DenseRSE-ASPPNet: An Enhanced DenseNet169 with Residual Dense Blocks and CE-HSOA-Based Optimization for IoT Botnet Detection

Mohd Abdul Rahim Khan
Department of Electrical Engineering and Computer Science, A’sharqiyah University, IBRA-400 OMAN

***Abstract*—The growing prevalence of Internet of Things (IoT) devices has heightened vulnerabilities to botnet-based cyberattacks, necessitating robust detection mechanisms. This paper proposes DenseRSE-ASPPNet, an advanced deep learning framework for botnet detection, incorporating comprehensive preprocessing, feature extraction, and optimization. The preprocessing pipeline includes data cleaning and Min-Max normalization to ensure high-quality input data. The DenseNet169 backbone is enhanced with Residual Squeeze-and-Excitation (RSE) blocks for channel-wise attention recalibration and Atrous Spatial Pyramid Pooling (ASPP) for capturing multi-scale spatial patterns, enabling effective feature extraction. Hyperparameter optimization is performed using the Cyclone-Enhanced Humboldt Squid Optimization Algorithm (CE-HSOA), which balances global exploration and local exploitation, ensuring faster convergence and enhanced robustness. Experimental results demonstrate the superior performance of the proposed framework, achieving 99.00 per cent accuracy, 96.40 per cent sensitivity, and 99.95 per cent specificity, significantly minimizing false positives and false negatives. The proposed DenseRSE-ASPPNet provides an efficient, scalable, and effective solution for mitigating botnet threats in IoT environments.**

Keywords*—Internet of Things; botnet detection; DenseRSE-ASPPNet; residual squeeze-and-excitation blocks; Cyclone-Enhanced Humboldt Squid Optimization Algorithm

I. INTRODUCTION

The connectivity of billions of intelligent objects with internet-based communication capabilities is known as the "Internet of Things." The number of commonplace machines that have sensors built in and are able to interact online has significantly increased in recent years. By fusing digital intelligence with physical equipment, the Internet of Things makes the world wiser. There is a lot of data exchange between the connected devices, and security is the main issue with IoT [1], [2], [3]. IoT devices are vulnerable to several types of cyberattacks since they connect objects to the internet and allow them to communicate with one another without human intervention. An ever-growing pool of attack resources is made possible by the quick spread of unsecured IoT devices and the simplicity with which attackers can find them via web services like Shodan. Attackers can now launch extensive attacks, including phishing, spam, and Distributed Denial of Service (DDoS), against Internet resources by assembling and utilizing many of these susceptible IoT devices [4], [5], [6]. At the very beginning of IoT device design and deployment, appropriate security requirements should be determined in order to guarantee the security of the IOT network and devices.

Since the Internet of Things is still in its infancy, it does not yet have a strong security framework or system, which puts sensitive data at risk. To keep IoT entities, businesses, and individuals safe, modern security techniques must be implemented on IoT networks. Botnet-based DDoS attacks, in which hackers infect devices with scripts, pose the biggest security threat to the Internet of Things [7], [8]. Botnet detection is a significant difficulty in the cybersecurity field due to the variety of botnet structures and protocols and the constant development of new, clever methods by attackers to damage networks through botnet-assisted attacks [9], [10]. An intrusion detection system (IDS) is more successful at defending a computer network from external threats, even if many solutions, like firewalls and encryption, are designed to tackle Internet-based cyberattacks. Therefore, identifying and stopping different kinds of harmful network communications and computer device usage is the main objective of an intrusion detection system (IDS) [11], [12], [13]. IDS, monitor and analyses a network's regular everyday activity to detect and identify hostile cyberattacks. Enhancing a system's security requires an intrusion detection system (IDS) that can detect botnets in the network and different botnet-assisted attacks.

The complexity and evolution of botnets have led to the proposal of numerous botnet detection techniques. The use of machine learning (ML) techniques for botnet identification has become increasingly popular within the past ten years. Before ML models are learned or trained, feature extraction is a crucial step. When learning and drawing conclusions, these characteristics act as discriminators. Although some of the current methods for detecting botnets rely on packet information or traffic features, they are rendered ineffective when traffic patterns are encrypted or secret, and traffic patterns can be purposefully changed to evade detection [14], [15]. Further, the inability of flow-based machine learning algorithms to identify botnets to capture the dynamic topological structure of communication networks is one of their main shortcomings.

The proposed approach presents an improved DenseNet169-based deep learning framework enriched with Squeeze-and-Excitation (SE) blocks and Atrous Spatial Pyramid Pooling (ASPP) to address the shortcomings of current botnet detection techniques. This design tackles issues including restricted spatial pattern identification in network traffic, shallow gradient propagation, and ineffective feature extraction. Whereas, ASPP captures multi-scale spatial data without adding computing overhead, the addition of SE blocks enhances channel-wise attention. Advanced pre-processing methods further guarantee high-quality input data, and the Self-Adaptive Humboldt Squid Optimization Algorithm (HSOA) optimizes the model's performance by fine-tuning the hyperparameters. For the detection of multi-class botnet attacks in IoT systems, this all-encompassing method improves detection accuracy and robustness, making it extremely effective. The following are the paper's main contributions:

Development of an advanced DenseNet169-based deep learning model, DenseSE-ASPPNet, integrating Residual Squeeze-and-Excitation (RSE) blocks for channel-wise attention and Atrous Spatial Pyramid Pooling (ASPP) for multi-scale feature extraction.

Incorporation of the Cyclone-Enhanced Humboldt Squid Optimization Algorithm (CE-HSOA) for efficient hyperparameter tuning, achieving a balance between global exploration and local exploitation.

The Residual Squeeze-and-Excitation (RSE) block is an enhancement to the standard Squeeze-and-Excitation (SE) block, incorporating a residual learning approach to improve feature recalibration, which helps with better channel-wise attention and more robust feature extraction.

The paper is structured as follows: Section II presents a comprehensive literature review on existing botnet detection methods. Section III details the DenseSE-ASPPNet framework. Section IV compares the performance of DenseSE-ASPPNet with other methods. Finally, Section V provides the conclusion.

II. Literature Review

This section discusses the recent existing papers related to the Botnet attack detection.

In 2022, Nookala Venu, et al., [16] employing machine learning to detect botnet assaults in the Internet of Things. The increasing number of IoT devices that are susceptible to botnet assaults has made them a serious threat to internet security. Many machine learning (ML)-based methods have been released so far to identify different types of botnet attacks. Regardless of the dataset, this study proposes a universal feature set that is extrapolated based on the frequency counting approach and the Logistic Regression method to better detect botnet attacks. There are six main steps in the process overall, starting with data collection and ending with the detection of botnet attacks.

In 2022, Alissa, et al., [17] Detecting botnet attacks in IoT with machine learning. UNSW-NB15, the most comprehensive dataset that is publicly accessible, was used in that study. Exploratory Data Analysis (EDA) is the statistical analysis stage that examines the entire dataset. In the future, the model will be able to be trained on a big dataset. SVM and Random Forest are two examples of machine learning classifiers that can be tested. Runtime Botnet detection can also be done with deep learning models in addition to ResNet50 and LSTM models.

In 2023, Al-Fawa'reh, et al., [18] Detecting malware botnets in IoT networks with deep reinforcement learning. MalBoT-DRL, a powerful malware botnet detector that uses deep reinforcement learning (RL), is presented in this paper. Enhanced generalizability and robustness against model drift are features of MalBoT-DRL, which is designed to detect botnets at every stage of their lifespan. Damped incremental statistics and an attention reward mechanism are combined in this model, which hasn't been thoroughly studied in the literature. The dynamic adaptation of MalBoT-DRL to the constantly evolving malware patterns in IoT environments is made possible by this integration.

In 2022, Kalakoti, et al., [19] Robust feature selection for automated botnet detection in Internet of Things networks using statistical machine learning. In this research, we minimize feature sets for machine learning tasks, which are structured as six distinct binary and multiclass classification problems according to the stages of the botnet life cycle. More precisely, for every classification task, we determined the best feature sets by combining filter and wrapper techniques with particular machine learning techniques. The SFS and SBS wrapper approaches worked well for identifying the best feature sets for each classification.

