International Journal of Safety and Security Engineering
Vol. 16, No. 10, October, 2026, pp. 2323-2335
Journal homepage: http://iieta.org/journals/ijsse

End-to-End Deep Learning-Driven Framework for IoT Botnet Attack Detection and Intelligent Mitigation

Ashok Kumar V*, N. Vijayaraj

Department of Computer Science and Engineering, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai 600062, India

Corresponding Author Email: ashokvijay872@gmail.com

Copyright: ©2026 The authors. This article is published by IIETA and is licensed under the CC BY 4.0 license (http://creativecommons.org/licenses/by/4.0/).

https://doi.org/10.18280/ijsse.161011

Received: 29 August 2026
Revised: 9 October 2026
Accepted: 20 October 2026
Available online: 31 October 2026

Keywords:
IoT, botnet, attack detection, AI, mitigation strategy

ABSTRACT

The widespread use of Internet of Things (IoT) devices has increased their vulnerability to botnet-driven cyber threats due to resource constraints and limited security mechanisms. Existing detection methods often rely on static rules and fixed thresholds, which may produce false alarms and fail to adapt to evolving attacks. To address these limitations, this paper proposes an end-to-end deep learning (DL) framework for IoT botnet detection and intelligent mitigation comprising preprocessing, detection, and mitigation stages. We first apply robust scaling to reduce the influence of outliers in network traffic. The detection stage employs a Modified Subnetwork-based Siamese Neural Network (MS-SNN) with an Adaptive Channel-Spatial Attention (ACSA) mechanism to enhance discriminative feature learning and similarity-based classification. For detected attacks, the Entropy with Adaptive Thresholding-based Mitigation (EATM) technique dynamically adjusts to traffic variations for effective attack isolation. Experimental results show that the proposed framework achieves 98.3% detection accuracy with 80% training data and demonstrates improved mitigation robustness compared with baseline approaches.

1. INTRODUCTION

Internet of Things (IoT) is a network of interconnected smart devices that communicate over the Internet to exchange data and perform tasks automatically [1]. By enabling remote monitoring, control and automation, IoT technology has transformed various sectors such as smart homes, healthcare, industrial automation, transportation and smart cities [2]. Its widespread adoption has improved operational efficiency, service quality and overall user convenience [3].

Despite these advantages, the continuous exchange of sensitive information among IoT devices introduces significant security and privacy challenges [4]. Due to their interconnected nature and limited computational resources, IoT devices are vulnerable to cyberattacks such as eavesdropping, unauthorized access and data theft [5]. These security weaknesses can lead to financial losses, disruption of critical services and, in some cases, risks to human safety [6]. To address these challenges, Intrusion Detection Systems (IDS) have been widely developed to identify and prevent malicious activities in IoT networks [7]. However, the increasing sophistication of cyberattacks highlights the need for faster, more accurate and resource-efficient intrusion detection techniques capable of providing reliable protection for modern IoT environments [8].

A botnet is a network of compromised devices that is remotely managed by an attacker, who is commonly referred to as the botmaster. The communication between the botmaster and the infected devices takes place through a Command and Control (C&C) server, which coordinates the activities of the botnet [9]. The lifecycle of a botnet generally includes several stages, such as device infection, communication with the C&C server, execution of malicious tasks and continuous maintenance [10]. Initially, malware infects vulnerable IoT devices and installs additional malicious components required for the attack. Once connected to the C&C infrastructure, the compromised devices receive commands to perform activities such as data theft and distributed denial-of-service (DDoS) attacks. To ensure long-term control and adapt to changing environments, the botmaster regularly updates the malware and modifies its behavior [11].

A more advanced form of this threat is the zero-day botnet, which exploits previously undiscovered vulnerabilities in software or hardware [12]. Since these security flaws are unknown to developers and security vendors, zero-day botnets can bypass existing defence mechanisms and remain undetected until the vulnerabilities are identified and patched [13]. Among these threats, botnet attacks have become a major concern that highlights the importance of developing effective detection and prevention techniques [14]. Consequently, researchers have explored a variety of Machine Learning (ML)-based approaches to improve the identification and mitigation of botnet activities in IoT environments. Despite their promising performance, these approaches often involve high computational complexity and depend on large volumes

of accurately labeled data for effective training. Their performance can also decline when working with limited or imbalanced datasets. These challenges reduce their efficiency and limit their practical deployment.

The main contribution of this work is as follows:

•Proposing an enhanced Siamese Neural Network architecture that overcomes the embedding quality limitations of standard Siamese networks, which enables more reliable similarity-based attack detection even with limited labeled data.

•Designing an entropy-based adaptive threshold mechanism that automatically adjusts to real-time traffic fluctuations, overcoming the false positive limitations of fixed-threshold mitigation strategies.

The remaining section is organized as follows: Section 2 discusses the related work on IoT botnet detection and mitigation techniques. Section 3 presents the proposed deep learning (DL)-driven framework. Section 4 offers the experimental investigation and comparative evaluation of the proposed framework against existing baseline approaches. Finally, Section 5 concludes the paper.

2. LITERATURE REVIEW

Existing research on IoT botnet detection can be broadly grouped into four methodological categories: feature-selection and conventional machine-learning approaches, DL and hybrid architectures, federated learning approaches, and mitigation-oriented approaches.

(a) Feature-selection and conventional machine-learning approaches

Feature-selection-based machine-learning approaches have been widely explored to improve IoT botnet detection by identifying informative network traffic features and reducing dimensionality. Existing studies demonstrate the effectiveness of combining hybrid feature selection and ML, ensemble learning, correlation analysis, optimization algorithms, and Lasso-based selection with conventional classifiers [15-17]. These techniques help eliminate irrelevant or redundant features and improve classification efficiency. However, their dependence on predefined feature-selection procedures may restrict their ability to capture complex, nonlinear, and evolving traffic patterns. Furthermore, the quality of detection may depend heavily on the selected feature subset, potentially affecting generalization across heterogeneous IoT network environments.

(b) DL and hybrid architectures

To address the limitations of conventional feature-based approaches, DL architectures have been investigated for automatically learning representative features from IoT network traffic. Hybrid architectures combining 1D-CNNs with attention mechanisms and transformers, as well as LSTM autoencoders integrated with multilayer perceptrons, have demonstrated the ability to capture important traffic characteristics and sequential dependencies [18, 19]. Furthermore, reinforcement-learning-based approaches, such as MalBoT-DRL [20], have explored the integration of intelligent detection and response capabilities. Despite these advances, the use of complex neural architectures and multiple learning components can increase computational and training overhead. Such requirements may limit their practical deployment in resource-constrained IoT environments, particularly where low-latency detection is essential. environments.

(c) Federated learning approaches

Federated learning has emerged as an alternative learning paradigm for distributed IoT botnet detection, particularly in environments where transferring raw network traffic to a centralized server raises privacy concerns. Comparative investigations involving handcrafted-feature-based ML, centralized DL, and federated DL highlight the potential of distributed model training for privacy-sensitive IoT environments [21]. However, federated learning introduces additional challenges related to communication overhead and distributed model training.

(d) Mitigation and adaptive defence approaches

Beyond detection, effective IoT botnet defence requires systematic threat analysis and adaptive mitigation capabilities. Existing threat-modeling research, particularly the I3TM framework, provides a structured analysis of IoT botnet attacks by identifying attack tactics and techniques across different attack stages [22]. Such frameworks contribute to understanding attack behaviors and identifying potential defensive strategies. Nevertheless, threat modeling alone does not necessarily provide an automated, real-time mitigation mechanism. However, concrete IoT-specific mitigation mechanisms remain future work, and continuous updates are required to address evolving attack behaviors.

Overall, the literature progresses from feature selection and conventional ML to DL, federated learning, and mitigation approaches. However, these methods are mostly investigated separately, with limited integration of deep botnet detection and adaptive mitigation in a unified framework. This motivates the proposed end-to-end framework combining preprocessing, Modified Subnetwork-based Siamese Neural Network (MS-SNN)-based detection, and Entropy with Adaptive Thresholding-based Mitigation (EATM)-based mitigation for IoT environments.

Table 1 presents a critical review of existing works on IoT botnet detection that summarizes the methods used, their key benefits and their associated shortcomings.

Table 1. Review of existing works

Citation	Methods	Benefits	Shortcomings
[18]	1D-CNN and Transformer	Captures important traffic patterns	High complexity
[21]	ML, Centralized DL, Federated DL	Supports distributed detection	Communication overhead
[15]	Hybrid feature selection and ML	Reduces irrelevant features	Depends on selected features
[16]	Feature selection and Ensemble ML	Improves classification	Feature-engineering dependency
[17]	Correlation, GND optimization and Lasso	Effective feature selection	Limited representation learning
[20]	MalBoT-DRL	Supports adaptive detection/response	Computational complexity
[19]	LSTM Autoencoder and MLP	Learns sequential features	Training cost
[22]	I3TM	Supports threat mitigation	Requires continuous updates

Note: 1D-CNN: One-Dimensional Convolutional Neural Network; ML: Machine Learning; DL: deep learning; GND: Generalized Normal Distribution; MalBoT-DRL: Malware Botnet with Deep Reinforcement Learning; I3TM: IoT Targeting-Threat Modeling.

