IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 17 January 2025, accepted 5 February 2025, date of publication 11 February 2025, date of current version 20 February 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3541125

RESEARCH ARTICLE

Ensemble Network Graph-Based Classification for Botnet Detection Using Adaptive Weighting and Feature Extraction

MUHAMMAD AIDIEL RACHMAN PUTRA[1], TOHARI AHMAD[1], (Member, IEEE), DANDY PRAMANA HOSTIADI[2], (Member, IEEE), AND ROYYANA MUSLIM IJTIHADIE[1]

[1]Department of Informatics, Institut Teknologi Sepuluh Nopember, Surabaya 60111, Indonesia
[2]Department of Magister Information Systems, Institut Teknologi dan Bisnis STIKOM Bali, Denpasar 80234, Indonesia

Corresponding author: Tohari Ahmad (tohari@if.its.ac.id; tohari@its.ac.id)

This work was supported by Pendidikan Magister menuju Doktor untuk Sarjana Unggul (PMDSU) Scholarship and research grant from the Ministry of Higher Education, Science, and Technology, Republic of Indonesia.

ABSTRACT The number of cybersecurity threats increases every year due to the rapid improvement of methods and tools used by hackers to infect devices. These threats form a network, which is called a botnet, to send and receive commands. Botnets can launch malicious attacks using malware to infect targets in the network and then control them to do illegal things. Previous research has demonstrated that security systems can identify attacks by analyzing communication among bots in a network using a graphing approach. While this analytical method demonstrates satisfactory accuracy, it still faces challenges related to low recall, precision, and F1-score, due to issues such as imbalanced data and the complexity of botnet behavior. This research addresses these challenges by analyzing network flow using adaptive weighting and feature extraction. Network flows are represented in a graph with IP addresses as vertices and communication links between IP addresses as edges. Since botnet attack activity forms a relatively small percentage compared with millions of recorded network flow data, the data is grouped using time gap analysis to handle the imbalance problem. Furthermore, network flows are represented in two graphs, and each edge is weighted based on the 16 types of weighting. The graph representation and weighting output are stored in out-degree and in-degree graph metadata for classification. The analysis is carried out in an ensemble manner with weighting and threshold values to determine whether an IP address is a botnet or a normal host. The experimental results obtained using CTU-13, NCC, and NCC-2 datasets produce reliable performance with an average accuracy of 99.99%, along with 80.91% precision, 93.10% recall, 82.15% f1-score and 39.55 second execution time. The proposed model can function as an effective tool for the forensic analysis of botnet attacks, allowing network administrators to analyze the characteristics of botnet activities and anticipate potential future threats.

INDEX TERMS Botnet detection, graph-based representation, machine learning, network security, information security, network infrastructure.

I. INTRODUCTION

The constant advancement of methods and tools used by attackers corresponds with the growing number of cyber attacks reported [1], [2], [3]. They commonly use malware to target and infiltrate computer networks [4]. By injecting malware into their targets, attackers can execute malicious activities, such as data theft, system damage, and target control [5], [6], [7]. The malware utilized to control a network of infected devices, known as a botnet [5], [6], [7], comprises bots/zombies and botmasters [8], [9]. Bots/zombies are a cluster of infected devices, while the botmaster is the entity issuing commands to that cluster [10]. Botmasters can direct groups of bots to engage in various illegal activities, including click fraud, mining, Distributed Denial of Service (DDoS), and SPAM [11], [12], [13].

The associate editor coordinating the review of this manuscript and approving it for publication was Alessio Giorgetti.

Detecting botnets is a challenging task due to their rapid evolution [5], [7], [12]. Initially, botnets used a centralized architecture with command and control (C&C) at the center [14]. The C&C facilitated communication between the botmaster and each bot. While a centralized architecture is easier to maintain, it is also easier to detect. The intense communication activities of C&C make their behavior easily recognizable [14]. In a centralized botnet, shutting down C&C access halts all botnet activity [14]. To address this vulnerability, botnet architecture evolved into a decentralized structure [11], where all bots on the network can serve as communication channels. Shutting down suspected C&C or botmaster access in a decentralized architecture does not stop botnet activity, as other connected bots can be utilized for communication [14]. It is more difficult to stop decentralized botnets but require substantial effort to maintain. As a hybrid botnet that combine centralized and decentralized architecture [15], it comprises three layers: a central server, multiple C&C servers, and bots. While a hybrid architecture is easier to maintain, its complex structure makes its activity difficult to detect [15].