In 2023, Taher, et al., [20] IIoT botnet detection using a dependable machine learning model. In this paper, we offer a unique feature selection algorithm, FGOA-kNN, to select the most relevant features. It is based on a hybrid filter and wrapper selection strategy. The Grasshopper algorithm (GOA) is used to reduce the features that are ranked highest in the new technique that is combined with clustering. Additionally, a suggested technique called IHHO chooses and modifies the hyperparameters of the neural network to effectively identify botnets. To improve the global search process for ideal solutions, three enhancements are made to the proposed Harris Hawks algorithm.

In 2022, Waqas, et al., [21] Botnet attack detection using machine learning in cloud-based Internet of Things devices. Investigating cyber security in the face of malware, DDOS, and B-IDS attacks is the goal of this research paper. In order to detect botnet attacks, various machine learning algorithms have been used, including support vector machines, naive Bayes, linear regression, artificial neural networks, decision trees, random forests, fuzzy classifiers, K-nearest neighbors, adaptive boosting, gradient boosting, and tree ensembles.

In 2022, Alrayes, et al., [22] a botnet detection model for the IoT environment is designed using the barnacles mating optimizer with machine learning (BND-BMOML). The BND-BMOML model that is being presented is centered on identifying and recognizing botnets in the context of the Internet of Things. To achieve this, the BND-BMOML model first adopts a data standardization strategy. The BMO algorithm is used in the given BND-BMOML model to choose a useful collection of characteristics. An Elman neural network (ENN) model is used in this study's BND-BMOML model for botnet detection. Lastly, to illustrate the work's originality, the proposed BND-BMOML model employs a chicken swarm optimization (CSO) technique for the parameter tuning procedure.

In 2022, Almuqren, et al., [23] botnet detection using hybrid metaheuristics and machine learning in an IoT context supported by the cloud. The Hybrid Metaheuristics with Machine Learning based Botnet Detection (HMMLB-BND) approach is presented in this paper for the Cloud Aided IoT context. In the context of cloud-based IoT, the proposed HMMLB-BND technique focuses on the identification and categorization of botnet attacks. The Modified Firefly Optimization (MFFO) method is used in the HMMLB-BND technique that is being presented for feature selection. For botnet identification, the HMMLB-BND algorithm employs a hybrid convolutional neural network (CNN)-quasi-recurrent neural network (QRNN) module. Using the chaotic butterfly optimization algorithm (CBOA), the best hyperparameter tuning procedure is carried out.

In 2022, Kumar, et al., [24] early IoT botnet detection based on machine learning and network-edge traffic. We introduce EDIMA, a lightweight IoT botnet detection tool that can be placed at home networks' edge gateways that aims to identify botnets before an attack is launched. A unique two-stage Machine Learning (ML)-based detector designed especially for IoT bot identification at the edge gateway is part of EDIMA. In order to identify individual bots, the ML-based bot detector first uses ML algorithms for classifying aggregate traffic, followed by tests based on the Autocorrelation Function (ACF). A policy engine, a feature extractor, a traffic parser, and a malware traffic database are also included in the EDIMA architecture.

In 2023, Catillo, et al., [25] a deep learning technique for IoT botnet detection that is portable and cross-device. Complex machine learning architectures are used in many of the current intrusion detection system (IDS) concepts for the Internet of Things. These architectures typically offer a single model for each device or assault. The size and dynamic nature of contemporary IoT networks make these methods inappropriate. In order to learn a single IDS model rather than numerous distinct models over the traffic of various IoT devices, this study suggests a novel IoT-driven cross-device technique. Since a semi-supervised strategy is more applicable to unforeseen attacks, it is used. The approach is built on an all-in-one deep autoencoder, which uses regular traffic from many IoT devices to train a single deep neural network. Table I compare the existing papers related to the Botnet attack detection.

TABLE I. COMPARISON OF THE LITERATURE REVIEW PAPERS

Study	**Method**	**Detection Technique**	**Advantages**	**Disadvantages**
Nookala et al. [16]	Logistic Regression	Botnet detection using frequency counting	Simple and efficient method; good for basic botnet detection tasks	May not handle highly complex attacks well due to the simplicity of the frequency counting method.
Alissa et al. [17]	SVM, Random Forest, ResNet50, LSTM	Botnet detection in IoT	Effective for large datasets; can utilize deep learning for more complex attack patterns	Requires large datasets for training; computationally intensive for real-time detection.
Al-Fawa'reh et al. [18]	Deep Reinforcement Learning (DRL)	Malware botnet detection	Enhanced generalizability and robustness; adapts dynamically to evolving malware patterns	Complexity of DRL models may lead to high computational cost and long training times.
Kalakoti et al. [19]	Statistical ML	Botnet detection	Effective for binary and multiclass classification; good for identifying relevant features	The feature selection process can be computationally expensive and may not generalize well across different datasets.
Taher et al. [20]	kNN, Harris Hawks Optimization	IIoT botnet detection	Combines hybrid filter and wrapper methods for better feature selection; effective for IIoT	High computational overhead due to the hybrid approach and complexity of the optimization algorithms.
Waqas et al. [21]	Various ML Algorithms (SVM, ANN, DT, RF, etc.)	Botnet detection	Offers a variety of classifiers for different attack types; flexible and adaptable	Limited by the effectiveness of individual classifiers in handling diverse types of botnet attacks.
Alrayes et al. [22]	Elman Neural Network (ENN)	Botnet detection in IoT	Efficient in IoT environments; uses BMO for effective feature selection	May struggle with real-time detection and the complexity of feature selection using the BMO method.
Almuqren et al. [23]	Hybrid CNN-QRNN	Botnet detection in cloud-based IoT	Combines CNN and QRNN for better detection performance in cloud IoT	High computational demand due to the hybrid neural network and feature selection processes.
Kumar et al. [24]	ML-based two-stage detector	Early IoT botnet detection at edge	Lightweight and fast detection at edge gateways; helps in early detection	May not be effective against sophisticated botnet attacks with complex behaviors or new attack patterns.
Catillo et al. [25]	Deep Autoencoder	Cross-device IoT botnet detection	Uses semi-supervised learning, which is beneficial for handling unforeseen attacks	Challenges in dealing with unforeseen or novel attack types due to the semi-supervised nature of the model.
Study	Methodology	Detection Technique	Advantages	Disadvantages

The increasing number of Internet of Things (IoT) devices has made them a prime target for botnet attacks, presenting a significant challenge for network security. The detection of botnet assaults in IoT environments is critical, yet existing approaches face various limitations in terms of computational efficiency, adaptability to evolving attack patterns, and the ability to handle complex or unforeseen attack types. Many machine learning (ML) and deep learning (DL) techniques have been proposed for botnet detection, utilizing methods like Logistic Regression, SVM, Random Forest, and deep reinforcement learning. However, these methods often struggle with issues such as high computational demands, limited generalizability, and difficulty in real-time detection. Additionally, feature selection and optimization processes, essential for improving detection accuracy, are computationally expensive and may not generalize well across diverse IoT environments.

Thus, there is a need for more efficient and adaptive botnet detection models that can operate effectively in dynamic and resource-constrained IoT environments. These models should be capable of detecting a wide range of attack types, including novel and sophisticated threats, with minimal computation overhead and in real-time. Developing such a model requires addressing the challenges of feature selection, optimization, and ensuring robustness against evolving malware patterns.

III. Proposed Methodology

The DenseSE-ASPPNet is proposed as the botnet detection system that combines advanced pre-processing, feature extraction, and optimization techniques. Pre-processing begins with the cleaning of data from entries that may be irrelevant or missing; then Min-Max normalization of features into a consistent scale to efficiently train the model is applied. For feature extraction, we use the DenseNet169 backbone, allowing for feature reuse through dense connections for the extraction of compact informative representations. In addition, RSE blocks improve channel-wise attention recalibration, which helps the model to focus more on important features. ASPP is used to capture multi-scale spatial patterns, which are very important for botnet activity detection at different resolutions. Finally, the hyperparameters of the model are optimized using the CE-HSOA, which combines global exploration and local exploitation to ensure faster convergence and enhanced robustness. Together, these modules enable DenseSE-ASPPNet to effectively detect botnet activities in IoT networks. The proposed Botnet attack detection model is shown in Fig. 1.

A. *Pre-processing*

Pre-processing in the DenseSE-ASPPNet architecture consists of two key operations: data cleaning and Min-Max normalization. Data cleaning cleans irrelevant, missing, or erroneous entries from the raw network traffic data so that only valid information is utilized. After data cleaning, all features are scaled within a fixed range by applying Min-Max normalization.