These limitations highlight the need for an integrated and efficient IoT botnet detection and mitigation framework.

•Existing DL models, such as 1D-CNN with self-attention [18] and LSTM+MLP [19], provide effective botnet detection but require high computational resources, limiting deployment on resource-constrained IoT devices.

•The integration of multiple DL models increases processing overhead, making real-time botnet detection challenging in constrained IoT environments.

•Existing mitigation frameworks, such as I3TM [22], require continuous updates and expert knowledge, making them resource-intensive and less suitable for dynamic IoT traffic conditions.

•These limitations highlight the need for a lightweight and accurate detection model combined with a low-overhead, adaptive mitigation mechanism capable of responding to real-time IoT traffic variations.

3. PROPOSED METHODOLOGY

The rapid proliferation of IoT devices has significantly expanded the attack surface available to malicious actors. Most IoT devices are resource-constrained and often developed with minimal security mechanisms, which makes them easy targets for compromise. Once affected, these devices can be recruited into large-scale botnets and used to launch data exfiltration or further malware propagation.

Traditional signature-based IDS struggle to keep pace with evolving botnet tactics that often fail against zero-day attacks and heterogeneous IoT traffic patterns. Moreover, most existing frameworks focus solely on detection without providing an integrated, automated mitigation mechanism. The DL-based approaches have shown promise for similarity-based botnet detection. However, their performance heavily depends on embedding quality; weak feature extractors lead to unreliable similarity scores and degraded accuracy. Additionally, static, fixed-threshold mitigation strategies are prone to false positives during traffic bursts and false negatives against evolving attacks.

The framework processes IoT network traffic in three stages (Figure 1). First, the preprocessing step normalizes the raw data using robust scaling that makes it resistant to outliers common in IoT traffic. This cleaned data then flows into the detection stage, where an MS-SNN is enhanced with an Adaptive Channel Spatial Attention (ACSA) module that extracts stronger and more discriminative feature embeddings than a standard Siamese network would, since the attention mechanism helps the model focus on the most important feature channels and spatial patterns. Similarity between paired samples is then measured using Euclidean distance, and the classification is done through a voting mechanism that picks the closest matching class.

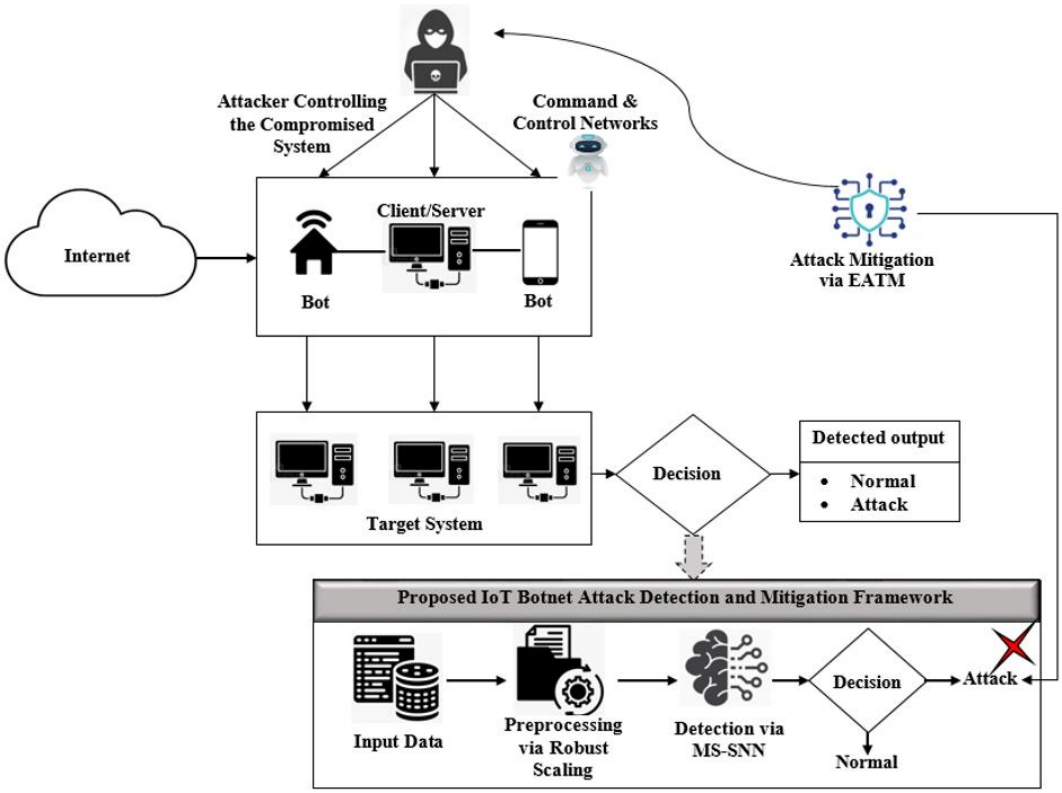


Figure 1. Outline of proposed framework

Once an attack is flagged, the framework moves to mitigation, where instead of using a fixed detection threshold, this work applies an EATM approach. This method monitors traffic entropy at each network switch and weighs different traffic features by how informative they are and compares current behavior against a self-adjusting threshold to decide whether a node is malicious. If confirmed, the node is isolated and blocked; if not, it is treated as normal.

3.1 Preprocessing

The preprocessing stage prepares raw IoT network traffic data (let us say, Dt^{Ntw}) before it is passed to the detection model. Since IoT traffic often contains noisy values and outliers caused by irregular device behavior or sensor faults, it is important to normalize the data in a way that is not distorted by these extreme values.

For this, the framework uses robust scaling [23], which normalizes each feature using the median and interquartile range (IQR) instead of the mean and standard deviation used in conventional scaling methods. The robust scaling is formulated using Eq. (1).

$$Dt^{Ntw''} = \frac{Dt^{Ntw} - Median(Dt^{Ntw})}{IQR} \quad (1)$$

where, the interquartile range (IQR) is computed as $IQR = Q_3(Dt^{Ntw}) - Q_1(Dt^{Ntw})$. Here, $Q_3(Dt^{Ntw})$ and $Q_1(Dt^{Ntw})$ denotes the 75th and 25th percentiles of the feature distribution, respectively.

Thus, the preprocessed data is indicated by $Dt^{Ntw''}$. This makes robust scaling particularly well-suited for IoT botnet traffic, which frequently contains sudden bursts or anomalies even during normal operation. The resulting normalized data has a more consistent and stable scale across features, which improves the quality of the embeddings learned by the detection model in the next stage.

3.2 Detection

After preprocessing, the normalized data is fed into the detection stage, where an MS-SNN is used to identify botnet attack patterns. Unlike conventional classifiers that rely on large volumes of labeled data for each class, a Siamese neural network [24] learns by comparing pairs of inputs and determining how similar they are. It consists of two identical subnetworks that share the same architecture and weights and are trained synchronously. Instead of directly predicting a class label, the network processes paired data samples and outputs a similarity score indicating whether the two inputs belong to the same class. This makes it especially effective for botnet detection, where labeled attack samples are often limited.

However, the performance of a standard Siamese network depends heavily on the quality of the embeddings produced by its feature extractor. If the extractor fails to capture discriminative patterns in the traffic data, the resulting similarity scores become unreliable, which reduces the overall detection accuracy. To address this, an MS-SNN framework has been introduced that redesigns the subnetwork architecture.

3.2.1 Modified Subnetwork-based Siamese Neural Network

The proposed MS-SNN framework (Figure 2) first concerns the preprocessed data that is split into training and testing sets, where each of which is fed into its own improved subnetwork named Artificial Neural Network (ANN). These two identical branches share the same weights and ensure both learn consistent feature representations. Each subnetwork processes its respective input and produces a feature vector. These two embeddings, one from the training branch and another from the testing branch, are then compared using a similarity check based on Euclidean distance, and the result feeds into the evaluation stage. The Euclidean distance [25] is computed using Eq. (2) and Eq. (3).

$$Dst(u_i, u_j) = \|v_i - v_j\|_2 \quad (2)$$

$$Dst(u_i, u_j) = \sqrt{\sum_{k=1}^{d} (v_i^k - v_j^k)^2} \quad (3)$$

During evaluation, the similarity between its embedding and the embeddings of every class pair is calculated for each testing instance. A voting mechanism is then used to select the class pair with the highest similarity score. If the correct class receives the highest vote, the instance is marked as a correct classification; otherwise, it is marked as a wrong classification. Together, this pipeline takes preprocessed data through shared-weight subnetworks to generate embeddings and compares them via similarity scoring. Finally, this approach uses a voting-based decision process to determine classification accuracy for each test sample.

