Communications on Applied Nonlinear Analysis
ISSN: 1074-133X
Vol 32 No. 10s (2025)

Real-Time Mechanism for Mitigation and Investigation of IoT Botnet Attacks

Vikrant1,*, Dr. Gesu Thakur2

1. Research Scholar, College of Smart Computing, COER University, Roorkee, Uttarakhand, India

2. Professor, College of Smart Computing, COER University, Roorkee, Uttarakhand, India

Article History:

Received: 15-07-2025

Revised: 22-08-2025

Accepted: 10-09-2025

Abstract:

In this research, a better method for examining, identifying, and categorizing IoT botnet attacks using sophisticated deep learning models is presented. The potential of botnet-driven cyber threats has grown dramatically with the exponential expansion of IoT devices, making strong detection techniques necessary. The efficacy of Transformer, Deep Neural Networks (DNN), Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) models in identifying and categorizing harmful Internet of Things traffic is assessed in this study. A realistic depiction of IoT network abnormalities is ensured by using the N-BaIoT dataset for training and evaluation. The most efficient model is identified by a thorough performance study based on accuracy, precision, recall, F1-score, confusion matrix, and AUC. The most dependable method for IoT botnet detection, according to the results, is the GRU model, which performs better than other designs with 96.0% accuracy, 95.7% precision, 96.2% recall, and an AUC score of 0.989. Additionally, the study emphasizes how crucial sequential learning models are for detecting intricate assault patterns while lowering false positives and false negatives. The suggested method improves real-time IoT security and offers a scalable and effective way to mitigate botnet attacks. To further bolster IoT network defenses, future studies should investigate hybrid deep learning models, federated learning strategies, and real-time deployment. By enabling proactive threat identification and mitigation for next-generation IoT environments, our work advances intelligent cybersecurity solutions.

Keywords: IoT Security, Botnet Detection, Machine Learning, Anomaly Detection, Cybersecurity, Deep Learning, Network Traffic Analysis.

INTRODUCTION

By facilitating smooth communication between networks, sensors, and devices, the Internet of Things (IoT) has completely changed the digital landscape. The emergence of IoT botnets—networks of compromised IoT devices managed remotely by malevolent actors—is a result of the serious security flaws brought forth by this development. IoT botnets execute extensive cyberattacks, including DDoS attacks, data exfiltration, and network intrusion, by taking advantage of the processing power of compromised devices. IoT device security flaws include default credentials, insecure authentication methods, out-of-date firmware, and a lack of encryption are the main reasons why IoT botnets are created and propagate. Because of financial limitations and an emphasis on functionality above protection, many IoT devices are produced with little regard for security. Consequently, they become the focus of hackers who aim to use these weaknesses for nefarious ends. Finding vulnerable IoT devices online and using brute-force attacks or known flaws to take advantage of their lax security configurations is one of the main ways attackers create an IoT botnet. Once penetrated, a device is infected with malware, like Mirai, Gafgyt, or Mozi, which gives the attacker remote control. After then, these botnets connect to a central C&C server, which enables the attacker to plan extensive attacks on numerous devices at once [1].

Lack of regular firmware upgrades and security patches is another important aspect that contributes to the establishment of IoT botnets. Since many IoT devices still use antiquated software, they are vulnerable to known vulnerabilities that hackers can use to obtain unauthorized access. IoT manufacturers frequently overlook long-term software maintenance, which over time increases the number of security flaws. This contrasts with traditional computer devices, which receive regular security updates. Because IoT devices are widely used in a variety of areas, such as healthcare, smart homes, industrial automation, and critical infrastructure, where security lapses can have dire repercussions, this problem is made worse. The lack of established security protocols in IoT ecosystems also makes it more difficult to counteract botnet attacks because different manufacturers use varied security measures, which are frequently insufficient to stop widespread infections [2]. IoT botnets have grown rapidly due to the emergence of automated tools for creating botnets and underground markets for crimes. The cybercriminals may readily buy or rent botnet services, allowing even those with little technical know-how to carry out complex assaults. These botnets are frequently employed for commercial purposes, such as spamming campaigns, cryptocurrency mining, and ransomware assaults. IoT botnets have also been used by state-sponsored entities for espionage, cyberwarfare, and critical infrastructure disruption. IoT networks are fragmented and frequently unregulated, which makes it difficult to identify and successfully counteract these threats. Because infected devices constantly change their IP addresses, attackers commonly use fast-flux tactics, which make it challenging to identify and take down botnet infrastructures.

The threats associated with IoT botnets have increased due to the quick implementation of IoT technology in smart cities and intelligent transportation systems. For example, compromised security cameras, traffic control systems, and connected cars can all be used to interfere with city operations and possibly cause anarchy. IoT botnets represent a serious threat to medical devices like pacemakers and infusion pumps in the healthcare industry. If compromised, these devices could put human lives in jeopardy. Furthermore, power grids, water supply networks, and manufacturing processes can all be seriously disrupted by botnet-driven attacks that target smart grids and industrial control systems (ICS) [3]. Cybercriminals find the growing interconnectedness of IoT devices in critical infrastructure to be a profitable target, underscoring the urgent need for improved security measures. End users' inadequate usage of security features is one of the main reasons why IoT botnets are growing. Many people and businesses neglect to segment their networks, disable superfluous services, or change default passwords to prevent unwanted access. Attackers can readily take advantage of misconfigured devices and incorporate them into botnets because to this carelessness. Additionally, users' ignorance of cybersecurity best practices is a breeding ground for IoT botnets. Using strong passwords, turning on multi-factor authentication, and updating firmware are just a few of the fundamental security steps that many users neglect to do because they are ignorant of the dangers linked devices pose. Users frequently put convenience ahead of cybersecurity, which unintentionally aids in the growth of botnets, making the security environment even more complicated by the increasing number of IoT devices [4].

An IoT botnet is a sophisticated, multi-tiered network that is intended to take control of several compromised IoT devices and allow cybercriminals to conduct extensive cyberattacks. Depending on the goals of the attacker and the level of malware complexity, IoT botnets can have a hierarchical or P2P design. A C&C server, bot herders, hacked IoT devices like bots or zombies, and propagation techniques that aid in the spread of malware make up the most popular architecture. The C&C server, which serves as the botnet's main control center, is at the top of the hierarchy. It instructs compromised IoT devices to carry out a variety of nefarious tasks, including distributing spam, initiating DDoS attacks, stealing private data, and mining bitcoins. To avoid being discovered and taken down by cybersecurity researchers and law enforcement, these servers are frequently concealed

behind layers of anonymity, such as fast-flux DNS techniques or The Onion Router (Tor). The C&C infrastructure is used by the bot herder, or the attacker in charge of the botnet, to transmit commands to thousands or perhaps millions of compromised IoT devices across the globe [5]. The bots or zombie devices, which are compromised IoT devices including smart cameras, routers, medical equipment, and industrial sensors that have been infected with botnet software, are the next crucial element of an IoT botnet. These gadgets continue to be controlled by the attacker and carry out operations without the owners' knowledge. IoT devices are frequently targeted because they lack built-in security features, which makes it simple to compromise them using unpatched vulnerabilities, old firmware, and weak passwords. Many IoT devices run for years without firmware upgrades, which makes them appealing targets for thieves in contrast to traditional PCs or smartphones that receive frequent security updates [6].

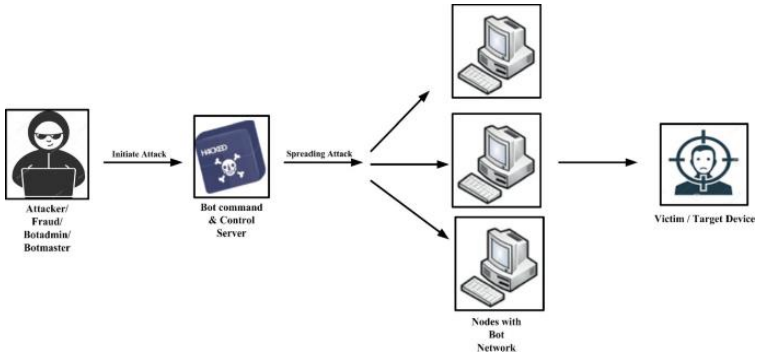

Fig.1. Structure of IoT botnet

The propagation mechanism, which allows malware to move from one device to another and expands the botnet's size and attack capabilities, is another crucial component of the IoT botnet structure. Propagation can take place via several techniques, including utilizing default credentials that users have never modified, brute-force attacks on weak authentication systems, and taking advantage of known flaws in IoT firmware. Self-propagating worms, which search the internet for susceptible devices and autonomously infect them without human assistance, are used by certain sophisticated IoT botnets. The Mirai botnet is a well-known illustration of this, as it quickly expanded by taking advantage of Internet of Things devices that had default users and passwords. An exponentially expanding network of compromised devices is produced once an infected device begins searching for additional susceptible IoT devices. Additionally, IoT botnets have communication channels that allow the compromised devices and the C&C server to connect. It is possible to create these channels of communication through hybrid, decentralized, or centralized architectures. The bot herder may more easily issue commands and manage the botnet in a centralized botnet architecture since all compromised IoT devices speak directly to the C&C server [7]. However, because law enforcement can knock down the C&C server and damage the entire network, centralized botnets are also more susceptible to takedown attempts.