1) Data cleaning: The main purpose of data cleaning is to remove unusual data from the original data, such as duplicating, missing, or illegal data. When an experiment is repeated, all duplicate data are eliminated and just the data that appears for the first time are retained. The gaps are filled in by averaging the data from the preceding and subsequent hours. This is shown in Eq. (1),

$$x_i = \frac{x_{i-1}+x_{i+1}}{2} \tag{1}$$

In the padding data, x_i represents the data to be filled, x_{i-1} represents the data from the previous hour, and x_{i+1} represents the data from the next hour. Unlawful data in this experiment are those that have a value of 0 but shouldn't be 0. It is also replaced by the average value of the data from the preceding and following hours, which is determined by Eq. (1).

2) Min- Max normalization: In information processing, data normalization is a crucial step. This entails standardizing data in order to reduce complexity, remove redundancy, and enhance data quality. Usually, this method entails scaling numerical data to a uniform range of values in order to standardize it and facilitate comparison and analysis. In this investigation, the min-max normalization method was employed.

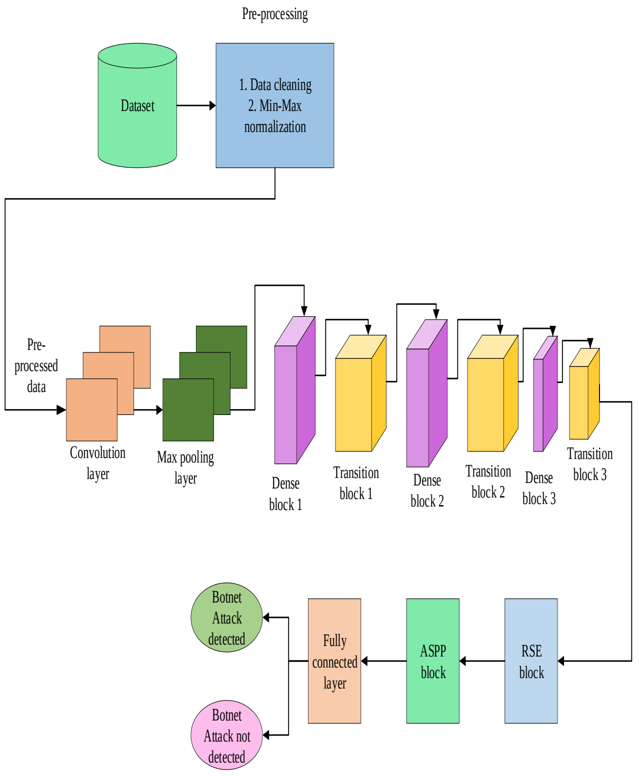

Fig. 1. Block diagram of the proposed Botnet attack detection model.

The initial data is linearly modified using Min-Max normalization. With this method, all scaled data between 0 and 1 is obtained. The following Eq. (2) can be used for this: The relationships between the original data's values are preserved using Min-Max normalization.

$$Z^* = \frac{z-\min(z)}{range(z)} = \frac{z-\min(z)}{max(z)-\min(z)} \tag{2}$$

The minimal value is denoted by min⁡ (z), while range (z) denotes the range between maximum and minimum. The breadth of the interval is 1, and the range of Z^* is within the range [0, 1].

B. *DenseRSE-ASPPNet*

DenseRSE-ASPPNet is proposed as an effective model for botnet detection using advanced feature extraction techniques. Using the DenseNet169 as the backbone, it explores dense connections that allow feature reuse while learning compact and informative representations for the input data. Furthermore, the RSE blocks enhance feature recalibration capabilities by making the model adaptive to important features while key information is preserved through residual learning. In addition, Atrous Spatial Pyramid Pooling (ASPP) is used to capture multi-scale spatial patterns that are critical for botnet activity detection, which can occur at different spatial resolutions. Combining DenseNet, RSE blocks, and ASPP enables the DenseRSE-ASPPNet model to effectively extract relevant features for accurate and robust botnet detection in IoT networks.

Convolutional layers, max pool layers, transition layers, and dense (fully connected) layers make up the DenseNet. ReLU is used throughout the model's design, whereas SoftMax is used to activate the top layer. The maxpool layers reduce the dimensionality of the input, while the convolutional layers recover the image's characteristics. In the stack, the first flattened layer is followed by the fully linked layers. The flatten layer functions as an artificial neural network and receives a single input array. The DenseRSE-ASPPNet model is shown in Fig. 2.

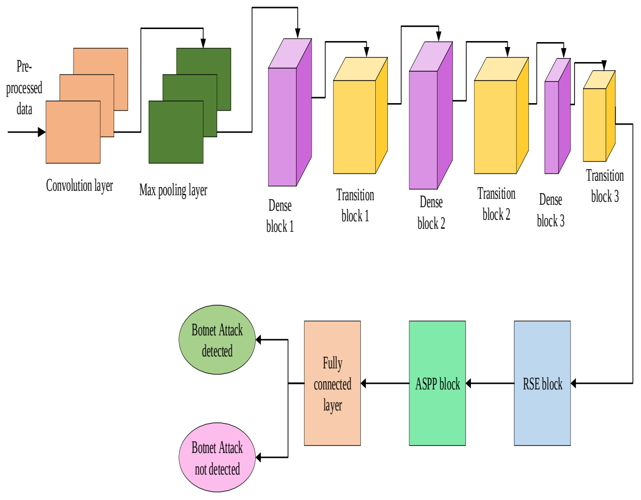

Fig. 2. Architecture of the DenseRSE-ASPPNet model.

1) Convolution layer: To put it simply, an activation occurs when a convolutional layer applies a filter to an input. Continuous application of the filter to an input result in a feature map that shows the intensity of the detected features at different locations within the input. ReLU and other activation methods can then be applied to a feature map that has been created using several filters. Often, the operation between these two entities is a dot product since the filter employed in a convolutional layer is narrower than the input data. Assuming a P×P square neuron element, the outcome of this layer would be (P-m+1)×(P-m+1), followed by a filter of size m×m. The nonlinear input to the unit x_{ij}^{l} is determined by summing the inputs from the layer cells preceding them, as per Eq. (3).

$$x_{ij}^{l} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} \mu_{ab} y_{(i+a)(j+b)}^{l-1} \tag{3}$$

The convolutional layer's implementation of the identified non-linearity is demonstrated by Eq. (4).

$$y_{ij}^{l} = \lambda(x_{ij}^{l}) \tag{4}$$

2) MaxPool layer: Adding a maxpool layer to a CNN is primarily done to reduce the dimensionality of the feature map. The maxpool layer summarizes the features in the region that the pooling layer has filtered, applying a filter on the feature map similarly to the preceding layer. n_h×n_w×n_c, which represent the feature map's height, width, and channels, respectively, are presumed to be present in a feature map. The feature map's dimensions are determined by Eq. (5) when the maximum pooling (〖max〗_p) across the size f and stride s filters is utilized.

$$max_p = \frac{(n_h-f+1)}{s} \times \frac{(n_w-f+1)}{s} \times n_c \tag{5}$$

3) Dense layer: The fully connected layer is where the majority of classification at the network's end occurs. Unlike pooling and convolution, it is a global procedure. A global analysis is performed on the output of all the preceding layers using the information gathered from the feature extraction steps. By doing this, it creates a non-linear blend of the characteristics that are utilized to classify information. The communication between all neurons in a thick layer and all neurons in the layer above it is referred to as strongly coupling in a neural network. A matrix-vector multiplication occurs whenever each neuron in this layer sends information to its matching neuron in the layer underneath. The formula for matrix-vector multiplication is provided in Eq. (6).

$$M.\lambda = \begin{matrix} m_{11} & m_{12} & \cdots & m_{1y} & p_1 \\ m_{21} & m_{22} & \cdots & m_{2y} & p_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ m_{x1} & m_{x2} & \cdots & m_{xy} & p_y \end{matrix} \tag{6}$$

A matrix with dimensions of × y and 1× y, respectively, is represented by the variables M and p in the equation above. Backpropagation can be used to update the previous layer's parameters, which comprise the variable matrix, during training. To backpropagate over the learning rate, which is defined by changing the weights for the layer ly designated by ω^ly and bias represented by the variable B^ly of the neural network, utilize Eq. (7) and Eq. (8) respectively.