Extensive research has been undertaken to create optimal detection models for botnets, which are highly complex and continuously evolving [1]. Previous studies have investigated various approaches, such as anomaly-based and signature-based methods, to tackle the challenge of differentiating botnet activity from legitimate network traffic [16]. Machine learning, deep learning, and graph-based detection models have demonstrated superior performance in recognizing botnet attack patterns; however, they remain limited in their ability to adapt to the evolving nature of botnets and the issues related to data imbalance [17], [18], [19]. Existing research indicates that graph-based approaches are more adaptable, as they emphasize activity intensity and the similarity between nodes. Nonetheless, they still face challenges in terms of precision and recall when working with imbalanced datasets [19], [20], [21].

This research introduces a detection model used for forensic analysis of botnet attacks by integrating graph-based approaches with ensemble classification. The motivation behind this work is to develop a detection model that can identify botnet behavior by analyzing communication patterns between bots within a network using a graph-based approach. Network flows are grouped into activity groups and then represented in a graph where IP addresses serve as nodes, and the communications between them are depicted as edges. This representation yields two graphs, in-degree and out-degree, with edges weighted using 16 distinct methods. The graph metadata, including vertex and edge information, is then extracted and transformed into new features for machine-learning classification. The confidence scores from both graph types are analyzed using an ensemble method, enhanced through weighting and threshold analysis, to assess whether an IP address is associated with a botnet.

The graph-based network flow analysis is designed to adapt to the evolving nature of botnets, as it examines one of their core characteristics: communication intensity. The primary objective of the proposed model is to detect botnet activity by analyzing the underlying communication network. The key contributions of this research are as follows:

1) A feature extraction model based on network flow representation using graphs and edge weighting, employing 16 weighting techniques.
2) A graph classification methodology derived from the metadata produced by the network flow representation in the graph.
3) An ensemble classification method that integrates the classification results from both in-degree and out-degree graphs, utilizing confidence score analysis with adaptive weighting and threshold values.
4) A robust model for botnet detection capable of identifying three distinct characteristics: periodic, sporadic, and simultaneous botnet activities.
5) A comparative analysis of the proposed model against prior research, focusing on network flow analysis and graph-based techniques.

This paper is organized as follows. Section II provides a review of previous research on botnet detection models, approaches for addressing data imbalance, and the use of graph-based methods for network analysis. Section III explains the proposed method in detail. Section IV describes the experimental environment, and analysis the experimental results. Finally, Section V concludes the study by summarizing the key findings from the experiment.

II. RELATED WORKS

Detecting botnets presents a significant challenge due to the unpredictable nature of their characteristics and their ability to adapt to normal network activity. Various detection models have been introduced to accurately identify these evolving traits. Three primary approaches commonly used are signature-based [16], [22], statistical-based [14], [23], [24], and artificial intelligence-based techniques [11], [17], [18], [25], [26], [27].

Signature-based models have proven effective in accurately detecting botnets, but they rely on the development of a knowledge base that must continuously evolve alongside botnet behavior [16], [22]. On the other hand, statistical-based approaches are more straightforward to implement and monitor than signature-based methods. These statistical models utilize fundamental calculations such as similarity measures, correlation, and analysis of variance [23], [24]. The third, the artificial intelligence-based approaches, particularly those utilizing deep learning algorithms, have gained popularity due to their ease of implementation, especially since they often do not require extensive data pre-processing. These methods are frequently implemented for their ability to achieve high accuracy in detecting botnet activity [11], [17], [18], [25], [26], [27]. Additionally, hybrid detection models that combine multiple approaches are proposed by integrating the strengths of different methodologies to improve detection performance [14], [24], [26].

A. FLOW-BASED BOTNET DETECTION

Botnets execute their activities across networks, with these actions being recorded as network flow, traffic, or activity logs. Detection models that aim to identify botnets within network flow are referred to as flow-based detection models. One of the primary challenges of this flow-based detection lies in the huge volume of data, which is overwhelmingly dominated by normal network activity. Some methods have been introduced for detecting botnets by classifying network flows into normal and malicious activity.

Hosseini et al. [28] proposed a method for detecting network flow and traffic using a combination of Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN). Their approach incorporates correlation analysis to reduce noise and select uncorrelated features, enhancing the accuracy of botnet detection. The training data is then processed using the Negative Selection Algorithm (NSA), which generates a model implemented in the classification phase using LSTM-CNN.