To get around this, attackers have turned to decentralized botnet architectures, including P2P botnets, in which compromised devices talk to one another rather than depending on a single C&C server. Every hacked device in a P2P botnet function as a client and a server, transmitting commands and disseminating malware without the need for a central command center. Because there is not a single point of failure, detection and mitigation become much more difficult. Additionally, hybrid botnets—which combine aspects of decentralized and centralized architectures—have surfaced, providing attackers with increased adaptability and resistance to takedown attempts. Modern IoT botnets use evasion strategies in addition to these fundamental elements to evade security system detection. These methods include the use of domain generation algorithms (DGA) to dynamically generate new

command-and-control domains, polymorphic malware that continuously modifies its code to avoid detection by signatures, and encryption of communication routes. Certain botnets use fast-flux tactics, which cause the IP addresses under their control to shift quickly, making it challenging for security researchers to monitor and stop harmful activity [8]. Using trustworthy cloud services, including social media sites or encrypted messaging apps, as secret routes of communication to send commands to compromised devices is another method of evasion. Because of this, botnets can mix in with regular network traffic, making detection more difficult.

An IoT botnet's payload, which dictates the kind of assault it is designed to carry out, influences how well it works. IoT botnets are most frequently used to launch DDoS attacks, in which thousands or millions of compromised devices overload a target server with traffic, making it unavailable. Websites, banks, and even vital infrastructure can be rendered inoperable by these attacks. Certain botnets are made to withstand credential stuffing attacks, in which internet accounts are accessed without authorization using passwords and usernames that have been stolen. Others are designed for data exfiltration, which involves gathering and sending private data from compromised devices to the attacker. Cryptojacking, a technique where compromised IoT devices mine bitcoins without the owner's awareness while using electricity and computing capacity, is now being carried out by more sophisticated botnets. Global distribution is a major obstacle in the fight against IoT botnets since compromised IoT devices are frequently dispersed over several nations, making it challenging to coordinate international takedown operations. To put proactive security measures like network monitoring, threat intelligence sharing, and IoT security standards into practice, governments, cybersecurity companies, and law enforcement organizations must collaborate. Additionally, users can help avoid IoT botnet infestations by securing their devices, changing default credentials, updating firmware, and segmenting their networks to stop malware from spreading [9].

Many mitigation and analytic techniques have been developed in response to the rapid spread of IoT botnets with the goal of preventing, identifying, and disarming these threats. Network-based defenses, device-level security mechanisms, AI-driven detection models, and collaborative cybersecurity frameworks are classifications for these strategies. Analyzing traffic patterns to spot odd behaviors that might point to botnet activity is known as anomaly detection and network traffic monitoring, and it is one of the core tactics for preventing IoT botnets. Methods like DPI and flow-based anomaly detection are frequently employed to differentiate between malicious botnet-controlled communications and genuine traffic from IoT devices. By instantly examining signatures and irregularities, intrusion detection systems (IDS) like Snort and Suricata, as well as Security Information and Event Management (SIEM) systems, are essential for identifying botnet traffic. Furthermore, honeypots—deceptive security devices intended to draw in attackers—are frequently used to investigate the behavior, propagation strategies, and C&C procedures of IoT botnets. Platforms based on honeypots, such as IoTPOT and Honeyd, assist researchers in gathering information on botnets to enhance mitigation techniques. Device authentication and access control systems are another essential mitigating strategy [10]. One of the main causes of IoT devices being susceptible to botnet attacks is weak authentication. The danger of unwanted access can be considerably decreased by implementing public key infrastructure (PKI) techniques, multi-factor authentication (MFA), and strong password regulations. Before deploying IoT devices, users and manufacturers should also make sure that the default credentials are modified. To stop botnets from propagating throughout a network, network administrators also use firewall rules and network segmentation to restrict IoT device communication to only the endpoints that are required. Patch management and firmware updates are two of the best ways to lessen botnet infestations. Numerous Internet of Things devices have out-of-date firmware with unpatched vulnerabilities, which hackers take advantage of to infect them with malware. To guarantee that devices are safe from constantly

changing threats, device manufacturers must put vulnerability management systems and automated firmware updates into place. To distribute security patches to IoT devices without requiring user interaction, some businesses have started utilizing over-the-air (OTA) update protocols.

AI and ML models are becoming crucial for identifying and thwarting IoT botnet activity as botnets develop. Both supervised and unsupervised machine learning techniques are frequently employed to examine enormous volumes of network traffic data and spot anomalies. By categorizing traffic according to labeled datasets, supervised learning models—such as random forests, SVM, and deep learning-based CNN-LSTM models—have proven effective in botnet identification. Meanwhile, by spotting irregularities in real-time network traffic, unsupervised learning methods like autoencoders and clustering algorithms might be helpful in finding hitherto unidentified botnet variations. The goal of AI-driven behavioral analysis approaches is to keep an eye on the typical actions of Internet of Things devices and identify any unusual deviations that might point to a botnet infection. Some sophisticated AI systems also make use of federated learning, which improves security and privacy by enabling several IoT devices to work together to identify attack patterns without disclosing raw data. Blockchain technology has become a viable alternative to conventional methods for protecting IoT networks against botnets. Blockchain-based decentralized authentication methods do away with the need for central servers, which lowers the possibility of botnet-exploited single points of failure [11]. Smart contracts built on the blockchain can stop unwanted access, enforce security regulations, and confirm the integrity of devices. To ensure that only verified IoT devices can join a network, several academics have suggested employing Distributed Ledger Technology (DLT) for secure device identity management. Businesses may produce tamper-proof logs of device interactions by utilizing blockchain's immutable and decentralized characteristics, which makes it more difficult for attackers to control devices infected by botnets.

Malicious domains used by C&C servers can be sinkhole and blacklisted as another efficient method of mitigating botnets. Law enforcement authorities, ISPs, and cybersecurity groups keep a close eye on and ban IP addresses and domains linked to botnet activity. Security researchers can examine botnet behavior and interfere with its operations by diverting botnet traffic to sinkholes, which are specialized servers that mimic actual C&C servers but instead log botnet activity. For example, by destroying their C&C infrastructure, law enforcement organizations have effectively brought down a number of botnets, including Mirai and Avalanche. Security experts must create increasingly sophisticated tracking tools, though, because attackers are always evolving and employing DGA and fast-flux DNS tactics to avoid detection. Regulatory compliance and IoT security frameworks are two more crucial mitigation strategies. IoT security standards and best practices have been created by governments and regulatory agencies. Examples include the European Union's Cybersecurity Act, the NIST Cybersecurity Framework, and the IETF's IoT security guidelines. Manufacturers of IoT devices are required by these requirements to have built-in security features including hardware-based encryption, secure boot procedures, and tamper-proof chips. Furthermore, IoT security frameworks are increasingly implementing zero-trust architecture (ZTA), which reduces the attack surface by enforcing stringent authentication and least-privilege access requirements. Organizations may secure IoT installations using standardized methods thanks to security frameworks like the Industrial Internet Consortium's (IIC) security best practices and the IoT Security Foundation's (IoTSF) standards [12].

IoT botnet mitigation also heavily relies on collaborative and crowdsourced threat intelligence platforms. Initiatives to share threat intelligence enable enterprises, ISPs, and cybersecurity experts to exchange up-to-date information on new attack patterns, vulnerabilities, and botnet threats. Organizations can proactively protect against botnet activity by using real-time threat feeds from

platforms like VirusTotal, MITRE ATT&CK, and the CTA. Furthermore, enterprises can improve their security by using honeynet projects like The Honeynet Project, which offer insightful information about botnet attack pathways. Lastly, a crucial component of IoT botnet mitigation continues to be user education and awareness. Users frequently neglect to take basic security precautions like updating firmware, changing default passwords, and turning off unused network services, which leads to botnet infections. To inform consumers about IoT security best practices, organizations and cybersecurity communities need to fund awareness campaigns and training initiatives [13]. Users are empowered to adopt preventative security steps against botnet threats through initiatives like Cybersecurity Awareness Month, online security training programs, and IoT security certification courses.

Using a deep learning model, the current article's contribution is to process a network traffic dataset and categorize its criticality as either low-critical (benign) or high-critical (malicious). It starts by choosing dataset features that are pertinent to the classification objective, guaranteeing a significant and successful prediction. Based on these characteristics, the suggested model is trained on labeled data to differentiate between malicious and benign traffic, using its statistical abilities to spot trends and connections. Following training, the model is assessed on a testset, where each data point's criticality is determined by predictions. For security, the test data is subsequently subjected to an adaptive encryption mechanism. XOR encryption is utilized for low-critical (benign) communication, providing a quick and effective technique appropriate for non-critical data. On the other hand, a more robust encryption technique that resembles homomorphic encryption is used to high-critical (malicious) traffic. This process guarantees data confidentiality and permits computation on encrypted data without the need for decryption. To handle the computational constraints of running on a CPU without GPU acceleration, the encryption is performed in batches, dividing the test data into manageable chunks for sequential processing [14]. This batch processing approach balances efficiency and security, making the implementation practical for systems without specialized hardware acceleration. The entire workflow integrates classification and encryption to enable secure handling of network traffic while addressing the varying security needs of different data categories. The remaining paper can be categorized into the following sections: section II focuses on the latest research which has been conducted for the mitigation and analysis techniques of IoT botnet. Section III discussed the setup configuration and proposed methodology of the current article. Section IV described the N-BaIoT dataset which is publicly available and designed for real-time IoT network traffic data with 9 distinct commercial IoT devices. Section V discuss the outcome and results generated from the proposed model and compare the results with the existing research models. Section VII discussed the conclusion of the current article.

