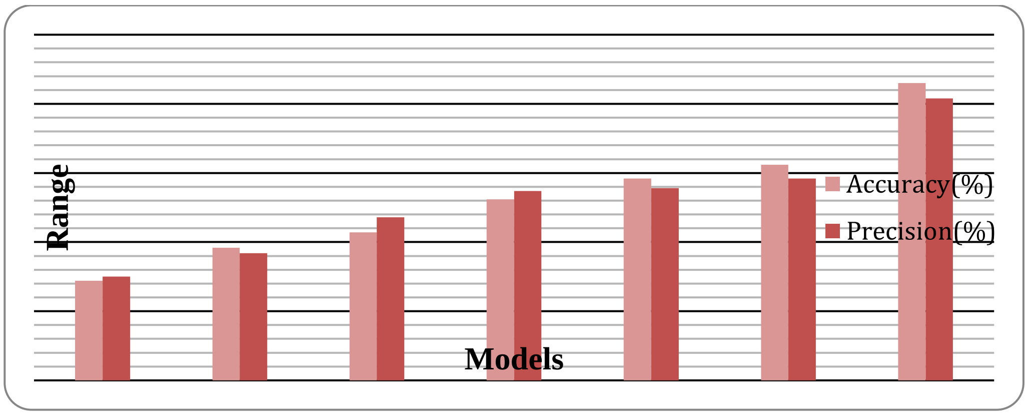


Fig 6: Analysis the suggested model on the second dataset

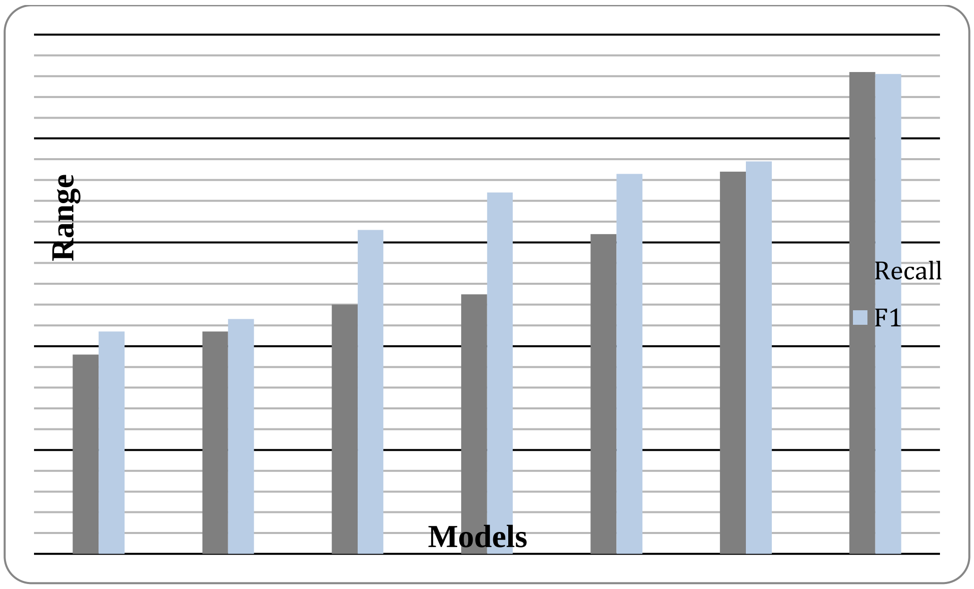


Fig 7: The Effectiveness of Different Classifiers

5. Conclusion

A variety of commercial sectors are using IoTs to enhance their services. These apps address many different subjects, such as education and health. The gadgets make cybersecurity attacks easier and faster, especially when botnets are used. Fortunately, botnets go through a number of steps before to initiating attacks, which might be utilised to identify these attacks in advance. This study proposes the DenseNet - Binary Moth Flame Optimisation (DenseNet-BMFO) model, which uses the BOT-IOT datasets to detect DDoS attacks on botnets. DenseNet-BMFO is outperforming other algorithms in terms of precision, accuracy, recall, and F1 score, according to a variety of performance criteria, and this is crucial for computer security and related domains. These findings demonstrated the DenseNet - Binary Moth Flame Optimisation (DenseNet-BMFO) model's resistance to both underfitting and overfitting. Furthermore, DenseNet-BMFO showed improved generalisation ability in minority classes. With an accuracy of 98.2%, experiments showed that the proposed DenseNet-BMFO technique worked better than other well-known metaheuristic algorithms. The features, efficacy, and constraints of the techniques have also been examined. The class's strong potential for identifying botnet assaults in Internet of Things applications is demonstrated by the analysis of several detection techniques. However, the research also identifies other areas in which optimization-based and deep learning techniques still require refinement. The focus of recent botnets has switched to intelligent gadgets like air conditioners and televisions. Therefore, creating effective IoT solutions to defend smart appliances from botmasters will be a future problem.

References

[1] Arshad, Amina, Maira Jabeen, Saqib Ubaid, Ali Raza, Laith Abualigah, Khaled Aldiabat, and Heming Jia. "A novel ensemble method for enhancing Internet of Things device security against botnet attacks." *Decision Analytics Journal* 8 (2023): 100307.