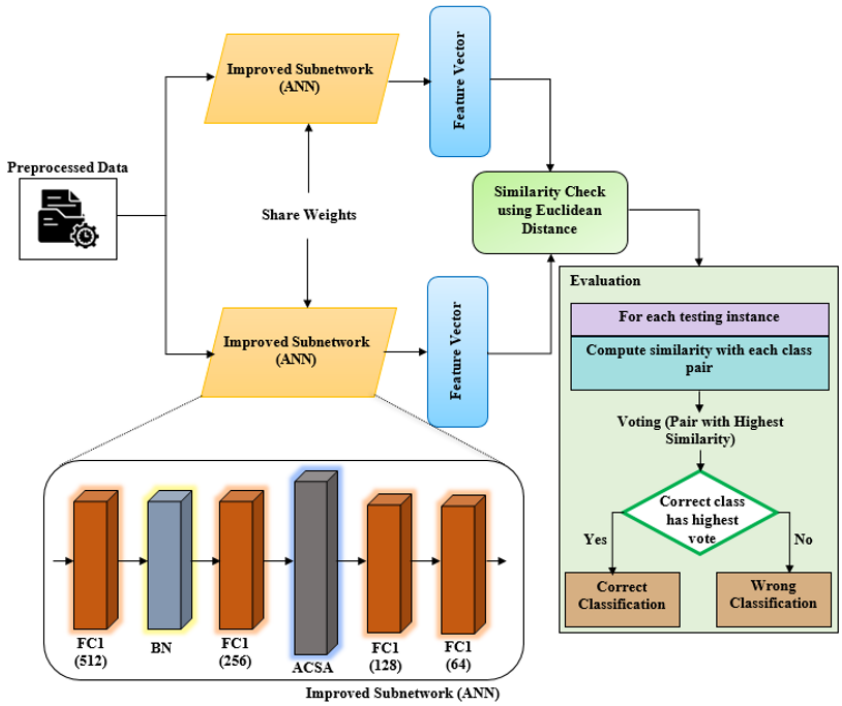


Figure 2. Layout of Modified Subnetwork-based Siamese Neural Network (MS-SNN) model

Improved Subnetwork: The improved subnetwork [26] first expands the input through a fully connected layer (FC1) with 512 units, followed by batch normalization (BN) to stabilize training, then compresses it through FC2 with 256 units. The resulting features pass through the ACSA module, which refines them by emphasizing the most important channels and spatial regions (Figure 3). This attention-enhanced output is then further compressed through FC3 with 128 units and FC4 with 64 units. This produces the final feature embedding used for similarity-based classification.

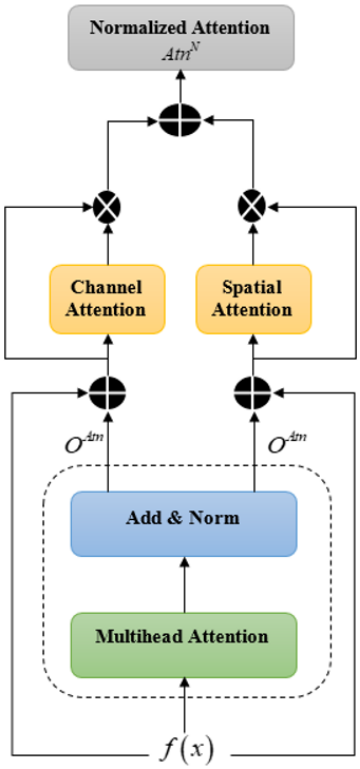


Figure 3. Representation of Adaptive Channel Spatial Attention (ACSA) module

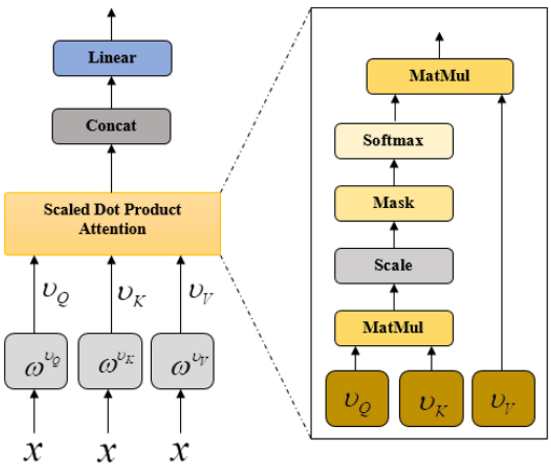


Figure 4. Depiction of multi-head attention

ACSA module: It takes the input feature map ($f(x)$), first passes it through a multi-head attention layer [27], which allows the model to simultaneously attend to different representation subspaces and capture relationships across feature dimensions. The multi-head attention (Figure 4) is computed using the following Eq. (4).

$$MH(\upsilon_Q, \upsilon_K, \upsilon_V) = Concat(head_1, head_2, \ldots, head_N).\omega^0 \quad (4)$$

where, $head_i = Atn(\upsilon_Q \omega_i^{\upsilon_Q}, \upsilon_K \omega_i^{\upsilon_K}, \upsilon_V {}_i^{\upsilon_V})$.

This output is then combined with the original input via a residual connection and normalized in an add-and-normalize step, which produces an intermediate representation labeled as O^{Atn}. This output (O^{Atn}) is then fed into two parallel branches [28], like a channel attention branch (Figure 5), which highlights the most important feature channels using average and max pooling followed by a shared MLP and sigmoid activation. The channel attention is computed using Eq. (5).

$$Ch_{Atn} = \sigma\left(Mlp\big(Avg.pool(f)\big) + Max.pool(f)\right) \quad (5)$$

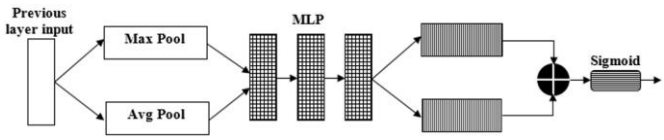


Figure 5. Structure of channel attention

The spatial attention branch highlights the most important spatial positions using a sigmoid over the channel-wise mean of O^{Atn}. The spatial attention is formulated using Eq. (6).

$$Sp_{Atn} = sigmoid(Atn_m) \quad (6)$$

Each branch's output is then multiplied element-wise with O^{Atn} to produce refined and attention-weighted feature maps, as expressed in Eq. (7) and Eq. (8).

$$Sp_{Atn} = O^{Atn} \otimes \omega_{Sp} \quad (7)$$

$$Ch_{Atn} = O^{Atn} \otimes \omega_{Ch} \quad (8)$$

These two weighted outputs are then fused through element-wise addition, as defined in Eq. (9) and the combined result is passed through a normalized attention function (Atn^N), which smoothly scales the fused signal using a tunable parameter to produce the final output of the module, as given in Eq. (10).

$$f(x) = Sp_{Atn} \oplus Ch_{Atn} \quad (9)$$

$$Atn^N(x) = 1 - \left(\frac{1 - f(x)}{1 + (exp(\delta) - 1).f(x)}\right) \quad (10)$$

where, δ denotes a scalable parameter such that $\delta \in (0,1)$.

Thereby, this structure refines the input features by emphasizing both the most informative channels and the most critical spatial regions, while the residual connections preserve the useful original information throughout the process. ACSA lies in the unified integration of multi-head, channel, and spatial attention within a residual-normalized pipeline tailored to IoT traffic embeddings.

3.3 Mitigation

Once the MS-SNN stage confirms that an attack is present, the framework moves into the mitigation stage, where an intelligent entropy-based adaptive mitigation strategy [29] is triggered to minimize the attack's impact by isolating

compromised devices, blocking malicious traffic, updating security policies and restoring normal network operations.

However, the standard mitigation strategy typically depends on fixed thresholds to decide when a node should be flagged as malicious. However, legitimate nodes may occasionally respond incorrectly, and the fixed thresholds cannot adapt to this; they can over-trigger, which means falsely flagging legitimate traffic bursts as attacks. This poor adaptability reduces the overall robustness of the mitigation system. To overcome this, the EATM strategy has been proposed and can be discussed in the following subsection.

3.3.1 Entropy with adaptive threshold-based mitigation strategy

Once an attack is confirmed by the detection stage, the EATM process works as follows:

For every active local IP address, the count of incoming packets sent through each edge switch is collected as $\chi = (z_1, z_2, \ldots, z_N)$, which represents the number of active local IP addresses receiving traffic through that edge switch.

From this packet-count vector, a probability (Pb_i) is computed for each IP address using Eq. (11).

$$Pb_i = \frac{\left[\frac{1}{\chi_i + \varepsilon}\right]}{\left[\sum_{i=1}^{N} \frac{1}{(\chi_i + \varepsilon)}\right]} \tag{11}$$

This converts raw packet counts into a normalized probability distribution across all destination IPs. Using these probabilities, the conventional information entropy of the edge switch is calculated with the standard information entropy formula as expressed in Eq. (12).

$$H_{S_j} = -\sum_{i=1}^{N\sum i} Pb_i log \tag{12}$$

Since conventional entropy alone is not sensitive enough to subtle shifts caused by attacks, the framework updates it into a modified entropy form that is expressed using Eq. (13).

$$H_{n-S_j} = -Y\left[\frac{1}{N-1}\right] \left[\sum_{i=1}^{N\sum i} (1 - Pb_i) log + Pb_i \, log(1 - Pb_i)\right] \tag{13}$$

where, $Y = \frac{1}{ln(2)}$ is a normalizing constant. This modified formula (H_{n-S_j}) accounts for both $H_n(S_j)$ and $H_a(S_j)$, which makes the entropy measure more responsive to distributional changes indicative of an attack. The modified entropy for switch (S_j) is then normalized using Eq. (14).

$$H'(S_j) = \frac{H_{n-S_j}}{log\, N} \tag{14}$$

So, the entropy values remain comparable across switches with different numbers of active IPs. Since some traffic features carry more discriminative information than others, each feature is assigned a weight w_i [30] based on the entropy weight method, as defined in Eq. (15).

$$w_i = \frac{1 - Et_i}{\sum_{i=1}^{n}(1 - Et_i)} \tag{15}$$

where, $Et_i = -\sum Pb_i \, log \, P \, b_i$.