$$\omega^{ly} = \omega^{ly} - \alpha \times d\omega^{ly} \tag{7}$$

$$B^{ly} = B^{ly} - \alpha \times dB^{ly} \tag{8}$$

The dω and db are calculated using a chain rule (from the output layer via the hidden layers to the input layer). These are dω and db, which are the partial derivatives of ω and b of the loss function. Eq. (9) through Eq. (12) are utilized to calculate dω and db.

$$d\omega^{ly} = \frac{\partial L}{\partial \omega^{ly}} = \frac{1}{n} dZ^{ly} A^{[ly-1]T} \tag{9}$$

$$\mathrm{dB}^{\mathrm{ly}} = \frac{\partial \mathrm{L}}{\partial \mathrm{B}^{\mathrm{ly}}} = \frac{1}{\mathrm{n}} \sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{dZ}^{\mathrm{ly(i)}} \tag{10}$$

$$\mathrm{dA}^{\mathrm{ly}-1} = \frac{\partial \mathrm{L}}{\partial \mathrm{A}^{[\mathrm{ly}-1]}} = \mathrm{W}^{\mathrm{ly}^{\mathrm{T}}} \mathrm{dZ}^{\mathrm{ly}} \tag{11}$$

$$\mathrm{dZ}^{\mathrm{ly}} = \mathrm{dA}^{\mathrm{ly}} \times \mathrm{g}'(\mathrm{Z}^{\mathrm{ly}}) \tag{12}$$

As per the previously mentioned equations, the layer ly linear activation is represented by the variable Z^ly, and the differential of the Z^ly-related non-linear function is denoted by g^' (Z^ly). The symbol for the nonlinear activation function at the same layer is A^ly.

4) Transition layer: A CNN uses a transition layer to make the model simpler. Usually, a transition layer uses an 11-layer convolution to lower the number of channels and a stride 2 filter to cut the input's height and breadth in half.

5) SE block with residual connection: The SE Block is utilized in this situation due to its simplicity in integrating into any model and its capacity to rectify information loss by recalibrating features with a negligible increase in parameters. By passing the input features via the GAP, the SE-Block-based attention module condenses each channel into a single feature, or scalar value. The two phases of the SE Block are excitation and squeezing. Every channel in the image is made one-dimensional during the squeeze stage by using global average pooling, or GAP. A rectified linear unit (ReLU) and a sigmoid are two completely connected layers that the squeezed vector passes through during the recalibration stage. In order to highlight the key information, the flattened vector is then multiplied by the image that has undergone a 1 × 1 convolution and the weight, which represents squeezed information. An SE Block is depicted in Fig. 3. The SE Block's reduction ratio is a hyperparameter that modifies the number of nodes in the ReLU and fully linked layer. The number of parameters rose as the reduction ratio dropped. The number of parameters dropped as the reduction ratio grew. In other words, it is a hyperparameter associated with variations in computing cost and capacity. The recalibrated output and the input layer are connected by a residual connection that is introduced at the recalibration stage. A direct shortcut between a module's input and output is another design element in the residual connection block that improves the gradient flow during backpropagation while maintaining information. Adding the recalibrated feature map back to the input is how a SE Block that uses residual connections is implemented.

The SE channel attention process involves several important equations. The Squeeze operation uses global average pooling to reduce the input feature map (H×W×C) to 1×1 ×C. The height, breadth, and number of channels of the original feature map are denoted by H, W, and C, respectively. This could be shown in Eq. (13),

$$z_c = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i,j) \tag{13}$$

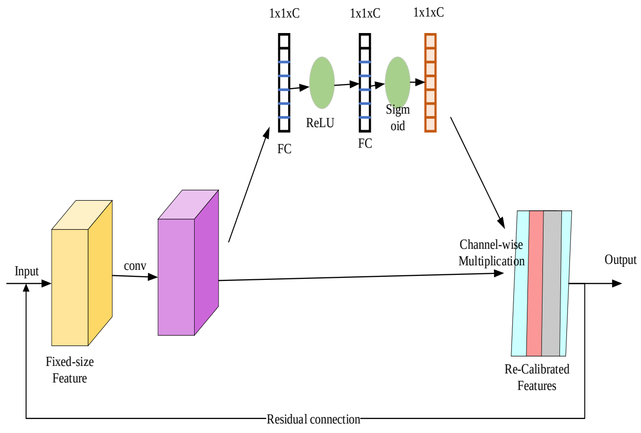

Fig. 3. Block diagram of the RSE block.

In this case, u_c is the feature value of the c^th channel at position (i,j) in the input feature map, and z_c is the Squeeze output of the c^th channel. An activating mechanism and two fully connected layers are used in the Excitation phase to establish channel weights and understand the correlations between channels. In Eq. (14),

$$\mathrm{s} = \sigma(\mathrm{W}_2 \delta(\mathrm{W}_1 \mathrm{z})) \tag{14}$$

where, z is the Squeeze phase's output, δ is the ReLU value, σ is the Sigmoid function, s is the generated channel weight vector, and W_1 and W_2 are learned weight parameters. Furthermore, after the SE attention system is executed, the feature specification is obtained by multiplying the channel's weights by the initial features. In Eq. (15), the recalibration procedure is displayed.

$$y_c = s_c . u_c \tag{15}$$

The input features (u_c) are appended to the recalibrated features (y_c) in order to create a residual connection:

$$\hat{\mathrm{y}}_{\mathrm{c}} = \mathrm{y}_{\mathrm{c}} + \mathrm{u}_{\mathrm{c}} \tag{16}$$

This equation can also be written as:

$$\hat{y}_c = (s_c . u_c) + u_c \tag{17}$$

The addition ensures that the recalibrated features enhance the input features without overwriting the original information, maintaining a balance between recalibration and preservation.

6) ASPP module: The ASPP module's dilated convolution, sometimes referred to as extended convolution or atrous convolution, is distinguished by the addition of gaps between the convolution kernel's constituent pieces. This preserves the original input feature map's height and width while expanding the kernel's receptive field. The convolution kernel's spacing is indicated by the dilation rate. By altering the dilation rate, the filter's receptive field can be adjusted appropriately. Every two convolutional kernel elements are separated by (r-1) zeros, as seen in Fig. 3. k ' = k + (k - 1) × (r - 1) is the kernel's effective size. Among these, r stands for the dilation rate and k for the convolutional kernel's size. Dilated convolution is the same as ordinary convolution when r=1. It can modify the convolutional kernel's receptive field by varying the dilation rate without requiring additional calculations or parameters. The basic dilation model is shown in Fig. 4.

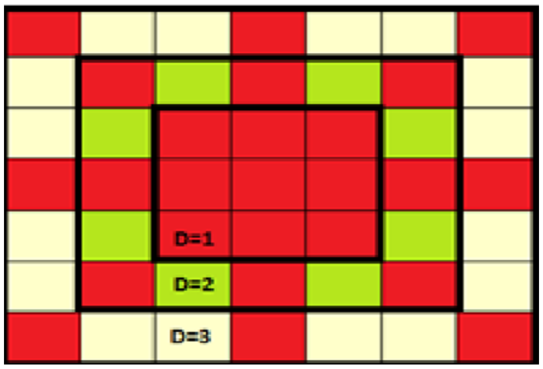

Fig. 4. 3×3 Filter with different dilation rate as 1, 2, and 3.

An ASPP module is added at the network's bottom to extract multi-scale features that will aid the network in comprehending and capturing data at various scales. Spatial Pyramid Pooling (SPP) is the foundation of the enhanced ASPP module. The use of dilated convolutions in place of ordinary convolutions is where ASPP and SPP diverge. This module uses dilated convolutions with varying dilation rates as the final prediction in order to merge multiple receptive field features. The ASPP module makes use of adaptive average pooling in conjunction with four parallel dilated convolutions (with dilation rates of 1, 6, 12, and 18), as illustrated in Fig. 5. Batch normalization (BN), ReLU activation, and a convolution operation make up each dilated convolution. Concat can be used to join parallel networks. To guarantee that the output image size stays the same as the input image size, use BN, ReLU, and ordinary convolution (with a kernel size of 1×1).

7) Fully Connected (FC) layer: The FC layer performs feature aggregation by combining the learned features from different areas of the input image. The network can provide more sophisticated representations by capturing higher-level patterns and correlations between features thanks to this aggregation. In this assignment, the burst assembly is carried out, and the output of the completely linked layer is frequently used to generate final predictions. The proposed model's final layer has two layers, provides the final output for prediction.