I. Literature Review

Three types of botnets can be distinguished: scattered, mixed, and central. Client-server (C/S) mode is used by central structures, whereas non-central P2P mode is used by dispersed structures. By combining dispersed and central structures, mixed structures improve the robustness and efficiency of communication. Controlling botnets requires these structures [15]. The paper offers a machine learning-based method for spotting botnet attacks, showcasing how well RF and SVM detect anomalies. The approach includes data pre-processing, feature extraction, and model training. Authors proposed a system which presents an efficient botnet detection system using network traffic flow analysis and machine learning, focusing on early detection and optimization using the Weka Toolbox and CTU13 dataset [16]. Researchers proposed a system that discusses the framework is having an enhanced machine learning framework for identifying botnet attacks on IoT, that will beat the default Decision tree and support vector machine models in terms of accuracy, precision, recall, and F1-score. It integrates the Bayesian optimization with Gaussian processes and decision tree

classification models. Its ability to discriminate between legitimate and malicious occurrences demonstrates its efficacy and resilience in Internet of Things settings [17].

Authors proposed a system that discusses about the ongoing development, botnets represent a danger to information security. The Signature-based and IDS-based solutions does not work in some cases. By using the DNS query data, a machine learning-based model for botnet identification is created. There are two stages to the model's implementation i.e. training and detection. Machine learning methods are used to extract and preprocess domain names during the training phase. DNS requests are tracked throughout the detection phase and categorized by the classifier. Researchers proposed a system that discusses about the CTU 13 dataset, the paper assesses machine learning techniques for identifying peer-to-peer botnets. It draws attention to the necessity of several traits and factors and emphasizes the potency of ensemble approaches [18]. Authors suggested a machine learning-based framework for detecting DNS-based botnet attacks. It uses information gain for feature selection and a genetic algorithm for hyper-parameter optimization. The TI-2016 DNS dataset evaluates the framework, reducing feature set size by up to 60% and achieving high detection accuracy, precision, recall, and F-score [19].

To combat cybersecurity risks, studies present a strong early-stage botnet detection technique that focuses on phishing, malware distribution, DDoS attacks, and identity theft. After choosing the best features, the method feeds them into machine learning classifiers such as Random Forest, SVM, Logistic Regression, and Multilayer Perceptron using PCA and IG techniques [20]. A multilayer framework for botnet identification using machine learning techniques is proposed by researchers [21]. It achieves over 90% accuracy and a false negative rate of less than 2.5%. It emphasizes that more testing is necessary and that problems with clustering decentralized botnets and creating a dynamic framework should be the focus of future studies. The four classifiers proposed by authors, Naïve Bayes, K-Nearest Neighbor, Support Vector Machine, and Decision Trees and discuss the application of machine learning for botnet attack detection. With 99.89% accuracy, precision, recall, and F-score, the Decision Trees model performs better than any other [22].

A system that addresses the use of machine learning for botnet attack detection in IoT networks is proposed by authors. The system focuses on timely intrusion detection because of large-scale attacks. To maximize performance, it employs filtering and wrapping techniques. According to the study, hybrid feature selection that combines wrapper and Fisher's score approaches reduce computational requirements while achieving high detection rates [23]. A solution that addresses the growing issue of cyberattacks targeting IoT devices and the necessity of efficient detection systems is proposed by a study [24]. The increase in botnet assaults against IoT devices and the significance of network intrusion detection as a defense strategy are covered first. The authors suggest a detection architecture to overcome these issues and draw attention to the shortcomings of current detection systems, especially those that rely on attack signatures. It employs DNS traffic analysis. By successfully separating botnet activity from normal traffic, the anomaly-based method improves network security against malevolent bot attacks. In their investigation of machine learning techniques for botnet detection, Sean Miller and Curtis Busby-Earle point out the advantages and disadvantages of each approach, as well as how well they detect botnet activity and the possibilities of clustering algorithms [25].

This article examines the notorious Mirai botnet, emphasizing its methods of propagation, exploitation of IoT vulnerabilities, and DDoS attack mechanisms [26]. The authors suggest a machine learning-based detection methodology to find traffic irregularities indicative of Mirai infestations. This work also investigates the use of deep learning methods for IoT botnet identification. It demonstrates the promise of deep learning in cybersecurity by recognizing botnet activities with high accuracy using CNN on network traffic data. This study suggests a lightweight anomaly detection system for IoT devices with limited resources. It detects botnet activity with

minimal CPU overhead by combining statistical models and feature reduction techniques. This study also examines how IoT botnets interact with compromised devices by breaking down their C&C infrastructure. Disrupting C&C channels is recommended by the authors as a successful mitigating technique [27].

The study, Behavioral Analysis of IoT Malware [28], focused on the ways in which botnets carry out assaults and engage with their host environment. The authors suggest using a signature-based method to pinpoint the distinctive characteristics of IoT malware. This study compares supervised and unsupervised methods to assess several machine learning algorithms for IoT botnet identification. K-means clustering and random forests are useful methods for examining abnormalities in network traffic. To improve IoT security against botnet attacks, the authors suggest incorporating blockchain technology. Blockchain reduces botnet spread and stops unwanted access by decentralizing power and guaranteeing data integrity. This study investigates the application of SDN for real-time IoT botnet mitigation. Dynamic traffic filtering and quick reaction to anomalies caused by botnets are made possible by SDN's centralized control plane. This study presents a federated learning method for cooperatively training detection models across dispersed IoT devices. This technique improves the accuracy of botnet identification while protecting data privacy [29]. The authors simulate various scenarios to assess the effectiveness of containment strategies (Table I).

Table I. Literature Review on IoT botnet

Reference	Contribution	Method	Outcome	Research Gap
Understanding Mirai: The IoT Botnet [13]	Analyzes Mirai botnet's propagation and vulnerabilities	Examines default credential exploitation	Highlights need for stronger authentication	Limited focus on post-attack recovery
Detection of IoT Botnets Using Machine Learning [14]	Employs ML models for detecting botnets	Random Forest, SVM, feature selection	Effective real-time detection pipeline	Lacks scalability to large-scale networks
Botnet Command and Control Mechanisms in IoT [15]	Investigates botnet C2 strategies	Analyzes centralized, decentralized models	Insights into disrupting C2 communications	Minimal exploration of hybrid C2 models
Survey on IoT Botnet Taxonomy and Countermeasures [16]	Categorizes botnets and reviews defenses	Literature survey	Comprehensive taxonomy	Does not evaluate countermeasures in practice
Lightweight Encryption for IoT Botnet Prevention [17]	Proposes lightweight cryptography for IoT	Implements lightweight encryption algorithms	Enhanced IoT security against botnets	No discussion on algorithm adaptability
IoT Botnet Detection Using Deep Learning	Leverages deep learning for botnet	LSTMs, CNNs	High detection accuracy	Limited to supervised learning

[18]	detection			scenarios
Botnet Evolution and IoT Vulnerabilities [19]	Examines botnet evolution and IoT weaknesses	Historical analysis	Identifies evolving attack strategies	No predictive modeling of future trends
Anomaly-Based Detection of IoT Botnets [20]	Proposes anomaly-based detection framework	Statistical models	Effective in identifying novel attacks	High false positive rates
IoT Botnet Simulation and Attack Impact Analysis [21]	Simulates botnet attacks	Controlled experiments	Metrics for assessing attack severity	Limited to small-scale scenarios
Collaborative Defense Mechanisms Against IoT Botnets [22]	Discusses collaborative defense strategies	Threat intelligence sharing	Promotes collective mitigation efforts	No practical implementation examples
Blockchain-Based IoT Botnet Mitigation [23]	Explores blockchain for securing IoT networks	Decentralized authentication	Enhances trust management and security	Scalability and energy consumption concerns
Energy-Efficient IoT Botnet Detection [24]	Optimizes detection algorithms for energy efficiency	Energy-aware optimization methods	Reduces energy footprint	Trade-off with detection accuracy
IoT Botnet Forensics and Traceability [25]	Focuses on forensic methods for botnet analysis	Logging and evidence collection	Improved traceability of botnet origins	Limited automation in forensic processes
Impact of IoT Botnets on Cloud Services [26]	Investigates IoT botnets' impact on cloud	Case studies and simulations	Highlights interdependence of IoT and cloud	Narrow scope of cloud service types studied
Real-Time IoT Botnet Detection Framework [27]	Proposes real-time detection framework	Edge computing, ML	Feasibility of instant botnet threat mitigation	No evaluation of latency in real-world deployment

II. Proposed Methodology

An approach for the analysis and mitigation of IoT botnet assaults was presented in the current part. The suggested methodology is divided into discrete sections, including findings & analysis, deep learning model, and experimental setting design. The suggested model with all its sub-modules is shown in Figure 3.