The features with lower entropy receive higher weight. These weights are then combined with the modified entropy values across multiple features to compute the overall proposed weighted entropy for the switch, as defined in Eq. (16).

$$PWE(S_j) = \sum_{i=1}^{m} w_i * H_{n-S_j}[i] \tag{16}$$

Rather than using a fixed cut-off, the framework computes a dynamic adaptive threshold (ATh) using Eq. (17).

$$ATh = \psi . \sigma \tag{17}$$

where, ψ can be described as $\psi = 2 * \left(1 - \frac{1}{1+exp\left(-H_{n-S_j}\right)}\right)$ and σ is the standard deviation of recent entropy fluctuations that can be expressed as in Eq. (18).

$$\sigma = \sqrt{\frac{1}{Y}\sum_{i=1}^{Y}\left(H_{n-S_j}[i] - PWE(S_j)\right)^2} \tag{18}$$

This makes the threshold responsive to real-time traffic behavior instead of remaining static. For each switch (S_j), the deviation between the weighted entropy and its baseline entropy is compared against the adaptive threshold:

If $PWE(S_j) - H'(S_j) > ATh$ then confirmed as under attack and the corresponding node is removed from the network; else, the node is treated as benign and remains in the network.

The EATM lies in integrating entropy-based feature weighting with a dynamically adaptive threshold to enable data-driven mitigation under changing IoT traffic conditions. Thus, the EATM approach provides a data-driven way to identify the most relevant traffic characteristics for distinguishing malicious nodes correctly. Because the threshold adjusts automatically based on ψ and σ rather than staying fixed, the system reduces false positives during normal traffic bursts while remaining sensitive enough to detect evolving or subtle attack patterns. This makes the mitigation process more accurate, adaptive, and robust than conventional fixed-threshold methods.

4. RESULTS AND DISCUSSION

4.1 Simulation procedure

The proposed End-to-End DL-Driven Framework for IoT Botnet Attack Detection and Intelligent Mitigation was implemented using Python 3.7.6. The processor employed was "12th Gen Intel (R) Core (TM) i5-12450HX 2.40 GHz and the Installed RAM size was 12.0 GB (11.7 GB usable)". Further, the analysis was performed using the Aposemat IoT-23 Dataset [31].

The IoT-23 (Aposemat IoT-23) dataset is collected from IoT devices and is designed for IoT security research. It was captured in the Czech Republic at AIC Group, CTU, Stratosphere Laboratory, FEL. The dataset consists of 23 scenarios, including 20 network captures from IoT devices infected with real malware samples and 3 captures containing benign IoT device traffic. Traffic labeling was performed using the Zeek (Bro) conn.log format with labels including Benign, C&C, DDoS, PartOfAHorizontalPortScan, FileDownload, Attack and Okiru, among others.

The experimental analysis contains 6,044,652 network traffic instances categorized into five classes, namely Benign, PartOfAHorizontalPortScan, Okiru, DDoS and C&C. The class-wise distribution consists of 638,506 Benign samples, 3,389,036 PartOfAHorizontalPortScan samples, 1,313,012 Okiru samples, 688,812 DDoS samples and 15,286 C&C samples. The dataset was divided using a stratified random flow-level split, maintaining each attack category in the training and testing sets. This split does not represent capture-wise, device-wise, or attack-scenario-wise validation. The training of the MS-SNN model was carried out using a learning rate of 0.0001.

4.2 Analysis of IoT botnet attack detection performance

In this experimental study, the MS-SNN model is evaluated against DenseNet, AlexNet, Siamese Network, DCNN, EfficientNet, 1D-CNN [18], and ML+CDL+FDL [21] for detecting C&C, Okiru, PartOfAHorizontalPortScan, DDoS, and Benign traffic using 80% training data. Various evaluation metrics are used to assess the detection performance. As shown in Figure 6, the comparative accuracy results of the evaluated models are presented. Figure 7 presents the False Positive Rate (FPR) and False Negative Rate (FNR) results for the considered models. In comparison, MS-SNN achieves an accuracy of 98.3%, sensitivity of 95.6%, specificity of 98.9%, and precision of 95.5%, with an FPR of 0.011 and FNR of 0.044. The comparative results for Negative Predictive Value (NPV), F-measure, and MCC are presented in Figure 8.

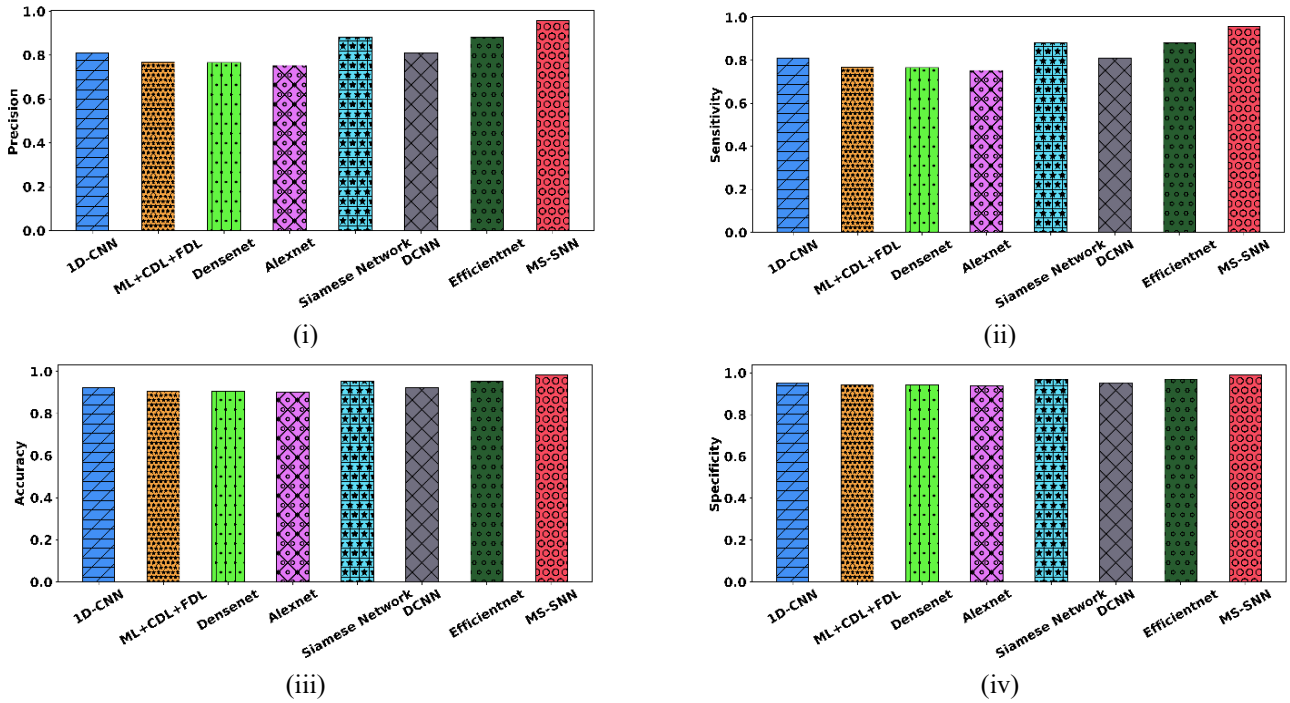


Figure 6. Comprehensive evaluation of Internet of Things (IoT) botnet attack detection performance of the Modified Subnetwork-based Siamese Neural Network (MS-SNN) Model "positive measures": (i) precision, (ii) sensitivity, (iii) accuracy, and (iv) specificity

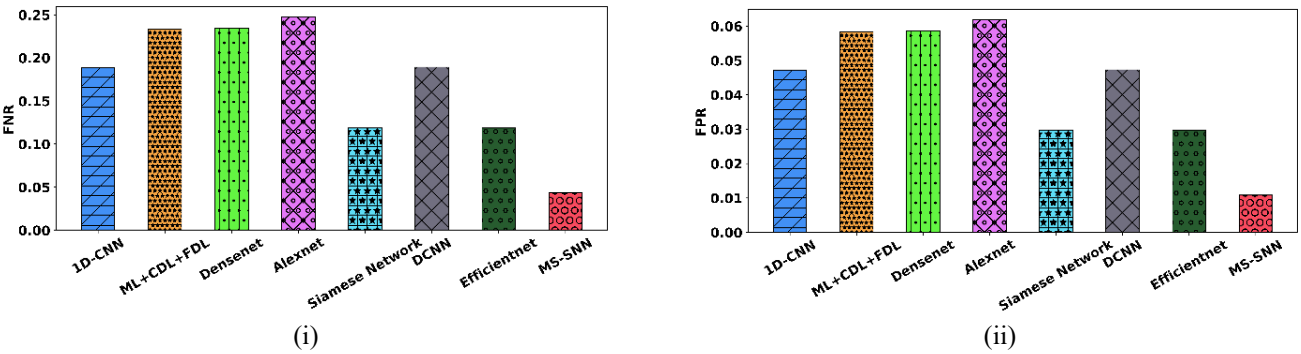


Figure 7. Comprehensive evaluation of Internet of Things (IoT) botnet attack detection performance of the Modified Subnetwork-based Siamese Neural Network (MS-SNN) model "negative metrics": (i) False Negative Rate (FNR) and (ii) False Positive Rate (FPR)