8) SoftMax activation layer: Deep learning systems commonly use the softmax activation function to address classification issues. In Eq. (18), where, weight is represented by the variable ω and bias by the variable b over an input vector x, defines the general form of a nonlinear activation function.

$$y = f(\omega \times x + b) \tag{18}$$

The output layer of a convolutional neural network employs the softmax function to estimate the likelihood of each output class. According to the softmax function's specifications, each neuron in the output layer receives a single value. Each of these neurons in the output layer determines the likelihood (or probability) that a certain node will reach the output. When applied to the input, the softmax function is defined over the softmax function Θ. According to Eq. (19), v_i relates to the exponential function of the input vector, represented by e^(v_i), and the exponent function of the output vector, represented by e^(v_o), with m instances.

$$\Theta(z)_x = \frac{e^{v_i}}{\sum_{y=1}^{m} e^{v_o}} \tag{19}$$

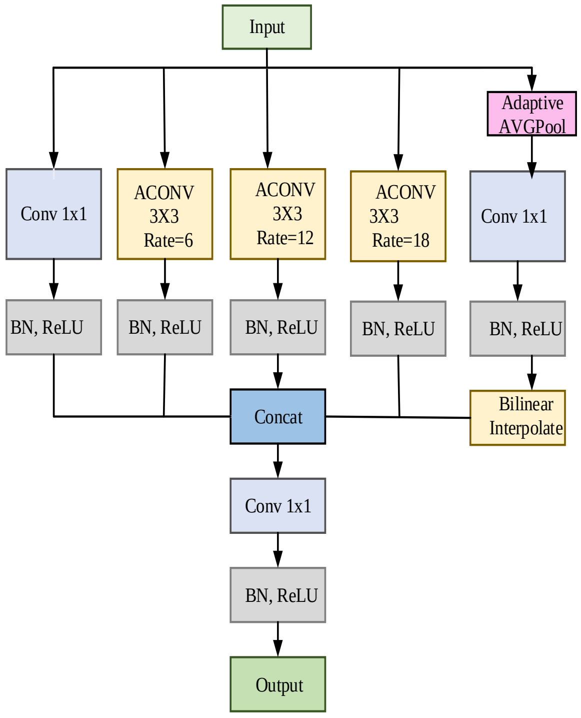

Fig. 5. Structure of the ASPP model.

This work uses softmax as the activation function and the binary cross-entropy loss function as the loss function. Binary cross-entropy has been used in the past to solve binarization challenges. Eq. (20) and Eq. (21) display the binary cross-entropy loss function for a network with n layers.

$$K(\omega, b) = \frac{1}{n}\sum_{i=1}^{n} L(a^{(i)}, a^{(i)}) \tag{20}$$

$$L(\hat{a}, a) = -(a \times \log \hat{a} + (1-a) \times \log(1-\hat{a})) \tag{21}$$

With the variable a representing output class 1 and (1-a) representing output class 0, aˆ represents the probability of output class 1 and (1-a ̂) for the class 0 result. The heatmap for the extracted features is shown in Fig. 6.

C. *Hyper Parameter Tuning of DenseRSE-ASPPNet using CE-HSOA*

Hyperparameter tuning is an important step while optimizing the performance of the model DenseRSE-ASPPNet. The effectiveness of deep learning models, such as DenseRSE-ASPPNet, may be highly reliant on hyperparameters like a learning rate, batch size, number of layers, etc. To effectively find a good combination of hyperparameters, we apply the CE-HSOA.

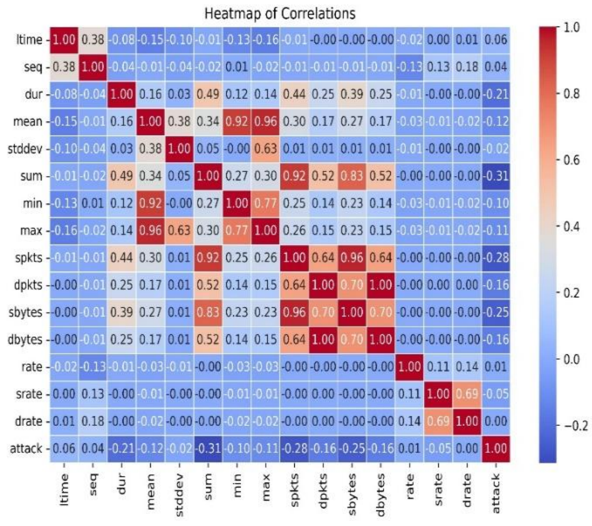

Fig. 6. Heatmap of correlations.

In HSOA, hunting, migration, and mating were important phases. For the search operation, five mechanisms are specified in order to quantitatively model this process. Attacking schools of fish, escaping fish, successfully attacking, attacking smaller squids, and mating Humboldt squids are the components of these methods. As CE-HSOA iterations increase, the search process shifts from exploration to exploitation through mating, bigger squids attacking smaller squids, and fish schools attacking. Fish Escape, however, manages exploration in each iteration.

1) Generating initial population: The CE-HSOA population is made up of fish swarms and Humboldt squid. Algorithm 1 is the pseudocode that CE-HSOA employs to create the first population. As may be observed, Humboldt squid are thought to be the best individuals in the population, whereas fish make up the remainder. Since the Hublot squid is larger and more fit than school fish, this problem is in line with nature.

2) Attack of fish schools: In CE-HSOA, the attack of fish schools is simulated using Eq. (22).

$$XS_{new,i}^{d} = X_b + V_{jet}.\left(-XF_{new,r_1}^{d} - PopAll_{r_2}^{d}\right) \quad (22)$$

According to Eq. (1), $PopAll_{r_2}^{d}$ is the saved r_2^{th} position in the CE-HSOA memory, XF_{new,r_1}^{d} is the position of r_1^{th} fish in d^{th} dimension, V_{jet} is the locomotion velocity parameter, and $XS_{new,i}^{d}$, i is the new position of i^{th} Humboldt squid in d^{th} dimension. Additionally, r_1 and r_2 are random integer numbers between 1 and the size of the PopAll and the population size of fish, respectively. Responsibility for V_{jet}.

3) Successful attack: The new position for Humboldt squid (XS_i) replaces the existing position for Humboldt squid after the new positions for fish and squid have been updated.

$$XS_i^d = \begin{cases} XS_i = XS_{new,i}, & if\ FS_{new,i} < FS_i \\ Successful\ escape, & Otherwise \end{cases} \quad (23)$$

The new and current fitness functions of the i^{th} Humboldt squid are denoted by $FS_{new,i}$ and FS_i in Eq. (23).

4) Successful escape: When the school of fish is attacked by the squid, the fish flee to a randomly chosen spot. The following equation is used in this escape to update the fish's position and velocity.

$$XF_{new,i} = \begin{cases} XF_i + \overrightarrow{rn}.(P_{best} - XF_i).wf, & if\ nfes < 0.1max_{nfes} \\ XS_i + \overrightarrow{rn}.\left(ArchiveX_{r_1} - PopAll_{r_2}\right), & Otherwise \end{cases} \quad (24)$$

The number of function evaluations in Eq. (24) is represented by $nfes$, the maximum number is represented by max_{nfes}, $ArchiveX_{r_1}$ is the r^{th} place in the archive of the best results, XS_i is the i^{th} location of the Humboldt squid, $\overrightarrow{rn}$ is the normal random vector, $wf = Fb$, $XF_{new,i}$ is the new position of i^{th} fish, XF_i is the current position of i^{th} fish, and P_{best} is N top of the best positions. The fitness function of i^{th} fish is F_{f_i}, while F_b is the best fitness function. If the function evaluation counter is in the first generation of this equation, the fish will migrate toward one of the N best solutions. If not, it shifts to a random location.

5) Attack of stronger squids to smallest squids: Fish and Humboldt squid are presumed to be out of the hunt if they are unable to locate a better position in the preceding steps. Thus, the larger Humboldt squid consumes the smaller ones. At this point, the following equation is used to determine the Humboldt squid's location:

$$XS_{new,i}^{d} = XS_{new,i}^{d} + V_{jet_2}.\left(XS_{new,i}^{d} - X_b^d\right) \quad (25)$$

The second velocity parameter in Eq. (25) is V_{jet_2}. Based on this connection, it is assumed that the smaller Humboldt squid is in the best position (X_b^d) and that the larger one goes toward it in order to search for the optimum solutions.