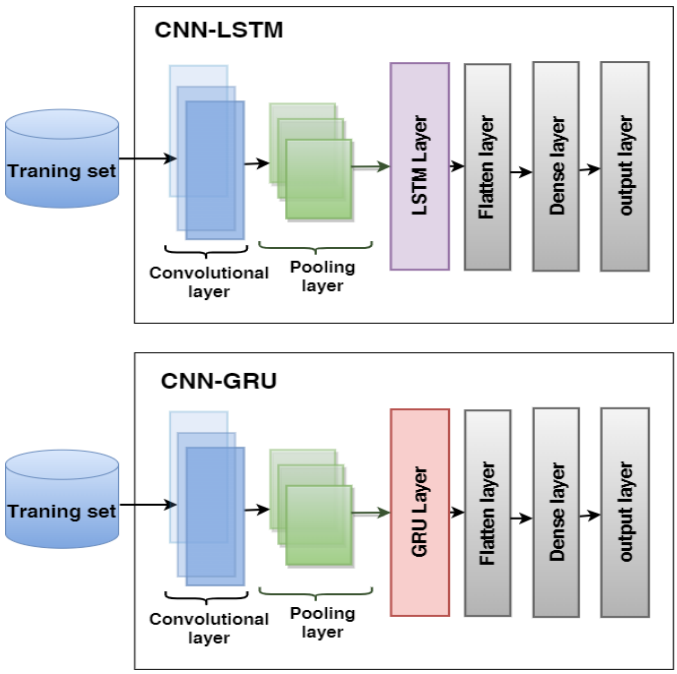

Fig. 2. Proposed Methodology

3.1 Experimental Setup Configuration

The N-BaIoT dataset's experimental setup configuration was created to record, in real time, both benign and botnet-infected network traffic from nine commercial IoT devices under carefully monitored circumstances. These gadgets, which were linked to a local network to replicate actual IoT surroundings, included motion sensors, doorbells, smart cameras, and baby monitors. At first, the gadgets were allowed to function under normal user interactions, which produced normal traffic. Mirai and Bashlite botnets were purposefully installed on the devices, launching a variety of assaults such network scanning, HTTP-based attacks, and TCP/UDP floods. Wireshark and tcpdump were used to record the network traffic, and CICFlowMeter was utilized to extract 115 network flow features. Flow-based characteristics, such as TCP flag distributions, inter-arrival delays, and packet length statistics, were the main emphasis of the data gathering. Researchers were able to efficiently train and assess machine learning-based detection models since the setup guaranteed a fair distribution of data. In order to differentiate between botnet and benign traffic, the dataset was tagged and preprocessed to eliminate duplicate entries. The high-fidelity depiction of IoT network behaviors offered by the experimental configuration made it possible to accurately detect anomalies, prevent intrusions, and analyze threats in real time to protect IoT ecosystems from botnet attacks.

3.2 Deep Learning Algorithms

For the classification of IoT network, we have utilized distinct deep learning algorithms such as CNN, LSTM, GRU, and DNN. A CNN is a deep learning architecture designed for processing structured grid data, such as images and time-series data. It is particularly effective for feature extraction and pattern recognition, making them suitable for IoT botnet attack detection by analyzing network traffic patterns. The convolution operation extracts the meaning features using the kernels.

$$Z(i, j) = m \sum n \sum X(i + m, j + n)\, W(m, n) + b \qquad (1)$$

Where X (i, j) is the input feature map, W (m, n) is the kernel, b is the bias term, and Z (i, j) is the resulting feature map. The activation function ReLU prevents the gradient issues and allow deeper networks for training effectively.

$$f(x) = \max(0, x) \qquad (2)$$

The pooling layer reduces the dimensions while preserving dominant features. The generic max pooling feature layer can be defined in terms of equation 3 that choose the maximum value in the selected window.

$$P(i,j) = Max_{m,n} Z(i + m, j + n) \qquad (3)$$

LSTM is a type of recurrent neural network (RNN) architecture designed to handle sequential data by overcoming the vanishing gradient problem. This can be achieved by introducing a memory cell and three gating mechanisms: the forget gate, input gate, and output gate. The LSTM unit is having the following components such as cell state (CtC_tCt) for storing the long-term dependencies, forget gate (ftf_tft) used to determines which past information to discard, input gate (iti_tit) decides which new information to add, and output gate (oto_tot) used to regulate the output of the cell.

$$ft=\sigma\,(W_f \cdot [h_{t-1}, x_t]+b_f) \qquad (4)$$

Where W_f is the weight matrix for the forget gate, h_{t-1} is the hidden state from the previous time step, x_t is the current input, b_f is the bias term, and σ is the sigmoid activation function. If f_t is close to 1, past information is retained; if close to 0, it is discarded. The input gate determines how much of the new input should be added to the cell state:

$$i_t=\sigma\,(W_i \cdot [h_{t-1}, x_t] + b_i) \qquad (5)$$

The output gate determines what the next hidden state should be, which is used for future time steps:

$$o_t=\sigma\,(W_o \cdot [h_{t-1}, x_t] + b_o) \qquad (6)$$

The number of layers, output shapes, parameters, and connection are described in the table I and Figure 3.

Table I. Parameters and Layer Description

Layer (Type)	Output Shape	Parameter	Connected To
input_layer (InputLayer)	(None, 39, 1)	0	-
multi_head_attenti… (MultiHeadAttentio…)	(None, 39, 1)	1345	input_layer[0][0… input_layer[0][0]
dropout_1 (Dropout)	(None, 39, 1)	0	multi_head_atten…
layer_normalization (LayerNormalizatio…)	(None, 39, 1)	2	dropout_1[0][0]
add (Add)	(None, 39, 1)	0	layer_normalizat… input_layer[0][0]
sequential (Sequential)	(None, 39, 60)	95560	add[0][0]

dropout_2 (Dropout)	(None, 39, 60)	0	sequential[0][0]
conv1d_18 (Conv1D)	(None, 39, 1)	61	dropout_2[0][0]
layer_normalizatio… (LayerNormalizatio…)	(None, 39, 1)	2	conv1d_18[0][0]
add_1 (Add)	(None, 39, 1)	0	layer_normalizat… add [0][0]
global_average_poo… (GlobalAveragePool…)	(None, 39)	0	add_1[0][0]
dense (Dense)	(None, 32)	1280	global_average_p…
dropout_3 (Dropout)	(None, 32)	0	dense[0][0]
dense_1 (Dense)	(None, 21)	693	dropout_3[0][0]

Total params: 99,343 (388.06 KB)

Trainable params: 98,983 (386.65 KB)

Non-trainable params: 360 (1.41 KB)

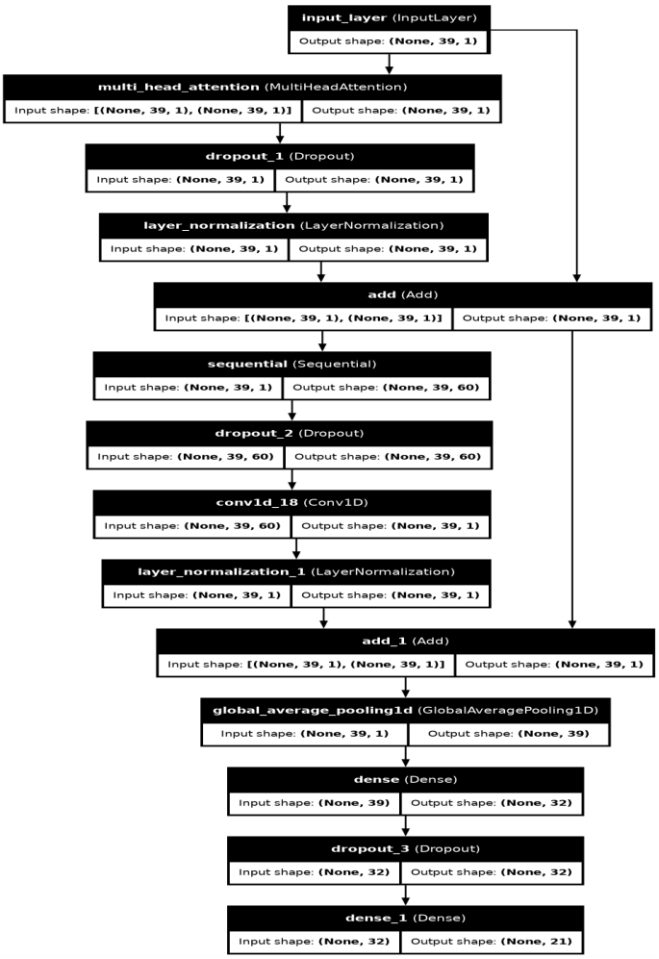

Fig.3. Training Parameters

III. Dataset Description

The Network-based Behavioral IoT (N-BaIoT) dataset is a benchmark dataset designed for detecting botnet attacks on IoT devices using network traffic analysis [30]. This dataset was built by collecting real-time network traffic from various IoT devices infected with botnet malware. The primary goal of the N-BaIoT dataset is to provide a diverse, labeled dataset for researchers to develop and evaluate machine learning and deep learning models for IoT botnet attack detection, mitigation, and analysis. The deataset consists of benign traffic generated from normal IoT device usage and malicious traffic captured after the devices were compromised by different botnet families, such as Mirai and Bashlite. Nine commercial IoT devices were deployed in a controlled network environment as part of the experimental setup configuration used to gather the N-BaIoT dataset. These Internet of Things gadgets include a Samsung SNH-1011 smart camera, a Philips B120N/10 baby monitor, two doorbells (Danmini and Ennio), a SimpleHome XCS7-1003-WHT webcam, a SimpleHome XCS7-1002-WHT motion sensor, and three security cameras (Ecobee and Panasonic). Every device was linked to a local network, and benign traffic was created by simulating typical user behaviors. Then, two of the most well-known botnets that target IoT devices, Mirai and Bashlite, were purposefully installed on the devices. To guarantee controlled execution, the malware infections were carried out under close observation. Infected devices were seen in a variety of attack scenarios, including reconnaissance scanning, HTTP-based attacks, and TCP and UDP floods.