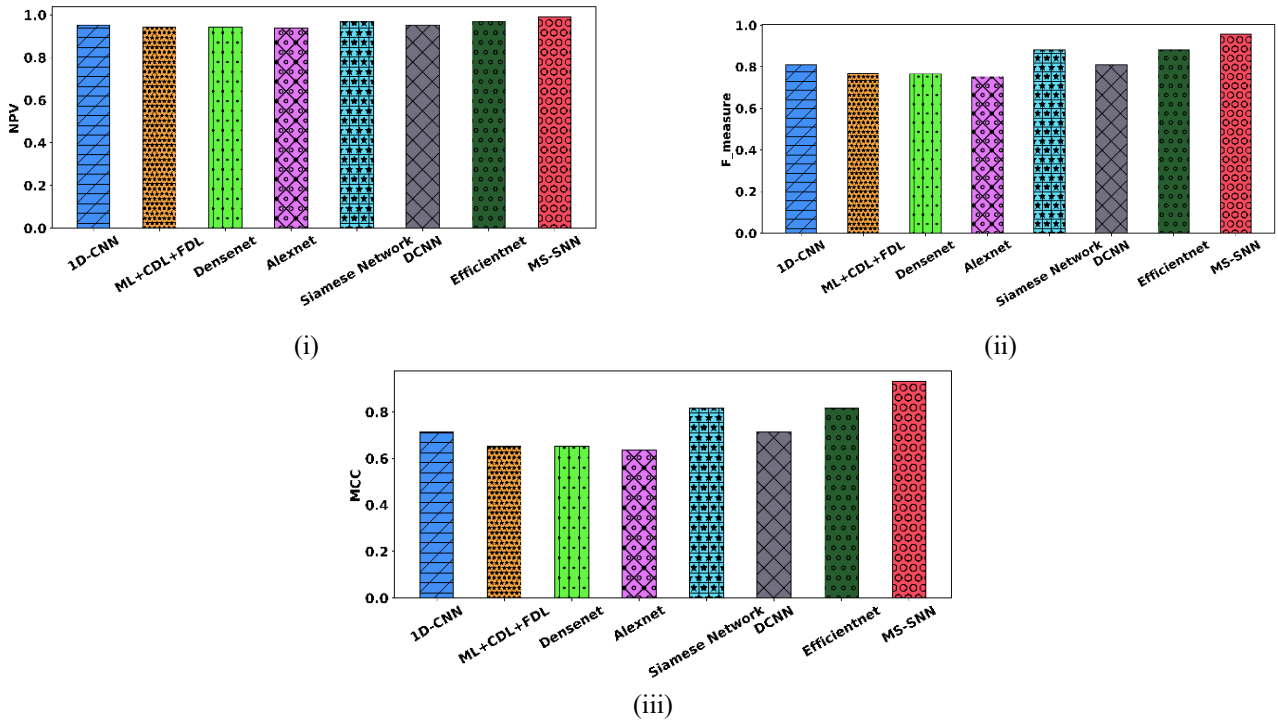


Figure 8. Comprehensive evaluation of Internet of Things (IoT) botnet attack detection performance of the Modified Subnetwork-based Siamese Neural Network (MS-SNN) Model: "other metrics" (i) Negative Predictive Value (NPV), (ii) F-measure, and (iii) Matthews Correlation Coefficient (MCC)

4.2.1 Ablation study

To examine the contribution of the MS-SNN framework towards achieving performance improvement, an ablation study is conducted under alternative configuration settings. Table 2 presents the ablation analysis of the MS-SNN using the 80% training data configuration. The proposed MS-SNN achieves the highest accuracy of 0.983, sensitivity of 0.956, specificity of 0.989, and MCC of 0.930 compared with the conventional Siamese and without-preprocessing configurations. It also achieves the lowest FPR (0.011) and FNR (0.044), demonstrating the effectiveness of the proposed approach.

Table 2. Ablation study on the MS-SNN approach using an 80% training data configuration

Metrics	Without Preprocessing	Conventional Siamese	MS-SNN
Accuracy	0.941	0.953	0.983
Sensitivity	0.851	0.881	0.956
Specificity	0.963	0.970	0.989
Precision	0.852	0.872	0.955
F-Measure	0.851	0.880	0.951
MCC	0.815	0.816	0.930
NPV	0.963	0.970	0.984
FPR	0.037	0.030	0.011
FNR	0.149	0.119	0.044

Note: MS-SNN: Modified Subnetwork-based Siamese Neural Network; MCC: Matthews Correlation Coefficient; NPV: Negative Predictive Value; FPR: False Positive Rate; FNR: False Negative Rate.

4.2.2 Class-wise metrics analysis

The class-wise precision, recall, and F1-score of the proposed MS-SNN and the comparison models across five classes are shown in Table 3. The proposed MS-SNN achieves higher precision and recall across all classes, with precision ranging from 0.888 to 0.957 and recall from 0.921 to 0.988. It also provides improved F1-scores for all classes compared with the other models, demonstrating its consistent class-wise performance. However, the F1-scores for Class 2 and Class 4 remain comparatively lower than the other classes, indicating scope for further improvement in these classes.

4.3 Statistical analysis

Table 4 presents the statistical accuracy analysis of the proposed MS-SNN and the comparison models. The proposed MS-SNN achieves the highest mean and median accuracy of 0.982, with a low standard deviation of 0.0006 and an accuracy range of 0.981–0.983, indicating consistent performance. In comparison, Siamese Network and EfficientNet achieve a mean accuracy of 0.950, while the other models obtain lower mean accuracies. These results demonstrate the stable and effective performance of the proposed MS-SNN model.

Robustness analysis:

The performance stability of the evaluated models is analyzed using Accuracy, Precision, Recall, and F-Measure over 10 independent runs. Each run uses a distinct stratified random 80:20 train-test split generated with a different random seed. The same 10 splits were applied to all models to ensure a fair comparison. Thus, the runs differ by random data partitions rather than repeated evaluation on the same test set. Table 5 presents the comparison of MS-SNN with DenseNet, AlexNet, Siamese Network, DCNN, EfficientNet, 1D-CNN [18], and ML+CDL+FDL [21]. MS-SNN achieves the highest mean Accuracy of 0.9817, Precision of 0.9549, Recall of 0.9536, and F-Measure of 0.9542, with low standard deviations, indicating consistent performance across the 10 runs. These results demonstrate the stability and reliability of MS-SNN for IoT botnet attack detection.

Table 3. Analysis of class-wise metrics

Class	1D-CNN	ML+CDL+FDL	Dense Net	Alex Net	Siamese Network	DCNN	Efficient Net	MS-SNN
class0 precision	0.765	0.712	0.712	0.696	0.852	0.766	0.853	0.945
class1 precision	0.783	0.732	0.732	0.718	0.865	0.783	0.865	0.949
class2 precision	0.774	0.722	0.722	0.707	0.858	0.774	0.859	0.947
class3 precision	0.608	0.543	0.542	0.525	0.729	0.608	0.729	0.888
class4 precision	0.815	0.770	0.771	0.758	0.882	0.813	0.884	0.957
class0 recall	0.696	0.637	0.636	0.620	0.798	0.696	0.799	0.921
class1 recall	0.944	0.928	0.927	0.922	0.967	0.944	0.967	0.988
class2 recall	0.815	0.771	0.770	0.757	0.884	0.815	0.884	0.958
class3 recall	0.875	0.842	0.842	0.832	0.923	0.874	0.923	0.973
class4 recall	0.786	0.738	0.737	0.723	0.864	0.785	0.864	0.949
class0 f1-score	0.813	0.771	0.769	0.758	0.884	0.814	0.884	0.957
class1 f1-score	0.800	0.754	0.753	0.740	0.874	0.799	0.874	0.953
class2 f1-score	0.060	0.046	0.046	0.043	0.101	0.061	0.101	0.253
class3 f1-score	0.782	0.728	0.736	0.722	0.863	0.792	0.869	0.951
class4 f1-score	0.112	0.087	0.087	0.081	0.181	0.113	0.181	0.400

Note: 1D-CNN: One-Dimensional Convolutional Neural Network; ML: Machine Learning; CDL: Centralized deep learning; FDL: Federated deep learning; DCNN: Deep Convolutional Neural Network; MS-SNN: Modified Subnetwork-based Siamese Neural Network.

Table 4. Accuracy-based statistical review of MS-SNN model vs. standard methods

Models	Mean	Median	Std-Dev	Min	Max
1D-CNN	0.919	0.920	0.0056	0.902	0.922
ML+CDL+FDL	0.903	0.903	0.0010	0.901	0.904
Densenet	0.904	0.903	0.0059	0.900	0.922
AlexNet	0.896	0.896	0.0016	0.894	0.899
Siamese Network	0.950	0.950	0.0013	0.949	0.953
DCNN	0.921	0.921	0.0012	0.918	0.922
EfficientNet	0.950	0.950	0.0013	0.948	0.953
MS-SNN	0.982	0.982	0.0006	0.981	0.983

Note: 1D-CNN: One-Dimensional Convolutional Neural Network; ML: Machine Learning; CDL: Centralized deep learning; FDL: Federated deep learning; DCNN: Deep Convolutional Neural Network; MS-SNN: Modified Subnetwork-based Siamese Neural Network.