6) Humboldt Squid mating: In CE-HSOA, the egg position is generated using Eq. (26). It was previously used to improve the deferential evolutionary (DE) method.

$$Eggs = \left(\omega.XS + (1 - \omega.P_{best})\right).\gamma + (1-\gamma).pop(r_1,:) + W.\left(pop(r_3,:) - popAll(r_2,:)\right) \quad (26)$$

The Humboldt squid egg mass is represented by the variable Eggs in Eq. (27), while the adaptive weights $\omega, \gamma, and\ W$ govern the search procedure. Between 0 and 1 are $\omega\ and\ \gamma$. This equation can be used to estimate W:

$$\mathrm{W} = \max\{\omega.\gamma, (1-\omega).\gamma, 1-\gamma\} \quad (27)$$

The current study defines the following equations [Eq. (28) and Eq. (29)] for estimating $\omega\ and\ \gamma$:

$$\omega = \mu_\omega + c_1.x \quad (28)$$

$$\gamma = \mu_\gamma + c_2.\overrightarrow{rn} \quad (29)$$

where, the user determines the constant parameters c_1 and c_2. Additionally, in the first generation, μ_ω and μ_γ are vectors with a value of 0.5, and they are updated in subsequent generations in the manner described below:

$$\mu_\omega = \frac{[Diff_F(I).\omega(I)^2]}{[Diff_F(I)].[\omega(I)]} \tag{30}$$

$$\mu_\gamma = \frac{[Diff_F(I).\gamma(I)^2]}{[Diff_F(I)].[\gamma(I)]} \tag{31}$$

where, $\mathrm{Diff_F}$ is the difference between the fitness of Humboldt squids and their eggs, and I is the index indicating which Humboldt squids are more fit than their eggs in Eq. (30) and Eq. (31). The mating motion in the CE-HSOA is performed multiple times because Humboldt squids mate multiple times during their lifetimes, at each generation. Keep in mind that the γ ought to be higher than zero. Therefore, the following equation is used to rectify the γ value if it falls below zero:

$$\gamma = \mu_\gamma + 0.1.\tan(\pi.\mathrm{rnd}) \tag{32}$$

The typical random number rnd in Eq. (32) falls between 0 and 1. The value of x is calculated using Eq. (33):

$$x = \frac{nfes}{max_{nfes}}.\overrightarrow{rnd}^{r.10} \tag{33}$$

In Eq. (33), the normal random vector over right around and the normal random number r are both between 0 and 1, respectively.

7) Control search process with cyclone foraging: There are several parameters that influence the CE-HSOA search process, such as $V_{jet}, V_{jet_2}, x, w_f, W, \omega, and\, \gamma$. To replicate Humboldt squids' shape of locomotion, V_{jet} and V_{jet_2} are used. This is accomplished by using a polynomial function. The third and fourth degrees, respectively, are assigned to the power of this polynomial function for V_{jet} and V_{jet_2}. To calculate V_{jet} and V_{jet_2}, the following formulas are used.

Incorporating the Cyclone Foraging phase from the Manta Ray Optimization algorithm significantly enhances the CE-HSOA's search process. The movement pattern provided in this phase is that of a spiral, thereby increasing the efficiency of exploring space while also decreasing the chances of falling into a local optimum. This strikes a balance between global exploration and local exploitation, ensuring that it is focused on the promising regions for better refinement of the solution. Moreover, due to its adaptive and dynamic nature, this mechanism accelerates convergence and enhances the algorithm's robustness against premature stagnation. Further improvement of versatility is achieved through incorporation of stochastic movements, thereby making CE-HSOA more effective in solving complex, nonlinear, or multimodal optimization problems.

$$V_{jet} = X_{best} + r.(X_{best} - X_i) + \beta.(X_{best} - X_i) \tag{34}$$

$$V_{jet_2} = (X - a_1).(X - a_2).(X - a_3).(X - a_4) \tag{35}$$

where, X_{best} is the best solution, and β is the weight factor. The value of β is given by,

$$\beta = 2e^{r\frac{T-t+1}{T}}.\sin(2\pi r) \tag{36}$$

The parameters a_1, a_2, a_3, and a_4 of the polynomial function that define its shape are found in Eq. (34) and Eq. (35) and can be used to derive X:

$$X = \frac{nfes}{max_{nfes}} \tag{37}$$

where, w_f adjusts the fish's escape radius based on the ratio of the fish's current objective function value to the value of the best objective function. The extent of fish escapement is limited by this parameter during the start of the search, when there are many possible options. However, this parameter approaches one and its effect is mitigated as the number of generations increases. In Eq. (25), the impacts of XS are greater than those of p_{best} because raising the generations in equation 8 raises the value of x. The local optima trap is avoided by CE-HSOA with the aid of these factors. In the mating portion, w_f, ω and γ oversee preventing the ensuing responses becoming convergent too soon. These settings alter the search range and strike a balance between exploration and exploitation based on the objective function's value and the number of generations.

IV. Results and Discussions

This section compares the performances of several classification techniques—Proposed Model, CNN, KNN, SVM, and Logistic Regression—across various metrics such as accuracy, sensitivity, specificity, precision, F-measure, NPV, FPR, FNR, and MCC. From the obtained results, the Proposed Model is seen to be performing better than the other techniques across most of these metrics, signifying superior classification performance.

A. Dataset Description

The N-BaIoT Dataset includes traffic information from nine industrial IoT devices. Of them, seven devices gathered data for eleven classes, while the other two devices gathered data for six classes. The information includes both benign traffic and a range of malicious assaults, including SYN, TCP, UDP, and scan. Within the current version of the dataset, there are 89 csv files totaling 7.58 GB in size, with 1486418 examples of both normal and attack cases. The ten attack and non-attack classifications into which the two botnet attacks, MIRAI and BASHLITE, were divided. These attacks fall into three categories: 1) scan instructions, which are used to identify susceptible IoT devices; 2) ACK, SYN, UDP, and TCP floods; and 3) combo or combination assaults, which are used to establish a connection and send spam to it [26].

B. Overall Comparison of the Proposed Botnet Attack Detection Model

The Table II compares the performance metrics of the Proposed Model, CNN, KNN, SVM, and Logistic Regression, highlighting the superiority of the Proposed Model across all evaluated criteria.

TABLE II. COMPARISON OF THE PERFORMANCE METRICS

Techniques	Sensitivity	Specificity	Accuracy	Precision	F-Measure	NPV	FPR	FNR	MCC
Proposed	0.9640	0.9995	0.9900	0.9878	0.9842	0.9964	0.0188	0.0548	0.9726
CNN	0.9377	0.9940	0.9744	0.9825	0.9856	0.9934	0.0283	0.1301	0.9608
KNN	0.9280	0.9861	0.9600	0.9767	0.9798	0.9901	0.0749	0.1550	0.8839
SVM	0.9054	0.9789	0.9484	0.9695	0.9755	0.9800	0.0777	0.1432	0.8760
Logistic Regression	0.8299	0.9344	0.9219	0.8980	0.8976	0.9769	0.0637	0.2172	0.8092

The Proposed Model shows the highest sensitivity, specificity, accuracy, precision, F-measure, NPV, and MCC at 0.9640, 0.9995, 0.9900, 0.9878, 0.9842, 0.9964, and 0.9726, respectively, and lowest FPR and FNR values of 0.0188 and 0.0548, respectively, indicating effective minimization of false classifications. CNN is the second-best performer with high sensitivity of 0.9377, specificity of 0.9940, and accuracy of 0.9744, but higher error rates than the Proposed Model. KNN shows average performance, with acceptable sensitivity (0.9280) and specificity (0.9861), but high FPR (0.0749) and FNR (0.1550). SVM further drops the sensitivity at 0.9054 and specifically at 0.9789, along with a decreased MCC at 0.8760. The Logistic Regression performs the worst, at the lowest sensitivity (0.8299), specificity (0.9344), and MCC (0.8092), and the highest FPR (0.0637) and FNR (0.2172). In general, the Proposed Model significantly outperforms the other alternatives, demonstrating its strength and effectiveness in classification tasks.