Network sniffing tools like Wireshark and TCPdump were used to capture the dataset, gathering packet-level and flow-based information from the traffic data. Each file in the dataset's structure corresponds to a distinct IoT device and contains both regular and malicious communication. CICFlowMeter, an open-source application for creating bidirectional network flow records, was used to extract statistical network flow features from the dataset rather than raw packet data. Among these features are TCP flag counts, flow byte rates, packet length distribution, flow duration, and inter-arrival time. Basic flow features, content-based features, time-based features, and statistical aggregation features are the different categories into which the dataset's 115 features can be divided. Attack and regular traffic are distributed evenly throughout the N-BaIoT dataset, facilitating efficient model training and assessment. Researchers can evaluate the generalization capacity of their models thanks to the dataset's distinct training and testing splits. The dataset has been extensively utilized for botnet identification based on machine learning, using classifiers including CNN, LSTM, SVM, DNN, and RF. Research on real-time intrusion detection systems, adversarial attack resistance, and anomaly detection methods has also been made easier by the dataset. An extensive, labeled dataset for detecting IoT botnet attacks, the N-BaIoT dataset provides a structured set of network flow characteristics from both malicious and benign traffic. The dataset continues to be a useful tool for cybersecurity researchers developing intelligent threat detection systems for IoT networks because of its wide range of IoT devices, attack scenarios, and well-defined feature sets. The characteristics in the N-BaIoT dataset are described with feature rank in Table II.

Table II. N-BaIoT Dataset Features

Feature Rank	Unique Values	Feature Name	Remark
1	Multiple	Flow Duration	Total time a flow remains active. and measured in microseconds.
2	Multiple	Total Bwd Packets	Indicates the number of packets received in the backward direction and useful in identifying request-response patterns.
3	Multiple	Total Fwd Packets	Indicates the number of packets sent in the forward direction and it helps to detect anomalies in

			communication patterns.
4	Multiple	Total Length of Bwd Packets	Sum of all packet sizes in the backward direction and helps in analyzing response behavior.
5	Multiple	Total Length of Fwd Packets	Sum of all packet sizes in the forward direction and indicates data transfer volume from sender to receiver.
6	Multiple	Flow Bytes/s	It indicates average number of bytes per second in the flow and useful for detecting bandwidth anomalies.
7	Multiple	Fwd Packet Length Mean	Average packet size in the forward direction and helps to detect unusual communication patterns.
8	Multiple	Flow IAT Mean	Average time interval between two packets in a flow and useful for detecting traffic delays and scanning attacks.
9	Multiple	Min Packet Length	Indicates the smallest packet size observed in the flow and can reveal probing or scanning attempts.
10	Multiple	Packet Length Variance	Measures variation in packet sizes within a flow and high variance can indicate mixed traffic behavior.
11	Multiple	Bwd Packets/s	Rate of packets received per second and sudden spikes indicate potential botnet activity.
12	Multiple	Fwd Push Flags	Count of TCP PUSH flags in forward direction and often associated with brute-force attacks.

To examine IoT botnet activity, 115 features were taken from network traffic statistics to create the N-BaIoT dataset. These properties are divided into basic flow features, time-based attributes, statistical measures, and TCP flag distributions. They are obtained from both packet-level and flow-based characteristics. Numerous numerical features are included in the dataset, including flow duration, inter-arrival intervals (IAT), byte/packet transfer rates, and packet length statistics (mean, min, max, and variance). By examining changes in traffic patterns, these metrics aid in the detection of botnet activity. Protocol-related characteristics, such as the quantity of TCP flags (SYN, ACK, PUSH, RST, and FIN), are also included in the dataset. These properties are crucial for spotting fraudulent traffic patterns like DDoS floods or scanning attempts. The structure of the dataset affects how many distinct values each feature has. While some aspects, like packet sizes and flow durations, contain a wide range of numerical values, others, like protocol type and TCP flag counts, have a small set of discrete values. The N-BaIoT dataset offers a thorough understanding of IoT network behavior by leveraging 115 different variables, which makes it a crucial tool for creating machine learning-based botnet detection algorithms. IDS and anomaly detection frameworks may be effectively trained and evaluated thanks to its large feature set and labeled attack scenarios, which guarantees improved security for IoT environments.

IV. Results and Discussion

The article's current section compares with previous research and presents the unique findings that were produced following the model analysis. The class distribution of the N-BaIoT dataset is shown in a pie-chart in Figure 4, which shows the percentage of various traffic categories, such as benign traffic and various botnet attack types. Each slice of the pie chart corresponds to a specific class, with the percentage values indicating the frequency of each type within the dataset. The chart includes botnet infections caused by Mirai and Gafgyt malware families, both of which are well-known for exploiting IoT devices for DDoS attacks. The benign traffic class is present in the dataset, indicating normal device communication, which serves as a baseline for attack detection. Distinct Mirai attack types such as mirai_udpplain, mirai_udp, mirai_syn, mirai_scan, and mirai_ack are represented, each

corresponding to different methods used by the botnet to disrupt network operations. For example, Mirai SYN and Mirai UDP attacks target TCP SYN flooding and UDP-based flooding, respectively, which are common in large-scale DDoS attacks. Similarly, Mirai scan represents malicious scanning activities where the botnet searches for vulnerable IoT devices. The Gafgyt malware family is represented with multiple attack variations, including gafgyt_tcp, gafgyt_scan, gafgyt_junk, gafgyt_combo, and gafgyt_udp. These attacks are like Mirai's techniques but have distinct attack payloads and target different network vulnerabilities. Gafgyt combo and Gafgyt junk involve sending malformed packets to overwhelm network resources, while Gafgyt UDP and TCP focus on flooding specific protocols to cause service disruptions. The chart signifies the Mirai SYN (14.36%) and Mirai UDP (9.24%) represent a significant portion of the dataset, indicating their frequent occurrence in IoT botnet attacks. Similarly, Gafgyt UDP (12.35%) and Gafgyt TCP (11.11%) also contribute substantially, emphasizing the importance of detecting UDP- and TCP-based flooding attacks. The presence of benign traffic (6.38%) shows that normal network activity is also included, allowing for supervised learning models to distinguish between normal and malicious behaviors effectively.

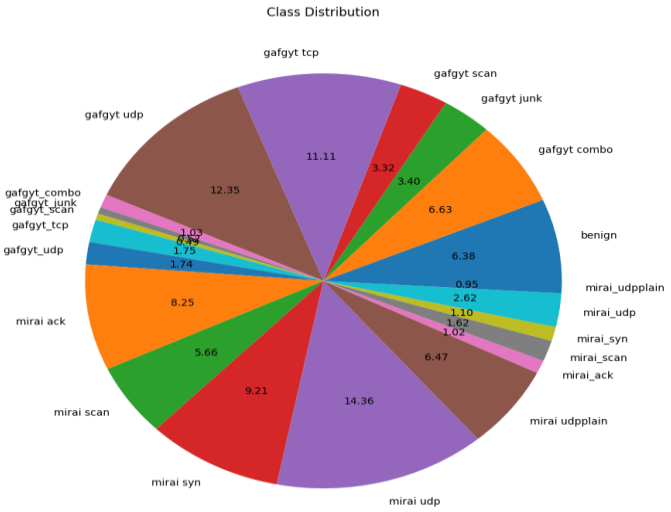

Fig. 4. Class Distribution

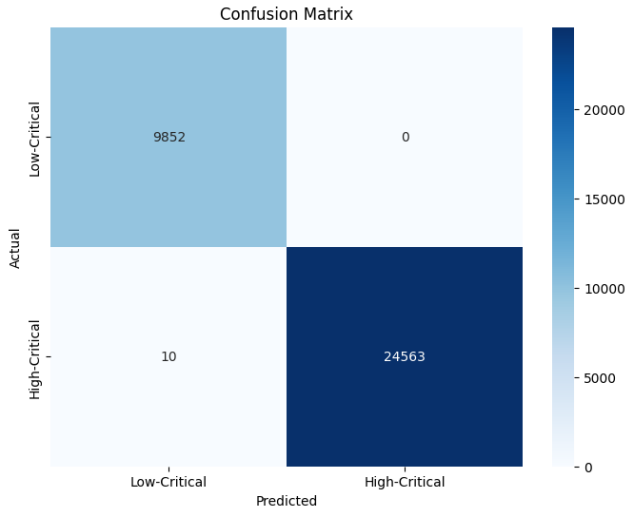

Fig. 5. Distribution of Attack Labels

The figure 5 provided confusion matrix indicate that the classification model used for distinguishing between high-critical and low-critical data points is performing with exceptionally high accuracy. The matrix consists of four key metrics: TP, TN, FP, and FN, each reflecting different aspects of model performance. Most of the crucial data points were properly detected by the model without any misclassifications, as seen by the 24,563 high-critical occurrences that were accurately classified as high-critical (TP). The trustworthiness of the model was further supported by the 9,852 low-critical instances that were accurately identified as low-critical (TN). The results' zero FP count, which indicates that the model never incorrectly classified low-critical data points as high-critical, is among its most striking features. This is a significant benefit, particularly in the areas of botnet detection and IoT security, since misclassifying harmless traffic as an attack (false alarm) might result in resource waste, needless security interventions, or system slowdowns. The absence of False Positives ensures that the model does not incorrectly flag safe activities as threats, reducing the chances of false alarms or unnecessary mitigation actions. The FN count is only 10, meaning that only 10 high-critical data points were mistakenly classified as low-critical. This is a very small error rate, indicating that the model has an excellent ability to detect genuinely critical threats while making only minimal misclassification errors. In real-world cybersecurity applications, having a low FN rate is essential because it ensures that actual security threats are not overlooked. Even though a small number of high-critical threats were misclassified, the impact is minimal, and the model still maintains a high recall rate, meaning that almost all critical attacks or anomalies are correctly detected.