Table 5. Analysis of mean and standard deviation across 10 independent runs

Models	Accuracy	Precision	Recall	F-Measure
1D-CNN	0.9188 Â ± 0.0056	0.7976 Â ± 0.0141	0.7966 Â ± 0.0140	0.7971 Â ± 0.0140
ML+CDL+FDL	0.9028 Â ± 0.0010	0.7576 Â ± 0.0026	0.7566 Â ± 0.0026	0.7571 Â ± 0.0026
DenseNet	0.9043 Â ± 0.0059	0.7615 Â ± 0.0147	0.7605 Â ± 0.0147	0.7610 Â ± 0.0147
AlexNet	0.8963 Â ± 0.0016	0.7415 Â ± 0.0039	0.7405 Â ± 0.0039	0.7410 Â ± 0.0039
Siamese Network	0.9502 Â ± 0.0013	0.8761 Â ± 0.0031	0.8749 Â ± 0.0031	0.8755 Â ± 0.0031
DCNN	0.9206 Â ± 0.0012	0.8023 Â ± 0.0030	0.8012 Â ± 0.0030	0.8018 Â ± 0.0030
EfficientNet	0.9503 Â ± 0.0013	0.8763 Â ± 0.0034	0.8751 Â ± 0.0034	0.8757 Â ± 0.0034
MS-SNN	0.9817 Â ± 0.0006	0.9549 Â ± 0.0015	0.9536 Â ± 0.0015	0.9542 Â ± 0.0015

Note: 1D-CNN: One-Dimensional Convolutional Neural Network; ML: Machine Learning; CDL: Centralized deep learning; FDL: Federated deep learning; DCNN: Deep Convolutional Neural Network; MS-SNN: Modified Subnetwork-based Siamese Neural Network.

Table 6. Analysis of statistical significance test

Proposed Model Vs	Friedman Chi-Square (Alpha=0.05)	*t*-test *p*-Value (Alpha=0.05)
1D-CNN	0.028	0.014
ML+CDL+FDL	0.004	0.002
DenseNet	0.020	0.010
AlexNet	0.039	0.020
Siamese Network	0.048	0.024
DCNN	0.002	0.001
EfficientNet	0.021	0.010

Note: 1D-CNN: One-Dimensional Convolutional Neural Network; ML: Machine Learning; CDL: Centralized deep learning; FDL: Federated deep learning; DCNN: Deep Convolutional Neural Network; MS-SNN: Modified Subnetwork-based Siamese Neural Network.

Table 7. Tukey HSD post hoc pairwise comparison results

Pairwise Comparison	Mean Difference	Adjusted *p*-Value	Significance
1D-CNN vs DCNN	0.0019	0.8986	Not significant
DenseNet vs ML+CDL+FDL	-0.0016	0.9594	Not significant
EfficientNet vs Siamese Network	-0.0001	1.0000	Not significant
All other Pairwise comparisons	–	<0.001	Significant

Note: HSD: Honestly Significant Difference; 1D-CNN: One-Dimensional Convolutional Neural Network; DCNN: Deep Convolutional Neural Network; ML: Machine Learning; CDL: Centralized deep learning; FDL: Federated deep learning.

4.3.1 Statistical significance test analysis

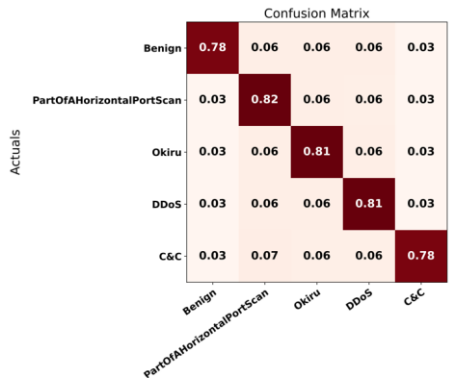
The statistical significance analysis of the proposed model against the comparison methods was conducted using the Friedman test and *t*-test at $\alpha = 0.05$, as shown in Table 6. All obtained *p*-values are below 0.05, indicating statistically significant differences between the proposed model and the evaluated methods. The results confirm the statistical significance of the proposed model's performance.

4.3.2 Tukey post hoc analysis

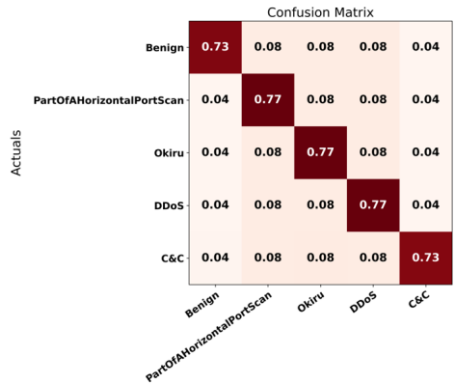
For the Tukey post hoc analysis, one-way ANOVA was first performed to evaluate the differences among the considered models, followed by Tukey's HSD post hoc test to identify specific pairwise differences while accounting for multiple comparisons. The results in Table 7 indicate that most model pairs exhibit statistically significant differences. However, the comparisons between 1D-CNN and DCNN, DenseNet and ML+CDL+FDL, and EfficientNet and Siamese Network were not statistically significant, with adjusted *p*-values of 0.8986, 0.9594, and 1.0000, respectively. All other pairwise comparisons were statistically significant, with adjusted *p*-values below 0.001. These results further demonstrate the statistically significant performance differences among the evaluated models.

4.3.3 Analysis of confusion matrix

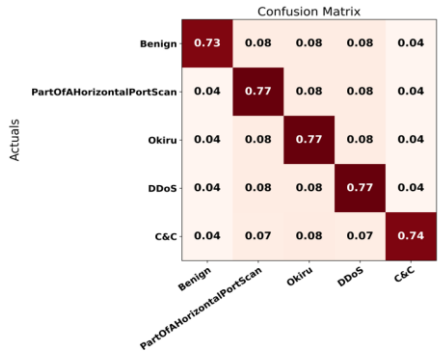
Figure 9 presents the confusion matrices of the evaluated models across the five traffic classes: Benign, PartOfAHorizontalPortScan, Okiru, DDoS, and C&C. The diagonal values indicate correctly classified samples, while the off-diagonal values represent misclassifications. MS-SNN achieves high correct-classification rates of 0.95–0.96 across the five classes. The major misclassifications occur between Okiru and PartOfAHorizontalPortScan, followed by Okiru-DDoS confusion.



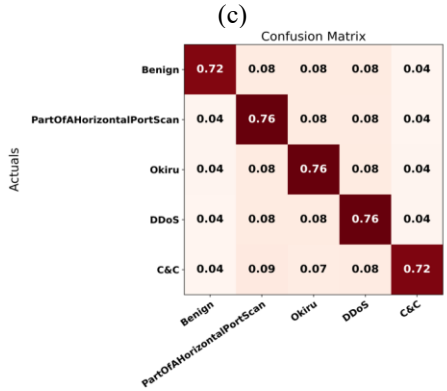
(a)



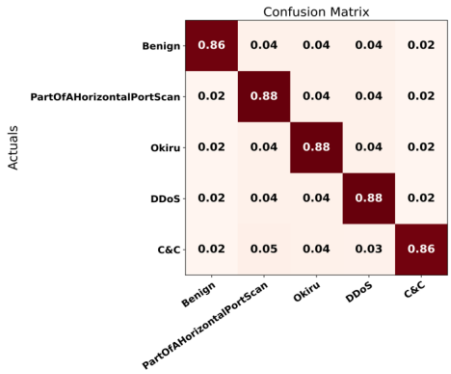
(b)



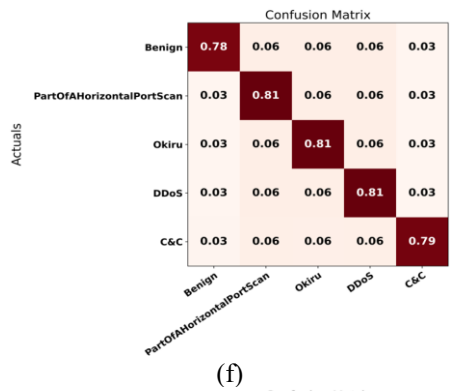
(c)



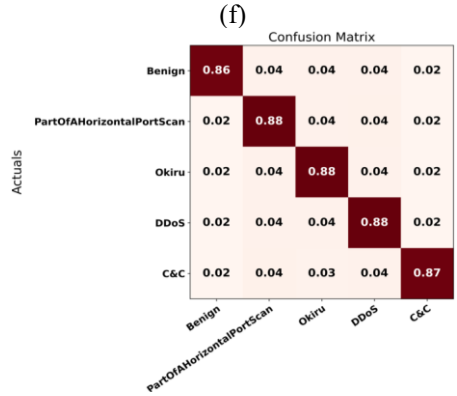
(d)



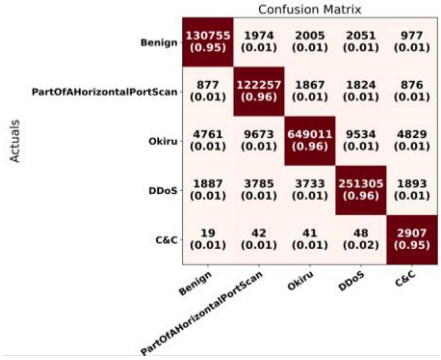
(e)



(f)



(g)



(h)

Figure 9. Confusion matrix analysis of (a) 1D-CNN, (b) ML+CDL+FDL, (c) DenseNet, (d) AlexNet, (e) Siamese Network, (f) DCNN, (g) EfficientNet, and (h) MS-SNN

Note: 1D-CNN: One-Dimensional Convolutional Neural Network; ML: Machine Learning; CDL: Centralized deep learning; FDL: Federated deep learning; DCNN: Deep Convolutional Neural Network; MS-SNN: Modified Subnetwork-based Siamese Neural Network.