Similarly, Hostiadi and Ahmad [14] applied correlation analysis in their hybrid detection model. They integrated correlation measurements with similarity analysis to identify patterns between activity groups. In this case, an activity group is defined as a cluster of botnet attacks occurring simultaneously. Pearson correlation measurements were taken to identify causal relationships between group attack activities, facilitating more accurate detection. Differently, Joshi et al. [17] combined fuzzy logic with an Artificial Neural Network (ANN) to detect the activity. All features within the network flow were extracted using fuzzy logic to generate fuzzy sets, which were then utilized in the ANN classification phase to distinguish between suspected botnet and normal network traffic.

These studies leverage various methodologies, including machine learning, neural networks, and correlation and similarity analysis, to enhance botnet detection. However, as botnets continue to evolve and mimic normal network activities more closely, this presents a significant challenge for flow-based detection models. To address the rapid evolution of botnet behavior within network flow, advanced techniques such as active learning, incremental learning models, and graph-based detection approaches have been proposed.

B. GRAPH-BASED BOTNET DETECTION

Although numerous studies have focused on detecting network flow anomalies [14], [17], [28], several others have specifically targeted the detection of botnet hosts [18], [19], [20], [21], [24], [29], [30], [31], [32]. Botnet host detection models predominantly implement graph-based approaches, which, while challenging to implement, are highly effective in detecting rapidly evolving botnets. This effectiveness is caused by their focus on analyzing botnet characteristics within the network rather than solely on network flow behavior [33].

Chowdhury et al. [29] employed a graph-based approach for botnet detection with fundamental technique, utilizing clustering with seven graph features, Self-organizing Maps, and filtering inactive nodes using Support Vector Machines (SVM). They reported achieving a 100% detection rate; however, the absence of additional evaluation metrics, such as precision and F-1 score, limits the ability to comprehensively assess the model's performance. In contrast, Wang et al. [24] introduced a hybrid botnet detection model that integrates both flow-based and graph-based approaches. This model identifies botnet behavior using a set of 18 features: 15 based on network flow and 3 on graph-based characteristics. The initial phase of the model involves filtering irrelevant network flows, such as those that fail to complete the handshake process. Additionally, network flows are filtered based on the origin server, referencing a list of 1,000 popular websites from Alexa.com that are considered safe and unlikely to serve as C&C botnet servers. The second applies network flow similarity analysis, based on the assumption that botmasters control botnets from the same location, causing the botnet's behavior to exhibit similarity [24]. Next, the model evaluates network flow based on the stability of distributed packet sizes. Following this, a graph-based analysis is conducted to examine the neighborhood characteristics of botnet-related nodes. Finally, the results of the similarity, stability, and anomaly analyses are combined using a voting mechanism. This hybrid model demonstrates reliable botnet detection performance and achieves high accuracy. However, it still faces challenges in distinguishing botnet activity from distributed devices that exhibit behavior similar to C&C botnets.

A detection model using a graph-based approach was also proposed by Daya et al. [20], which implements a two-phase methodology to detect botnets through both supervised and unsupervised machine learning techniques. It comprises three primary components: data bootstrapping, model training, and inference. The data bootstrapping phase involves several steps of data preparation and feature extraction, including flow ingestion, graph transformation, feature extraction, and feature normalization. The training phase consists of two parts: Phase 1 uses unsupervised learning to form host clusters, while Phase 2 focuses on botnet detection using supervised learning techniques. Experimental results using the CTU-13 dataset demonstrate its efficacy in detecting botnets, achieving an impressive detection accuracy of 99.99%. However, this method exhibits certain limitations, notably in precision values and execution time [20].

Similarly, Blaise et al. [30] introduced BotFP (BotFinger-Printing), which follows a five-stage process: flow records collection, host network filtering and grouping, quantification, offline training, and online classification. In this approach, network flow records are grouped based on IP addresses and ports. The quantification stage is key to the model, as it determines the signature of each host, defined as the concatenation of the normalized frequency distributions of relevant attributes. During the training

phase, it applies various techniques, including clustering, supervised learning, and neural networks, to train the model to distinguish between malicious and benign hosts. The final stage, online classification, is used to classify hosts as either benign or malicious (bot). Experiments conducted with the CTU-13 dataset yielded optimal results, with a 100% detection rate for botnet hosts and an accuracy nearing 100%. Nevertheless, similar to other approaches, BotFP has limitations, particularly with respect to precision values.