[2] Nasir, Muhammad Hassan, Junaid Arshad, and Muhammad Mubashir Khan. "Collaborative device-level botnet detection for internet of things." *Computers & Security* 129 (2023): 103172.

[3] Thota, Swapna, and D. Menaka. "Botnet detection in the internet-of-things networks using convolutional neural network with pelican optimization algorithm." *Automatika* 65, no. 1 (2024): 250-260.

[4] Sudhakar, and Sushil Kumar. "ABBDIoT: Anomaly-Based Botnet Detection Using Machine Learning Model in the Internet of Things Network." In *International Conference on IoT, Intelligent Computing and Security: Select Proceedings of IICS 2021*, pp. 235-245. Singapore: Springer Nature Singapore, 2023.

[5] KOLCU, Yağız Onur, Ahmet Haşim YURTTAKAL, and Berker BAYDAN. "INTERNET OF THINGS BOTNET DETECTION VIA ENSEMBLE DEEP NEURAL NETWORKS." *International Journal of 3D Printing Technologies and Digital Industry* 7, no. 2 (2023): 191-197.

[6] Gharehchopogh, Farhad Soleimanian, Benyamin Abdollahzadeh, Saeid Barshandeh, and Bahman Arasteh. "A multi-objective mutation-based dynamic Harris Hawks optimization for botnet detection in IoT." *Internet of Things* 24 (2023): 100952.

[7] Al-Fawa'reh, Mohammad, Jumana Abu-Khalaf, Patryk Szewczyk, and James Jin Kang. "MalBoT-DRL: Malware Botnet Detection Using Deep Reinforcement Learning in IoT Networks." *IEEE Internet of Things Journal* (2023).

[8] Nazir, Ahsan, Jingsha He, Nafei Zhu, Ahsan Wajahat, Xiangjun Ma, Faheem Ullah, Sirajuddin Qureshi, and Muhammad Salman Pathan. "Advancing IoT security: A systematic review of machine learning approaches for the detection of IoT botnets." *Journal of King Saud University-Computer and Information Sciences* (2023): 101820.

[9] Taher, Fatma, Mahmoud Abdel-salam, Mohamed Elhoseny, and Ibrahim M. El-hasnony. "Reliable Machine Learning Model for IIoT Botnet Detection." *IEEE Access* (2023).

[10] Nguyen, Dat-Thinh, and Kim-Hung Le. "The robust scheme for intrusion detection system in internet of things." *Internet of Things* 24 (2023): 100999.

[11] Bojarajulu, Balaganesh, Sarvesh Tanwar, and Thipendra Pal Singh. "Intelligent IoT-BOTNET attack detection model with optimized hybrid classification model." *Computers & Security* 126 (2023): 103064.

[12] Alabsi, Basim Ahmad, Mohammed Anbar, and Shaza Dawood Ahmed Rihan. "CNN-CNN: Dual Convolutional Neural Network Approach for Feature Selection and Attack Detection on Internet of Things Networks." *Sensors* 23, no. 14 (2023): 6507.

[13] de Caldas Filho, Francisco Lopes, Samuel Carlos Meneses Soares, Elder Oroski, Robson de Oliveira Albuquerque, Rafael Zerbini Alves da Mata, Fábio Lúcio Lopes de Mendonça, and Rafael Timóteo de Sousa Júnior. "Botnet Detection and Mitigation Model for IoT Networks Using Federated Learning." *Sensors* 23, no. 14 (2023): 6305.

[14] Kumar, Atul, and Ishu Sharma. "Augmenting iot healthcare security and reliability with early detection of iot botnet attacks." In *2023 4th International Conference for Emerging Technology (INCET)*, pp. 1-6. IEEE, 2023.

[15] Essa, Mohamed Saied, and Shawkat Kamal Guirguis. "Evaluation of Tree-Based Machine Learning Algorithms for Network Intrusion Detection in the Internet of Things." *IT Professional* 25, no. 5 (2023): 45-56.

[16] Yang, Changjin, Weili Guan, and Zhijie Fang. "IoT Botnet Attack Detection Model Based on DBO-Catboost." *Applied Sciences* 13, no. 12 (2023): 7169.

[17] DK Priya, MP Ramkumar, D Menaka, "Rail Fastener Fault Detection Based on Enhanced YOLOv7 Model", 2023 International Conference on Circuit Power and Computing Technologies, 2023.

[18] Alani, Mohammed M., Ali Ismail Awad, and Ezedin Barka. "ARP-PROBE: An ARP spoofing detector for Internet of Things networks using explainable deep learning." *Internet of Things* 23 (2023): 100861.

[19] Saied, Mohamed, Shawkat Guirguis, and Magda Madbouly. "Review of artificial intelligence for enhancing intrusion detection in the internet of things." *Engineering Applications of Artificial Intelligence* 127 (2024): 107231.

[20] Alkhudaydi, Omar Azib, Moez Krichen, and Ans D. Alghamdi. "A deep learning methodology for predicting cybersecurity attacks on the internet of things." *Information* 14, no. 10 (2023): 550.

[21] Elsayed, Nelly, Zag ElSayed, and Magdy Bayoumi. "IoT Botnet Detection Using an Economic Deep Learning Model." *arXiv preprint arXiv:2302.02013* (2023).