The minority C&C class achieves approximately 95% correct detection despite its limited representation in the dataset. Compared with the baseline models, MS-SNN reduces C&C misclassification, indicating better handling of minority-class traffic. Overall, MS-SNN provides more consistent classification across the five traffic classes and reduces confusion between attack categories, demonstrating balanced detection with low misclassification rates and acceptable computational cost.

4.4 Analysis on learning rate

The learning rate analysis is conducted by evaluating the MS-SNN model under different learning rate configurations to assess the suitable settings for achieving the maximum detection effectiveness. The analysis determines how the learning rate affects the training efficiency and convergence behavior of the MS-SNN model, thereby providing insight into its capability to learn important patterns from the IoT network traffic and optimize its parameters effectively under different learning conditions. By analyzing the multiple learning rate values (0.1, 0.01, 0.01 and 0.0001), it expresses the classification performance using various perspectival views of measures in Table 8. Among the evaluated learning rates, the MS-SNN model achieved the highest performance at LR = 0.0001 by obtaining an accuracy of 0.983, specificity of 0.989, F-measure of 0.956, with the lowest FPR of 0.011 compared with other learning rate settings.

Table 8. Analysis of learning rate influence on model performance

Metric	Learning Rate			
	0.1	0.01	0.001	0.0001
MCC	0.766	0.892	0.817	0.930
Precision	0.846	0.917	0.882	0.956
FPR	0.038	0.051	0.030	0.011
F-measure	0.846	0.917	0.882	0.956
Sensitivity	0.846	0.917	0.882	0.956
Accuracy	0.938	0.942	0.953	0.983
FNR	0.154	0.083	0.118	0.044
Specificity	0.962	0.949	0.970	0.989
NPV	0.962	0.949	0.970	0.989

Note: MS-SNN: Modified Subnetwork-based Siamese Neural Network; LR: Learning Rate; MCC: Matthews Correlation Coefficient; FPR: False Positive Rate; FNR: False Negative Rate; NPV: Negative Predictive Value.

4.5 Confidence interval analysis of detection accuracy

Table 9 presents the 95% confidence interval-based accuracy evaluation of the proposed MS-SNN model over the different standard methods, namely DenseNet, AlexNet, Siamese Network, DCNN, EfficientNet, 1D-CNN [18] and ML+CDL+FDL [21].

The confidence intervals were obtained from the same 10 independent experimental runs. Each run used a distinct stratified random 80:20 train-test split generated using a different random seed, and the same set of 10 splits was applied consistently across all evaluated models. The proposed model achieves the highest accuracy interval, with a lower limit of 0.9812 and an upper limit of 0.9821, compared with the other evaluated models.

Table 9. Confidence interval-based accuracy (95%) evaluation of the MS-SNN model

Models	Lower Limit	Upper Limit
1D-CNN [18]	0.9145	0.9230
ML+CDL+FDL [21]	0.9020	0.9035
DenseNet	0.8999	0.9088
AlexNet	0.8952	0.8975
Siamese Network	0.9492	0.9511
DCNN	0.9197	0.9216
EfficientNet	0.9492	0.9513
PROPOSED	0.9812	0.9821

Note: 1D-CNN: One-Dimensional Convolutional Neural Network; ML: Machine Learning; CDL: Centralized deep learning; FDL: Federated deep learning; DCNN: Deep Convolutional Neural Network; MS-SNN: Modified Subnetwork-based Siamese Neural Network.

4.6 Macro-average results

Table 10 compares the performance of the evaluated models using five metrics. MS-SNN achieves the highest accuracy (0.96), macro precision (0.80), macro recall (0.95), macro F1-score (0.84), and weighted F1-score (0.96), outperforming the other models across all reported metrics.

Table 10. Macro-average performance comparison of the evaluated models

Model	Accuracy	Macro Precision	Macro Recall	Macro F1	Weighted F1
1D-CNN	0.81	0.63	0.80	0.65	0.83
ML+CDL+ FDL	0.77	0.59	0.75	0.61	0.79
DenseNet	0.77	0.59	0.76	0.61	0.79
AlexNet	0.75	0.58	0.74	0.60	0.77
Siamese Network	0.88	0.70	0.88	0.73	0.89
DCNN	0.81	0.63	0.80	0.65	0.83
EfficientNet	0.88	0.70	0.88	0.73	0.89
MS-SNN	0.96	0.80	0.95	0.84	0.96

Note: 1D-CNN: One-Dimensional Convolutional Neural Network; DCNN: Deep Convolutional Neural Network; MS-SNN: Modified Subnetwork-based Siamese Neural Network; F1: F1-score.

The area under the receiver operating characteristic curve (ROC-AUC) validation is conducted under this concept of IoT Botnet Attack Detection to assess the capability of classification models to differentiate between the attack traffic (C&C, Okiru, PartOfAHorizontalPortScan, DDoS) and benign network traffic over a range of decision thresholds. Here, the ROC curve illustrates the relationship between the attack detection rate and the false alarm rate, while the area under the

curve (AUC) value provides a single quantitative measure of the model's ability to distinguish between attack and normal traffic. From the review of the ROC-AUC graph (Figure 10), it is observed that the MS-SNN model depicts a maximal AUC score of 0.98, signifying stronger classification capability with minimal misclassification compared to other standard methods.

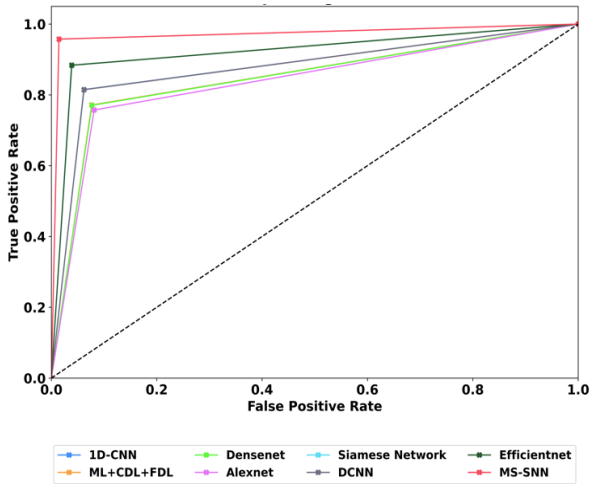


Figure 10. Analysis of attack detection capability using area under the receiver operating characteristic curve (ROC-AUC)

5. CONCLUSIONS

This paper proposed an end-to-end DL-driven framework for IoT botnet attack detection and intelligent mitigation that is structured around three integrated stages: preprocessing, detection and mitigation. In the preprocessing stage, network traffic was normalized using robust scaling to minimize the influence of outliers common in IoT environments. The detection stage employed an MS-SNN augmented with a new ACSA mechanism, which strengthened feature extraction by emphasizing discriminative feature channels and critical spatial regions, thereby producing more reliable similarity-based classification through Euclidean distance and voting-based evaluation. Upon attack detection, the mitigation stage applied an EATM technique that continuously adjusted to real-time traffic fluctuations. In conclusion, the experimental results confirm the effectiveness of the proposed framework, achieving detection accuracy of 98.3% with 80% training data while demonstrating superior mitigation robustness compared to conventional baseline methods.