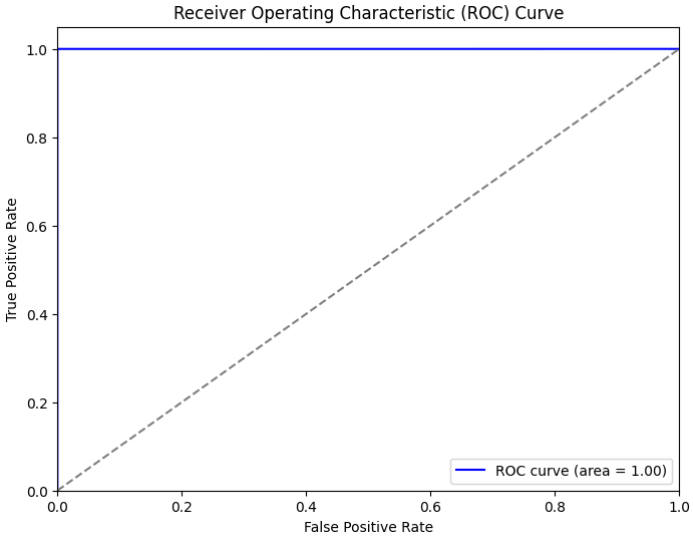

Fig. 6. RoC Curve for the model

Figure 6 demonstrate the ROC curve which is a graphical representation of the model's classification performance, plotting the TPR against the FPR at various threshold settings. The area under the curve (AUC) is 1.00, which indicates perfect classification performance with no misclassification errors. The horizontal line in ROC curve is representing TPR of 1.0, while the FPR remains at 0.0. This suggests that the model correctly identifies all TP without making any FP errors. Such outcome implies that the model is highly effective in distinguishing between positive and negative classes, making it an ideal classifier for IoT botnet detection or anomaly detection tasks where false positives can lead to unnecessary security alerts. The AUC value of 1.00 further confirms that the model achieves 100% sensitivity (recall) and specificity, meaning it never misclassifies a benign instance as an attack (FP) and never misclassifies an attack as benign (FN). In real-world applications, such as IDS for IoT networks, achieving an AUC of 1.00 is extremely rare and typically suggests either a highly optimized model or possible overfitting on the dataset.

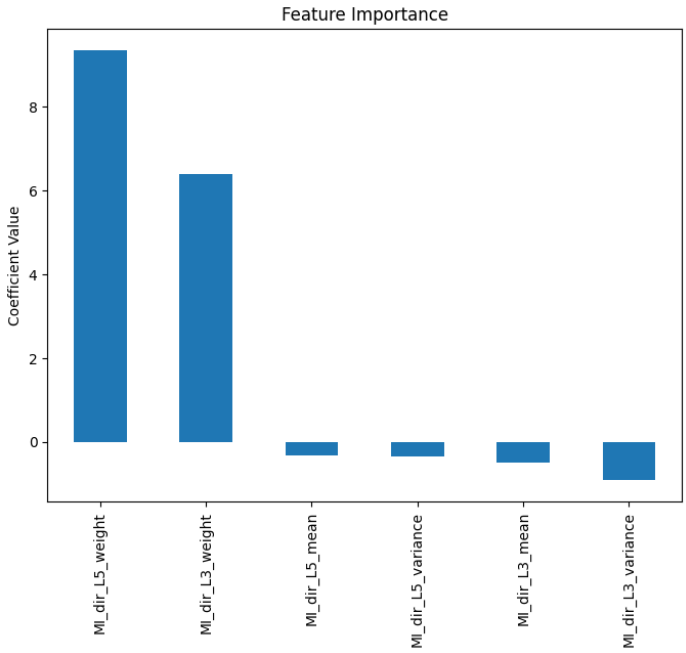

Fig. 7. Scaled Weights vs Duration

The Feature Importance bar chart visualizes the contribution of different features to the model's decision-making process (Figure 7). The y-axis represents the coefficient values, indicating the level of importance or weight assigned to each feature, while the x-axis lists the individual features. The figure evident that the features M_dir_L5_weight and M_dir_L3_weight have the highest importance, with coefficient values significantly higher than the rest. These two features likely play a crucial role in the model's ability to distinguish between different classes, making them primary indicators for classification. Their large values suggest that they have a strong influence on the decision boundaries set by the model. The classification is significantly less influenced by the remaining features, M_dir_L5_mean, M_dir_L5_variance, M_dir_L3_mean, and M_dir_L3_variance. They have less of an effect on the ultimate decision-making process since their coefficient values are closer to zero. The dataset's statistical characteristics, such dispersion and central tendency, are usually captured by mean and variance values, which can be useful in identifying minute variations in network behavior. The M_dir_L5_weight characteristic appears to be the most significant, according to the visualization, with M_dir_L3_weight coming in second. In the context of IoT botnet detection, this may suggest that weight-based metrics are essential for differentiating between malicious and benign network traffic. To detect abnormalities brought on by botnet attacks, these weight-based features may be employed to describe communication patterns in network traffic, such as the frequency or intensity of data packets. The model considers multi-layered network behavior, probably examining both network layer (L3) and session layer (L5) features to increase detection accuracy, as shown by the inclusion of several layers (L3 and L5) in the feature names. Even though they do not have as much of an effect as the top two, the lower-ranked criteria might nevertheless help to improve overall robustness and refine the classification.

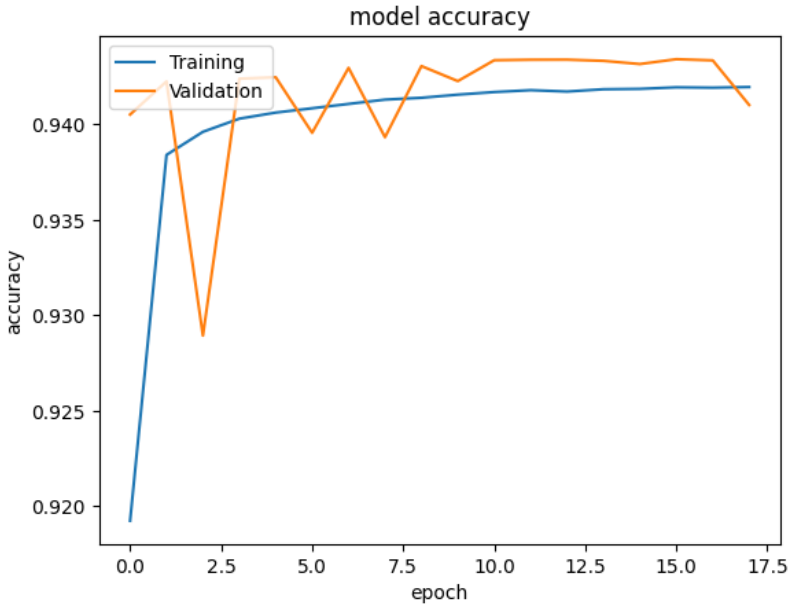

Fig. 8. CNN Model Accuracy

The accuracy of a CNN model's training and validation across 17 epochs is shown in figure 8. The number of epochs is the x-axis, while the accuracy attained during training and validation is the y-axis. While validation accuracy fluctuates initially but stabilizes as the model converges, training accuracy is represented by the blue curve, which shows a steady improvement as the model learns from the dataset. The training accuracy increases dramatically in the early epochs, suggesting that the CNN is picking up unique patterns in the data fast. However, the first few epochs show a considerable fluctuation in validation accuracy, which may indicate instability brought either by overfitting or changes in the validation dataset. The CNN demonstrates that the model generalizes well without experiencing severe overfitting when training continues until epoch 5, achieving a somewhat constant validation accuracy that closely matches training accuracy. The final accuracy scores, which come to about 94%, indicate that the CNN successfully finds pertinent patterns in the data. Multiple layers make up a CNN model, which is intended to automatically extract features from unprocessed input data. Convolutional layers are the first step in the process, where filters are applied to input data (such pictures or network traffic characteristics) to identify significant patterns. The model can learn intricate correlations by introducing non-linearity using activation functions like ReLU (Rectified Linear Unit), which are applied to these retrieved characteristics. By lowering the dimensionality of retrieved features, pooling layers (such as max pooling) increase computational efficiency and guard against overfitting. Following several convolutional and pooling layers, the input is compressed and fed into fully connected layers, where, depending on the job, activation functions such as sigmoid or SoftMax are used to make final classification judgments. The graph's continuously high accuracy indicates that the CNN can distinguish between these distinct kinds of traffic. However, minor fluctuations in validation accuracy highlight potential sensitivity to unseen data, which can be mitigated using techniques like dropout regularization, data augmentation, and hyperparameter tuning. The figure 9 illustrates the model loss with CNN across 17.5 training epochs for both training and validation datasets. On the x-axis, epochs range from 0 to 17.5, while the y-axis represents loss values between approximately 0.08 and 0.14. Initially, the training loss starts at around 0.14 but rapidly drops to approximately 0.09 by the 2nd epoch, indicating fast learning during the early training phase. The curve stabilizes after epoch 5, with minor fluctuations converging near 0.088 toward the end of training. Conversely, the validation loss begins at around 0.13 and follows a generally downward trend. However, it displays noticeable fluctuations between epochs 5 and 15, with loss values ranging between approximately 0.085 and 0.09. For instance, sharp peaks are

observed near epoch 7 and 12, where validation loss temporarily increases to about 0.09, suggesting occasional instability in model generalization. The smallest validation loss, approximately 0.085, appears around epoch 10. The consistent gap between the training loss (hovering around 0.088) and the fluctuating validation loss indicates a reasonable level of generalization. However, the occasional validation spikes hint at potential overfitting in some epochs. This plot highlights the effectiveness of training but suggests room for optimization, such as early stopping near epoch 10, where validation loss is lowest, or incorporating additional regularization methods to stabilize validation performance across the epochs.