C. *Accuracy, Sensitivity and Specificity*

The Table II shows a comparative performance of accuracy, sensitivity, and specificity for different classification techniques. The Proposed Model achieves the highest accuracy (0.9900), signifying its superior ability to classify cases correctly, both positive and negative. CNN follows with high accuracy at 0.9744, while KNN, SVM, and Logistic Regression follow with progressively lower accuracies of 0.9600, 0.9484, and 0.9219, respectively.

Sensitivity, measuring the model to correctly identify positive cases is also highest for the Proposed Model (0.9640), as it signifies the efficiency in minimizing the false negatives. On the other hand, CNN indicates a competitive sensitivity of (0.9377), while KNN indicates somewhat lower at 0.9280; whereas SVM, and Logistic Regression indicate extremely poor sensitivity in detecting positive cases at 0.9054 and 0.8299, respectively. Specificity, which measures the accuracy in detection of negative cases, approaches near perfection for the Proposed Model (0.9995), thus reflecting an excellent ability to reduce false positives. CNN (0.9940) and KNN (0.9861) are also highly specific, while SVM (0.9789) and Logistic Regression (0.9344) performed much weaker. Fig. 7 shows accuracy, sensitivity and specificity values.

D. *Precision and F-Measure*

The Table II also reports Precision and F-Measure, two important metrics that reflect the performance of a model in handling positive classifications. Precision measures the proportion of correctly identified positive cases out of all predicted positives, which is a measure of the ability of the model to minimize false positives. The Proposed Model has the highest precision at 0.9878, which means it can very well classify true positives while keeping false positives at bay. The accuracy of CNN is 0.9825, whereas, KNN follows with 0.9767 and then comes SVM with 0.9695, and then Logistic Regression shows the least accuracy with 0.8980. The F-Measure is the harmonic means of precision and sensitivity. It offers a comprehensive view of the performance of the model in correctly identifying positive cases by weighing the trade-off between these two measures. The Proposed Model has the best F-Measure of 0.9842, which indicates its excellent balance between high precision and sensitivity. CNN is also performing well with an F-Measure of 0.9856, which is slightly higher than its precision due to its strong sensitivity. KNN and SVM have moderate F-Measure values at 0.9798 and 0.9755, respectively, while Logistic Regression lags far behind at 0.8976. Precision and F-Measure values are shown in Fig. 8.

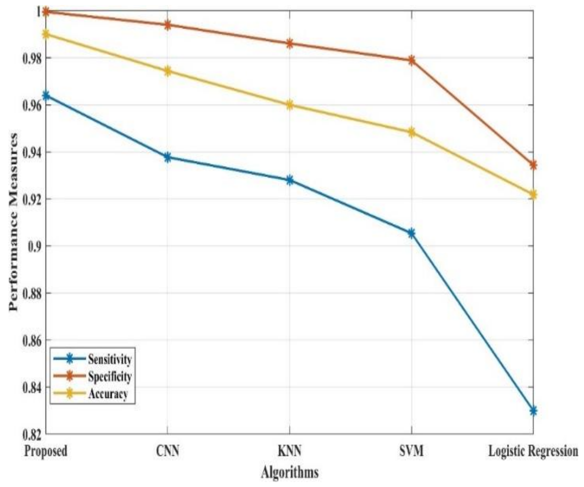

Fig. 7. Comparison of the accuracy, sensitivity and specificity values.

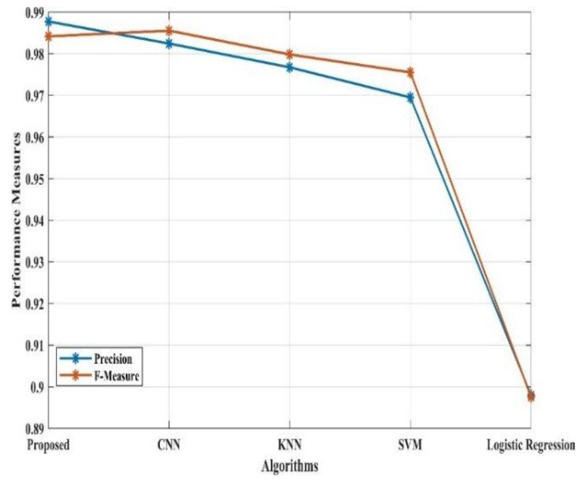

Fig. 8. Comparison of the precision and F-Measure values.

E. NPV and MCC

The Table II also shows Negative Predictive Value (NPV) and Matthews Correlation Coefficient (MCC), furthering the interpretation of the model's performance. NPV is defined as the proportion of true negatives in all predicted negatives, thus representing how well the model can predict negative cases with minimal false negatives. The Proposed Model attains the highest NPV (0.9964), indicating its high reliability in terms of true negatives. CNN is followed by a strong NPV of 0.9934, followed by KNN at 0.9901, SVM at 0.9800, and Logistic Regression at 0.9769, which means that the performance is declining, and Logistic Regression has the worst ability to classify negative cases. MCC is a comprehensive metric which calculates the correlation between the true and predicted values with all possible outcomes: true positives, true negatives, false positives, and false negatives. The Proposed Model has the highest MCC of 0.9726, which means the model is well-balanced and robust in its prediction. CNN has a high MCC of 0.9608, while KNN and SVM have moderate correlation values at 0.8839 and 0.8760, respectively. Logistic Regression with the lowest MCC is at 0.8092, which reflects the weakest overall predictive power. The NPV and MCC values are shown in Fig. 9.

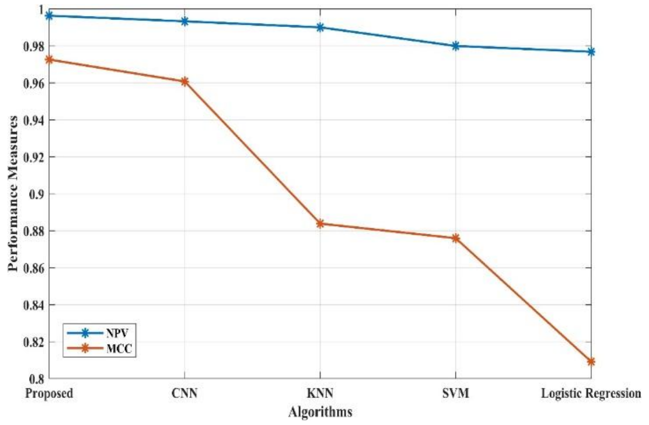

Fig. 9. Comparison of the NPV and MCC values.

F. FPR and FNR

The Table II evaluates False Positive Rate (FPR) and False Negative Rate (FNR), which evaluate the model's error rates in specific contexts. FPR is the proportion of the false positives to all true negatives, indicating the capacity of the model to get negative cases wrongly classified as positives. The Proposed Model acquired the lowest FPR, which is 0.0188, thereby demonstrating their outstanding capability to minimize false positive and correctly classify negative instances. CNN (0.0283) has an FPR that is slightly above KNN (0.0749), SVM (0.0777), and Logistic Regression (0.0637). Though logistic regression does better than both KNN and SVM in its FPR, it significantly lags behind the proposed model and CNN.

Finally, FNR represents how many of the actual positives in the set were not discovered as positives by the model-thus representing the model's rate of missing true positive occurrences. The Proposed Model has the lowest FNR of 0.0548, indicating its higher efficiency in terms of identifying the right cases with fewer misses. CNN comes next with an FNR of 0.1301, while KNN has an FNR of 0.1550, SVM 0.1432, and Logistic Regression 0.2172, showing higher rates and a greater possibility of missing true positives. FPR and FNR values are compared in Fig. 10.

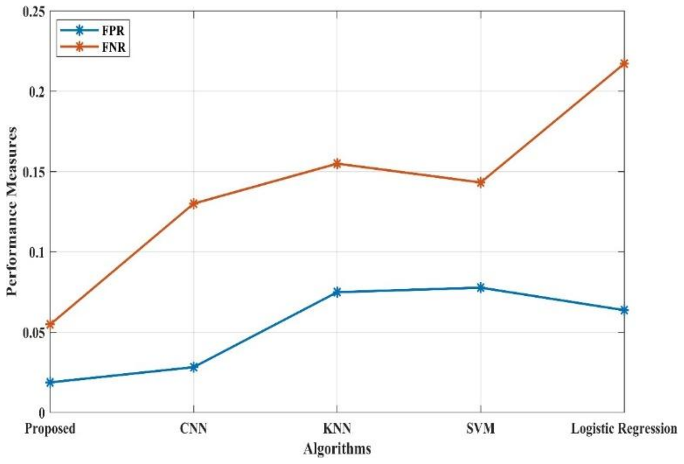

Fig. 10. Comparison of the FPR and FNR values.