REFERENCES

[1] Wasswa, H., Abbass, H.A., Lynar, T. (2026). VLSA-CL: A variational latent space alignment and contrastive learning framework for robust detection of IoT botnet attacks under concept drift. Neurocomputing, 696: 134098. https://doi.org/10.1016/j.neucom.2026.134098

[2] Wasswa, H., Abbass, H.A., Lynar, T. (2025). Latent space alignment for robust detection of IoT botnet attacks in non-stationary environments. Knowledge-Based Systems, 330: 114749. https://doi.org/10.1016/j.knosys.2025.114749

[3] Hossain, M.A., Islam, M.S. (2026). HTA-BotDef: A novel hierarchical tree aggregation for scalable and privacy preserving botnet attack detection with federated learning. Journal of Information and Intelligence, 4(3): 251-271. https://doi.org/10.1016/j.jiixd.2025.12.004

[4] Nasir, M.H., Arshad, J., Khan, M.M. (2023). Collaborative device-level botnet detection for internet of things. Computers & Security, 129: 103172. https://doi.org/10.1016/j.cose.2023.103172

[5] Alsubaei, F.S., Almazroi, A.A., Ayub, N. (2026). Federated ConvNeXt-swin temporal fusion network for malware and botnet detection in IoT systems. Scientific Reports, 16(1): 20147. https://doi.org/10.1038/s41598-026-55768-3

[6] Muñoz, D.C., Valiente, A.D.C. (2023). A novel botnet attack detection for IoT networks based on communication graphs. Cybersecurity, 6(1): 33. https://doi.org/10.1186/s42400-023-00169-6

[7] Ali, M., Mushtaq, M.F., Akram, U., et al. (2025). Botnet detection in internet of things using stacked ensemble learning model. Scientific Reports, 15(1): 21012. https://doi.org/10.1038/s41598-025-02008-9

[8] Asadi, M., Heidari, A., Jafari Navimipour, N. (2025). A new flow-based approach for enhancing botnet detection efficiency using convolutional neural networks and long short-term memory. Knowledge and Information Systems, 67(7): 6139-6170. https://doi.org/10.1007/s10115-025-02410-9

[9] Esmaeilyfard, R., Shoaei, Z., Javidan, R. (2026). GELAX: IoT botnet detection using dynamic graph pruning and anchored explainable AI. Cybersecurity, 9(1): 58. https://doi.org/10.1186/s42400-025-00440-y

[10] Alshehri, M.S., Ahmad, J., Almakdi, S., Al Qathrady, M., Ghadi, Y.Y., Buchanan, W.J. (2024). SkipGateNet: A lightweight CNN-LSTM hybrid model with learnable skip connections for efficient botnet attack detection in IoT. IEEE Access, 12: 35521-35538. https://doi.org/10.1109/ACCESS.2024.3371992

[11] Arnold, D., Gromov, M., Saniie, J. (2024). Network traffic visualization coupled with convolutional neural networks for enhanced IoT botnet detection. IEEE Access, 12: 73547-73560. https://doi.org/10.1109/ACCESS.2024.3404270

[12] Al-Shurbaji, T., Anbar, M., Manickam, S., et al. (2025). BoT-EnsIDS: Approach for detecting IoT botnet attacks leveraging bio-inspired based ensemble feature selection and hybrid deep learning model. Alexandria Engineering Journal, 129: 744-767. https://doi.org/10.1016/j.aej.2025.06.030

[13] Hossain, M.A., Saif, S., Islam, M.S. (2025). A novel federated learning approach for IoT botnet intrusion detection using SHAP-based knowledge distillation. Complex & Intelligent Systems, 11(10): 422. https://doi.org/10.1007/s40747-025-02001-9

[14] Korba, A.A., Diaf, A., Bouchiha, M.A., Ghamri-Doudane, Y. (2025). Mitigating IoT botnet attacks: An early-stage explainable network-based anomaly detection approach. Computer Communications, 241: 108270. https://doi.org/10.1016/j.comcom.2025.108270

[15] Maazalahi, M., Hosseini, S. (2025). A novel hybrid method using grey wolf algorithm and genetic algorithm for IoT botnet DDoS attacks detection. International Journal of Computational Intelligence Systems, 18(1): 61. https://doi.org/10.1007/s44196-025-00774-y

[16] Hossain, M.A., Islam, M.S. (2025). A novel feature selection-driven ensemble learning approach for accurate

botnet attack detection. Alexandria Engineering Journal, 118: 261-277. https://doi.org/10.1016/j.aej.2025.01.042
[17] Alshaeaa, H.Y., Ghadhban, Z.M. (2024). Developing a hybrid feature selection method to detect botnet attacks in IoT devices. Kuwait Journal of Science, 51(3): 100222. https://doi.org/10.1016/j.kjs.2024.100222
[18] Sureshkumar, A., Shanmugam, P., Mal, S., Malviya, L., Jadhav, A. (2026). Attention based CNN-transformer ensemble for multiclass botnet detection in industrial IoT networks. Discover Computing, 29(1): 198. https://doi.org/10.1007/s10791-026-09987-x
[19] Ali, S., Ghazal, R., Qadeer, N., et al. (2024). A novel approach of botnet detection using hybrid deep learning for enhancing security in IoT networks. Alexandria Engineering Journal, 103: 88-97. https://doi.org/10.1016/j.aej.2024.05.113
[20] Al-Fawa'reh, M., Abu-Khalaf, J., Szewczyk, P., Kang, J.J. (2024). MalBoT-DRL: Malware botnet detection using deep reinforcement learning in IoT networks. IEEE Internet of Things Journal, 11(6): 9610-9629. https://doi.org/10.1109/JIOT.2023.3324053
[21] Rasool, A.M., Safa, N.S., Mbarushimana, C. (2026). Towards privacy-preserving deep learning for intelligent IoT botnet detection. Applied Sciences, 16(3): 1665. https://doi.org/10.3390/app16031665
[22] Jin, H., Jeon, G., Choi, H.W.A., Jeon, S., Seo, J.T. (2024). A threat modeling framework for IoT-based botnet attacks. Heliyon, 10(20): e39192. https://doi.org/10.1016/j.heliyon.2024.e39192
[23] De Amorim, L.B., Cavalcanti, G.D., Cruz, R.M. (2023). The choice of scaling technique matters for classification performance. Applied Soft Computing, 133: 109924. https://doi.org/10.1016/j.asoc.2022.109924
[24] Hindy, H., Tachtatzis, C., Atkinson, R., Bayne, E., Bellekens, X. (2021). Developing a Siamese Network for intrusion detection systems. In Proceedings of the 1st Workshop on Machine Learning and Systems, United Kingdom, pp. 120-126. https://doi.org/10.1145/3437984.3458842
[25] Ahmed, S., Lee, Y., Hyun, S.H., Koo, I. (2018). Covert cyber assault detection in smart grid networks utilizing feature selection and euclidean distance-based machine learning. Applied Sciences, 8(5): 772. https://doi.org/10.3390/app8050772
[26] Chou, E.P., Hsieh, B.C. (2025). Enhancing anomaly detection in structured data using Siamese neural networks as a feature extractor. Mathematics, 13(7): 1090. https://doi.org/10.3390/math13071090
[27] Reza, S., Ferreira, M.C., Machado, J.J., Tavares, J.M.R. (2022). A multi-head attention-based transformer model for traffic flow forecasting with a comparative analysis to recurrent neural networks. Expert Systems with Applications, 202: 117275. https://doi.org/10.1016/j.eswa.2022.117275
[28] Woo, S., Park, J., Lee, J.Y., Kweon, I.S. (2018). CBAM: Convolutional block attention module. In European Conference on Computer Vision, Munich, Germany, pp. 3-19. https://doi.org/10.1007/978-3-030-01234-2_1
[29] Wang, R., Jia, Z., Ju, L. (2015). An entropy-based distributed DDoS detection mechanism in software-defined networking. In 2015 IEEE Trustcom/BigDataSE/ISPA, Helsinki, Finland, pp. 310-317. https://doi.org/10.1109/Trustcom.2015.389
[30] Chen, L., Wang, X., Wang, Y., Gao, P. (2023). Improved entropy weight methods and their comparisons in evaluating the high-quality development of Qinghai, China. Open Geosciences, 15(1): 20220570. https://doi.org/10.1515/geo-2022-0570
[31] Garcia, S., Parmisano, A., Erquiaga, M.J. (2020). IoT-23: A labeled dataset with malicious and benign IoT network traffic. https://doi.org/10.5281/zenodo.4743746

NOMENCLATURE

1D-CNN	One-Dimensional Convolutional Neural Network
CNN	Convolutional Neural Network
DCNN	Deep Convolutional Neural Network
LR	Learning Rate
GND	Generalized Normal Distribution
ANOVA	Analysis of Variance
HSD	Honestly Significant Difference
ROC-AUC	Area under the receiver operating characteristic curve
ACSA	Adaptive Channel-Spatial Attention
AIC	Artificial Intelligence Center
ANN	Artificial Neural Network
AUC	Area Under the Receiver Operating Characteristic Curve
BN	Batch Normalization
C&C	Command and Control
CDL	Centralized deep learning
CTU	Czech Technical University in Prague
DDoS	Distributed Denial-of-Service
DL	Deep learning
DRL	Deep Reinforcement Learning
EATM	Entropy with Adaptive Thresholding-based Mitigation
ET	Extra Trees
FC	Fully Connected
FDL	Federated deep learning
FEL	Faculty of Electrical Engineering
FNR	False Negative Rate
FPR	False Positive Rate
GW-GA	Grey Wolf and Genetic Algorithm
I3TM	IoT Targeting-Threat Modeling
IDS	Intrusion Detection Systems
IIoT	Industrial Internet of Things
IoT	Internet of Things
IQR	Interquartile Range
LSTM	Long Short-Term Memory
MalBoT-DRL	Malware Botnet with Deep Reinforcement Learning
MCC	Matthews Correlation Coefficient
ML	Machine Learning
ML+CDL+FDL	Machine Learning with Centralized deep learning and Federated deep learning
MLP	Multilayer Perceptron
MS-SNN	Modified Subnetwork-based Siamese Neural Network
NPV	Negative Predictive Value
RF	Random Forest
ROC	Receiver Operating Characteristic
TPR	True Positive Rate