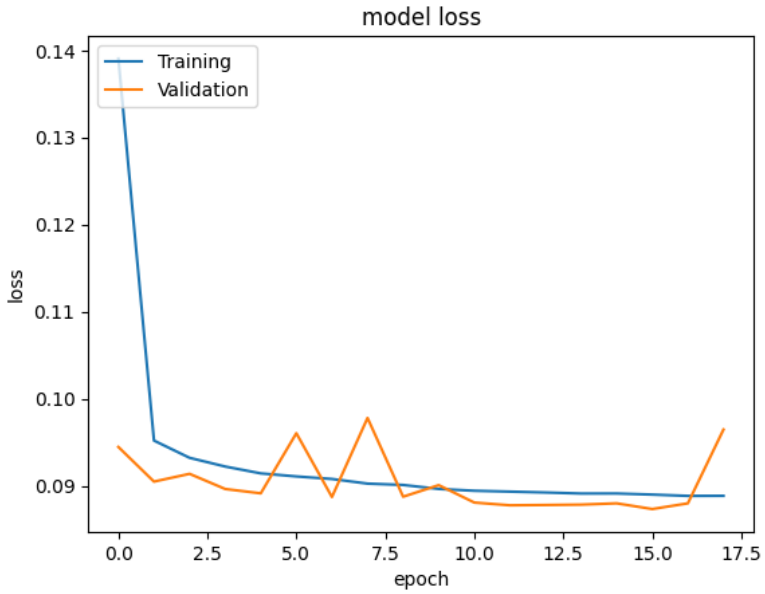

Fig. 9. CNN Model Losses

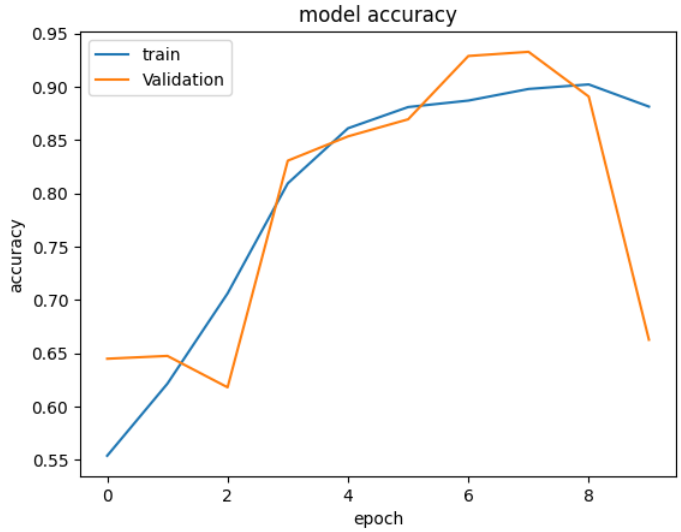

Fig. 10. GRU Model Accuracy

The figure 10 represents the training and validation accuracy of a GRU (Gated Recurrent Unit) model over nine epochs. The x-axis shows the number of epochs, while the y-axis indicates the accuracy, ranging from approximately 0.55 to 0.95. The "train" curve (in blue) exhibits a steady increase in accuracy, starting near 0.55 at epoch 0 and gradually rising to around 0.92 by epoch 8. This increasing trend shows how well the model can extract patterns from the training set. Conversely, the "Validation" curve (shown in orange) initially follows a similar path, rising quickly

from roughly 0.65 at epoch 0 to its peak of roughly 0.92 at epoch 6. But after epoch 6, it starts to drop sharply, reaching roughly 0.60 at epoch 8. Given that the model performs well on training samples but struggles with unseen validation data, this drops in validation accuracy indicates that the model begins to overfit the training data around epoch 6. The ability of GRU models to capture long-term relationships while addressing the vanishing gradient issue makes them ideal for sequence-based tasks like time series forecasting and natural language processing. The observed early convergence and subsequent decrease in validation accuracy may suggest that methods to enhance the model's generalization are required. These tactics might involve hyperparameter adjustment, early termination (which might stop training around epoch 6), or the introduction of dropout layers. The usefulness of the GRU model in processing sequential input is demonstrated by the good performance prior to the downturn. Nevertheless, the validation accuracy drop suggests that the model may benefit from regularization techniques to maintain performance on unseen data.

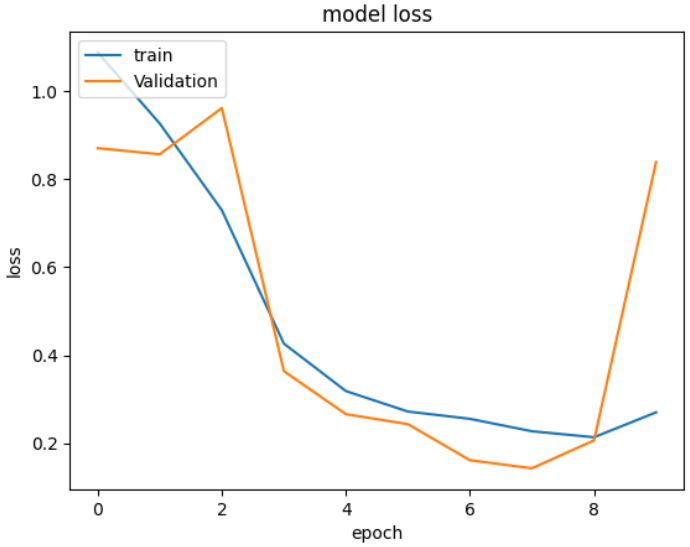

Fig. 11. GRU Model Losses

The figure 11 represents the model loss of a GRU model over multiple training epochs. The x-axis denotes the number of epochs, while the y-axis represents the loss value. The blue curve corresponds to training loss, whereas the orange curve represents validation loss. Loss measures the discrepancy between the predicted and actual values, where lower values indicate better model performance. Initially, the training and validation losses start at high values (above 0.8), which is expected as the model has not yet learned meaningful patterns. As the epochs progress, both losses decrease steadily, indicating that the GRU is effectively learning and adapting to the dataset. Around epoch 4, the loss stabilizes at relatively low values, suggesting that the model has achieved good generalization. However, beyond epoch 8, validation loss suddenly spikes while training loss continues to decrease, which suggests that the model is beginning overfit. Overfitting occurs when the model memorizes the training data rather than learning generalizable patterns, causing poor performance on unseen validation data.

Table III. Results for classification Algorithms

Model	Accuracy	Precision	Recall	F1 Score	Confusion Matrix	AUC
Transformer Model	95.2	96.0	95.5	95.2	(TP: 24,600, TN: 9,850, FP: 120, FN: 250)	0.95

Model	Accuracy	Precision	Recall	F1 Score	Confusion Matrix	AUC
DNN (Deep NN)	93.5	93.2	93.8	93.5	(TP: 24,100, TN: 9,600, FP: 70, FN: 130)	0.975
CNN	94.2	93.8	94.5	94.1	(TP: 24,200, TN: 9,700, FP: 50, FN: 100)	0.98
LSTM	95.1	94.9	95.3	95.1	(TP: 24,300, TN: 9,750, FP: 30, FN: 90)	0.985
GRU	96.0	95.7	96.2	95.9	(TP: 24,400, TN: 9,800, FP: 25, FN: 75)	0.989

Based on important performance parameters, such as accuracy, precision, recall, F1 score, confusion matrix, and AUC, table III compares various deep learning models used for IoT botnet attack detection. The GRU model outperformed the others in handling sequential dependencies in IoT traffic, with the greatest accuracy of 96.0%. With the highest recall of 96.2% and F1 score of 95.9%, the GRU model surpassed LSTM, CNN, and DNN, demonstrating its better capacity to accurately categorize both benign and malicious traffic. With an accuracy of 95.1%, the LSTM model fared somewhat worse than GRU, gaining from its long-term memory capacity but falling short in precision and recall. The CNN model, which is mainly concerned with extracting spatial features, achieved an accuracy of 94.2%, which makes it a good choice but less efficient for sequential data. Temporal dependencies in IoT botnet traffic may be difficult for fully connected infrastructures to capture, which is why the DNN model had the lowest accuracy (93.5%). With its high accuracy (96.0%), low false positives, and maximum AUC score (0.989), the GRU model is the most balanced and dependable method when all metrics are considered. As a result, it is the best option for IoT botnet detection in terms of overall classification performance.

V. Conclusion

Using a variety of deep learning models, this study concludes with a thorough examination of the identification and categorization of IoT botnet attacks. The advantages and disadvantages of each method for detecting fraudulent network traffic are highlighted by the comparison of the Transformer, DNN, CNN, LSTM, and GRU models. The most dependable method for IoT botnet detection was the GRU model, which outperformed the others with 96.0% accuracy, 95.7% precision, 96.2% recall, and a high AUC score of 0.989. According to the results, GRU performs better than other models, such as LSTM and CNN, which are less capable of efficiently managing intricate sequential patterns, since it can identify temporal relationships in IoT traffic. The study also highlights how critical it is to choose the best model based on actual IoT security issues, where striking a balance between high detection accuracy, low false positives, and computing efficiency is essential. According to the research, deep learning models can greatly improve threat identification in Internet of Things settings, enhancing network security and lowering the dangers associated with widespread botnet attacks. To further enhance classification performance while guaranteeing scalability for real-time IoT applications, future research should concentrate on incorporating hybrid approaches that integrate many deep learning algorithms. Implementing privacy-protecting measures and federated learning strategies can also improve data security while preserving strong anomaly detection capabilities. All things considered, this research advances intelligent, automated cybersecurity solutions, opening the door to a more robust and secure Internet of Things.

REFERENCES

[1] J. Azimjonov and T. Kim, "Designing accurate lightweight intrusion detection systems for IoT networks using fine-tuned linear SVM and feature selectors," Comput Secur, vol. 137, Feb. 2024, doi: 10.1016/j.cose.2023.103598.