V. Conclusion

In conclusion, the DenseRSE-ASPPNet model gives the best and most efficient approach to botnet detection in IoT networks, surpassing other traditional machine learning techniques. Through the use of advanced methods such as the DenseNet169 backbone, RSE blocks, and ASPP, it can efficiently extract informative features and capture multi-scale spatial patterns. Optimized for hyperparameters of the model, applying the CE-HSOA, which boosts the results to have more robust and faster convergence. Therefore, a model that yields better Metrics and shows a high percentage of correctness for identifying bot activities while having very less errors involving false positives and false negatives.

The performance comparison highlights the advantages of DenseRSE-ASPPNet over other models such as CNN, KNN, SVM, and Logistic Regression. The proposed model achieves the highest Accuracy (0.9900) and Sensitivity (0.9640), demonstrating its strong ability to correctly identify botnet traffic. Its Specificity (0.9995) and Precision (0.9878) further showcase its reliability in minimizing false positives while maintaining high detection performance. In contrast, models like CNN and SVM show lower performance, particularly in terms of FNR, with SVM having an FNR of 0.1432. KNN also struggles with a higher False Positive Rate (FPR) of 0.0749, indicating that it is less effective in distinguishing botnet traffic. Logistic Regression exhibits the lowest performance across most metrics, especially in Sensitivity and Accuracy, underscoring its limitations for complex tasks like botnet detection. Overall, the results demonstrate that DenseRSE-ASPPNet provides a significant improvement in botnet detection performance, making it a highly effective solution for securing IoT networks.

Acknowledgment

I am grateful to Dr. Talal Alwahaibi Dean College of Engineering at Ashariqiyah University Ibra for their encouragement and assistance in helping me finish this research and also support of Ashariqiyah University Ibra Oman.

References

[1] Nasir, M.H., Arshad, J. and Khan, M.M., 2023. Collaborative device-level botnet detection for internet of things. Computers & Security, 129, p.103172.

[2] Li, R., Li, Q., Huang, Y., Zhang, W., Zhu, P. and Jiang, Y., 2022, September. Iotensemble: Detection of botnet attacks on internet of things. In European Symposium on Research in Computer Security (pp. 569-588). Cham: Springer Nature Switzerland.

[3] Mudassir, M., Unal, D., Hammoudeh, M. and Azzedin, F., 2022. Detection of botnet attacks against industrial IoT systems by multilayer deep learning approaches. Wireless Communications and Mobile Computing, 2022(1), p.2845446.

[4] Ali, M.H., Jaber, M.M., Abd, S.K., Rehman, A., Awan, M.J., Damaševičius, R. and Bahaj, S.A., 2022. Threat analysis and distributed denial of service (DDoS) attack recognition in the internet of things (IoT). Electronics, 11(3), p.494.

[5] Nadeem, M.W., Goh, H.G., Aun, Y. and Ponnusamy, V., 2023. Detecting and mitigating botnet attacks in software-defined networks using deep learning techniques. IEEE Access, 11, pp.49153-49171.

[6] Rehman Javed, A., Jalil, Z., Atif Moqurrab, S., Abbas, S. and Liu, X., 2022. Ensemble adaboost classifier for accurate and fast detection of botnet attacks in connected vehicles. Transactions on Emerging Telecommunications Technologies, 33(10), p.e4088.

[7] Khanday, S.A., Fatima, H. and Rakesh, N., 2023. Towards the Development of an Ensemble Intrusion Detection Model for DDoS and Botnet Mitigation using the IoT-23 Dataset. Journal of Harbin Engineering University, 44(5).

[8] Maha, A.J., Al-Shurman, M. and Al-Duwairi, B., Attention-based deep learning approach for detecting IoT botnet-based distributed denial of service attacks.

[9] Alshahrani, S.M., Alrayes, F.S., Alqahtani, H., Alzahrani, J.S., Maray, M., Alazwari, S., Shamseldin, M.A. and Al Duhayyim, M., 2023. IoT-Cloud Assisted Botnet Detection Using Rat Swarm Optimizer with Deep Learning. Computers, Materials & Continua, 74(2).

[10] Hoang, X.D. and Vu, X.H., 2022. An improved model for detecting DGA botnets using random forest algorithm. Information Security Journal: A Global Perspective, 31(4), pp.441-450.

[11] Onyema, E.M., Kumar, M.A., Balasubaramanian, S., Bharany, S., Rehman, A.U., Eldin, E.T. and Shafiq, M., 2022. A security policy protocol for detection and prevention of internet control message protocol attacks in software defined networks. Sustainability, 14(19), p.11950.

[12] Attou, H., Mohy-eddine, M., Guezzaz, A., Benkirane, S., Azrour, M., Alabdultif, A. and Almusallam, N., 2023. Towards an intelligent intrusion detection system to detect malicious activities in cloud computing. Applied Sciences, 13(17), p.9588.

[13] Madhu, B., Chari, M.V.G., Vankdothu, R., Silivery, A.K. and Aerranagula, V., 2023. Intrusion detection models for IOT networks via deep learning approaches. Measurement: Sensors, 25, p.100641.

[14] Abu Bakar, R. and Kijsirikul, B., 2023. Enhancing Network Visibility and Security with Advanced Port Scanning Techniques. Sensors, 23(17), p.7541.

[15] Lawrence, H., Ezeobi, U., Tauil, O., Nosal, J., Redwood, O., Zhuang, Y. and Bloom, G., 2022. CUPID: A labeled dataset with Pentesting for evaluation of network intrusion detection. Journal of Systems Architecture, 129, p.102621.

[16] Nookala Venu, D., Kumar, A. and Rao, M.A.S., 2022. Botnet attacks detection in internet of things using machine learning. NeuroQuantology, 20(4), pp.743-754.

[17] Alissa, K., Alyas, T., Zafar, K., Abbas, Q., Tabassum, N. and Sakib, S., 2022. Botnet attack detection in iot using machine learning. Computational Intelligence and Neuroscience, 2022(1), p.4515642.

[18] Al-Fawa'reh, M., Abu-Khalaf, J., Szewczyk, P. and Kang, J.J., 2023. MalBoT-DRL: Malware botnet detection using deep reinforcement learning in IoT networks. IEEE Internet of Things Journal.

[19] Kalakoti, R., Nõmm, S. and Bahsi, H., 2022. In-depth feature selection for the statistical machine learning-based botnet detection in IoT networks. IEEE Access, 10, pp.94518-94535.

[20] Taher, F., Abdel-Salam, M., Elhoseny, M. and El-Hasnony, I.M., 2023. Reliable machine learning model for IIoT botnet detection. IEEE Access, 11, pp.49319-49336.

[21] Waqas, M., Kumar, K., Laghari, A.A., Saeed, U., Rind, M.M., Shaikh, A.A., Hussain, F., Rai, A. and Qazi, A.Q., 2022. Botnet attack detection in Internet of Things devices over cloud environment via machine learning. Concurrency and Computation: Practice and Experience, 34(4), p.e6662.

[22] S. Alrayes, F., Maray, M., Gaddah, A., Yafoz, A., Alsini, R., Alghushairy, O., Mohsen, H. and Motwakel, A., 2022. Modeling of botnet detection using barnacles mating optimizer with machine learning model for Internet of Things environment. Electronics, 11(20), p.3411.

[23] Almuqren, L., Alqahtani, H., Aljameel, S.S., Salama, A.S., Yaseen, I. and Alneil, A.A., 2023. Hybrid metaheuristics with machine learning based botnet detection in cloud assisted internet of things environment. IEEE Access.

[24] Kumar, A., Shridhar, M., Swaminathan, S. and Lim, T.J., 2022. Machine learning-based early detection of IoT botnets using network-edge traffic. Computers & Security, 117, p.102693.

[25] Catillo, M., Pecchia, A. and Villano, U., 2023. A deep learning method for lightweight and cross-device IoT botnet detection. Applied Sciences, 13(2), p.837.

[26] Dataset is taken from https://www.kaggle.com/datasets/mkashifn/nbaiot-dataset.