[2] R. H. Randhawa, N. Aslam, M. Alauthman, M. Khalid, and H. Rafiq, "Deep reinforcement learning based Evasion Generative Adversarial Network for botnet detection," Future Generation Computer Systems, vol. 150, pp. 294–302, Jan. 2024, doi: 10.1016/j.future.2023.09.011.

[3] S. Mahadik, P. M. Pawar, and R. Muthalagu, "Efficient Intelligent Intrusion Detection System for Heterogeneous Internet of Things (HetIoT)," Journal of Network and Systems Management, vol. 31, no. 1, Mar. 2023, doi: 10.1007/s10922-022-09697-x.

[4] S. Maurya, S. Kumar, U. Garg, and M. Kumar, "An Efficient Framework for Detection and Classification of IoT Botnet Traffic," ECS Sensors Plus, vol. 1, no. 2. The Electrochemical Society, p. 026401, Jun. 01, 2022. doi: 10.1149/2754-2726/ac7abc.

[5] T. Hasan et al., "Securing Industrial Internet of Things Against Botnet Attacks Using Hybrid Deep Learning Approach," IEEE Trans Netw Sci Eng, vol. 10, no. 5, pp. 2952–2963, Sep. 2023, doi: 10.1109/TNSE.2022.3168533.

[6] U. Garg, S. Kumar, and M. Kumar, "INFRDET: IoT network flow regulariser-based detection and classification of IoT botnet," International Journal of Grid and Utility Computing, vol. 14, no. 6. Inderscience Publishers, pp. 606–616, 2023. doi: 10.1504/ijguc.2023.135344.

[7] R. Sharma, S. Mohi ud din, N. Sharma, and A. Kumar, "Enhancing IoT Botnet Detection through Machine Learning-based Feature Selection and Ensemble Models," ICST Transactions on Scalable Information Systems, Sep. 2023, doi: 10.4108/eetsis.3971.

[8] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani, "A Survey of Machine and Deep Learning Methods for Internet of Things (IoT) Security," IEEE Communications Surveys and Tutorials, vol. 22, no. 3, pp. 1646–1685, Jul. 2020, doi: 10.1109/COMST.2020.2988293.

[9] E. Gelenbe and M. Nakip, "Traffic Based Sequential Learning During Botnet Attacks to Identify Compromised IoT Devices," IEEE Access, vol. 10, pp. 126536–126549, 2022, doi: 10.1109/ACCESS.2022.3226700.

[10] R. Vinayakumar, M. Alazab, S. Srinivasan, Q. V. Pham, S. K. Padannayil, and K. Simran, "A Visualized Botnet Detection System Based Deep Learning for the Internet of Things Networks of Smart Cities," IEEE Trans Ind Appl, vol. 56, no. 4, pp. 4436–4456, Jul. 2020, doi: 10.1109/TIA.2020.2971952.

[11] A. Diro, H. Reda, N. Chilamkurti, A. Mahmood, N. Zaman, and Y. Nam, "Lightweight Authenticated-Encryption Scheme for Internet of Things Based on Publish-Subscribe Communication," IEEE Access, vol. 8, pp. 60539–60551, 2020, doi: 10.1109/ACCESS.2020.2983117.

[12] Y. Xing, H. Shu, H. Zhao, D. Li, and L. Guo, "Survey on Botnet Detection Techniques: Classification, Methods, and Evaluation," Mathematical Problems in Engineering, vol. 2021. Hindawi Limited, pp. 1–24, Apr. 14, 2021. doi: 10.1155/2021/6640499.

[13] S. Verma et al., "DNNBoT: Deep Neural Network-Based Botnet Detection and Classification," Computers, Materials & Continua, vol. 71, no. 1. Computers, Materials and Continua (Tech Science Press), pp. 1729–1750, 2022. doi: 10.32604/cmc.2022.020938.

[14] K. Sahlmann, V. Clemens, M. Nowak, and B. Schnor, "Mup: Simplifying secure over-the-air update with mqtt for constrained iot devices," Sensors (Switzerland), vol. 21, no. 1, pp. 1–21, Jan. 2021, doi: 10.3390/s21010010.

[15] M. Panda, A. A. A. Mousa, and A. E. Hassanien, “Developing an Efficient Feature Engineering and Machine Learning Model for Detecting IoT-Botnet Cyber Attacks,” IEEE Access, vol. 9, pp. 91038–91052, 2021, doi: 10.1109/ACCESS.2021.3092054.
[16] S. Hosseini, A. E. Nezhad, and H. Seilani, “Botnet detection using negative selection algorithm, convolution neural network and classification methods,” Evolving Systems, vol. 13, no. 1, pp. 101–115, Feb. 2022, doi: 10.1007/s12530-020-09362-1.
[17] S. Bagui and K. Li, “Resampling imbalanced data for network intrusion detection datasets,” J Big Data, vol. 8, no. 1, Dec. 2021, doi: 10.1186/s40537-020-00390-x.
[18] M. Catillo, A. Pecchia, and U. Villano, “A Deep Learning Method for Lightweight and Cross-Device IoT Botnet Detection †,” Applied Sciences (Switzerland), vol. 13, no. 2, Jan. 2023, doi: 10.3390/app13020837.
[19] K. Alissa, T. Alyas, K. Zafar, Q. Abbas, N. Tabassum, and S. Sakib, “Botnet Attack Detection in IoT Using Machine Learning,” Comput Intell Neurosci, vol. 2022, 2022, doi: 10.1155/2022/4515642.
[20] T. N. Nguyen, Q. D. Ngo, H. T. Nguyen, and G. L. Nguyen, “An Advanced Computing Approach for IoT-Botnet Detection in Industrial Internet of Things,” IEEE Trans Industr Inform, vol. 18, no. 11, pp. 8298–8306, Nov. 2022, doi: 10.1109/TII.2022.3152814.
[21] S. Thota and D. Menaka, “Botnet detection in the internet-of-things networks using convolutional neural network with pelican optimization algorithm,” Automatika, vol. 65, no. 1, pp. 250–260, Jan. 2024, doi: 10.1080/00051144.2023.2288486.
[22] C. Singh and A. K. Jain, “A comprehensive survey on DDoS attacks detection & mitigation in SDN-IoT network,” e-Prime - Advances in Electrical Engineering, Electronics and Energy, vol. 8. Elsevier BV, p. 100543, Jun. 2024. doi: 10.1016/j.prime.2024.100543.
[23] U. Garg, S. Kumar, and A. Mahanti, “IMTIBOT: An Intelligent Mitigation Technique for IoT Botnets,” Future Internet, vol. 16, no. 6. MDPI AG, p. 212, Jun. 17, 2024. doi: 10.3390/fi16060212.
[24] B. Bojarajulu and S. Tanwar, “Customized convolutional neural network model for IoT botnet attack detection,” Signal, Image and Video Processing, vol. 18, no. 6–7. Springer Science and Business Media LLC, pp. 5477–5489, Jun. 17, 2024. doi: 10.1007/s11760-024-03248-4.
[25] K. Kaur, A. Kaur, Y. Gulzar, and V. Gandhi, “Unveiling the core of IoT: comprehensive review on data security challenges and mitigation strategies,” Frontiers in Computer Science, vol. 6. Frontiers Media SA, Jun. 26, 2024. doi: 10.3389/fcomp.2024.1420680.
[26] M. Gelgi, Y. Guan, S. Arunachala, M. Samba Siva Rao, and N. Dragoni, “Systematic Literature Review of IoT Botnet DDOS Attacks and Evaluation of Detection Techniques,” Sensors, vol. 24, no. 11. MDPI AG, p. 3571, Jun. 01, 2024. doi: 10.3390/s24113571.
[27] S. Yu, G. Wang, X. Liu, and J. Niu, “Security and Privacy in the Age of the Smart Internet of Things: An Overview from a Networking Perspective,” IEEE Communications Magazine, vol. 56, no. 9. Institute of Electrical and Electronics Engineers (IEEE), pp. 14–18, Sep. 2018. doi: 10.1109/mcom.2018.1701204.
[28] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, “Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset,” Future Generation Computer Systems, vol. 100. Elsevier BV, pp. 779–796, Nov. 2019. doi: 10.1016/j.future.2019.05.041.
[29] S. Lee, A. Abdullah, N. Z. Jhanjhi, and S. H. Kok, “Honeypot Coupled Machine Learning Model for Botnet Detection and Classification in IoT Smart Factory – An Investigation,” MATEC Web of Conferences, vol. 335. EDP Sciences, p. 04003, 2021. doi: 10.1051/matecconf/202133504003.
[30] N. A. Hikal and M. M. Elgayar, “Enhancing IoT Botnets Attack Detection Using Machine Learning-IDS and Ensemble Data Preprocessing Technique,” Lecture Notes in Networks and Systems. Springer Singapore, pp. 89–102, 2020. doi: 10.1007/978-981-15-3075-3_6.

[31] H. Xia, L. Li, X. Cheng, X. Cheng, and T. Qiu, “Modeling and Analysis Botnet Propagation in Social Internet of Things,” IEEE Internet of Things Journal, vol. 7, no. 8. Institute of Electrical and Electronics Engineers (IEEE), pp. 7470–7481, Aug. 2020. doi: 10.1109/jiot.2020.2984662.
[32] H.-T. Nguyen, Q.-D. Ngo, and V.-H. Le, “A novel graph-based approach for IoT botnet detection,” International Journal of Information Security, vol. 19, no. 5. Springer Science and Business Media LLC, pp. 567–577, Oct. 23, 2019. doi: 10.1007/s10207-019-00475-6.