The methods proposed by Kurt et al. [19] build upon the work of Abbas et al. [20] and Blaise et al. [30] by enhancing botnet detection models through the extraction of graph-based features, which are subsequently classified using machine learning algorithms. A total of ten graph features were extracted and then refined through five different feature selection techniques. These features were filtered based on consistency, correlation, and informational content. The filtered features were then used in the training and classification phases, taking six different classification algorithms: Naive Bayes, Decision Tree, Random Forest, AdaBoost, ExtraTrees Classifier, and k-Nearest Neighbors. The experimental results demonstrated that this approach significantly reduced time, computational complexity, and resource demands, while still delivering strong detection performance, as evidenced by precision and accuracy metrics.

Neural networks were adapted in [21] to detect botnet nodes. The proposed model facilitates automated forensic analysis through the use of grouped reversible residual connections. These connections, combined with graph isomorphism networks, are used to analyze expressive node representations from botnet communication graphs. The core of the research lies in the use of GNNExplainer, where saliency maps highlight nodes suspected to be part of botnets. Experimental results revealed that this model outperforms state-of-the-art methods such as GCN and XGBoost in detecting P2P botnets [21].

Several previous studies have also applied graph-based approaches to detect IoT botnets, each addressing different challenges [31], [32]. Nguyen et al. [32] proposed an approach that involves two graph generation processes: Function-call graph and PSI-Graph, which are then classified using a Convolutional Neural Network (CNN). The model was tested with 3,845 botnet samples and 6,165 benign samples. The proposed model achieved a commendable performance with an accuracy of 98.7%, an FNR of 1.83%, an FPR of 0.78%, and a processing time of 88 minutes. Similarly, Lefoane et al. [31] developed a graph-based detection model that implements a two-phase approach for detecting IoT botnets. The initial phase employs feature selection alongside machine learning classification to identify IoT botnets. The resulting detection data is then refined in the second phase through clustering with Latent Semantic Analysis (LSA), which helps uncover correlations between alerts. Utilizing a graph-based model and LSA for alert correlation analysis, this research has achieved a True Positive Rate (TPR) exceeding 99% and an F1-score of up to 100%. The experiments conducted in this study utilized the IoT-23 [34] dataset, which features distinct data characteristics compared to traditional botnets. Although this study attained optimal performance, it did not encounter issues such as data imbalance, as the IoT-23 dataset maintains a balanced proportion of benign data alongside attack data, all recorded in a controlled setting. In contrast, traditional botnets, captured in datasets like CTU-13 [35], NCC [36], and NCC-2 [37], along with network flow recordings from real environments, present significant data imbalance and substantial variation between benign and background data.

This research extends our previous work [18], which introduced a detection model based on the analysis of network flow visualization into graph form. The previous approach focused on detecting botnet activity by examining communication intensity. Network flows were visualized into two directed graphs: in-degree and out-degree, where each edge was weighted according to the communication intensity between nodes. The results of the visualization and weighting processes were used to extract four key features: Address, Degree, Weighted Degree, and Label. Experiments conducted using three different botnet datasets, each with distinct attack characteristics, showed strong performance in terms of accuracy and recall. However, the precision of the model still requires improvement. The analysis revealed that botnets exhibit unique characteristics that vary depending on the type of botnet, the nature of the attack, and the number of bots involved. These unique characteristics can serve as a valuable knowledge base for developing more robust detection models.

III. PROPOSED METHOD

This research introduces a botnet detection model employing a graph-based approach that incorporates adaptive weighting and feature extraction. The proposed method is specifically designed for forensic analysis, enabling network administrators to analyze the characteristics of botnet attacks. In this model, network flow is represented as a graph, where each edge is assigned with 16 a different type of weighting. Both in-degree and out-degree graph representations are utilized, and the resulting graphs are processed to extract metadata. This graph metadata serves as input for the subsequent classification process. The classification phase yields two outputs: one for the in-degree graph and another for the out-degree graph. In the final stage, these two classification results are analyzed based on their respective confidence scores to derive the final detection outcome.

An illustration of the proposed method is depicted in Fig. 1, while Fig. 2 provides a detailed explanation of the graph metadata extraction process. The terms and processes associated with the proposed method are defined as follows.

Definition 1 (Network Flow): Each network flow (nf) is represented as a tuple comprising multiple feature values. Let i denote the row index of nf and n represent the total number

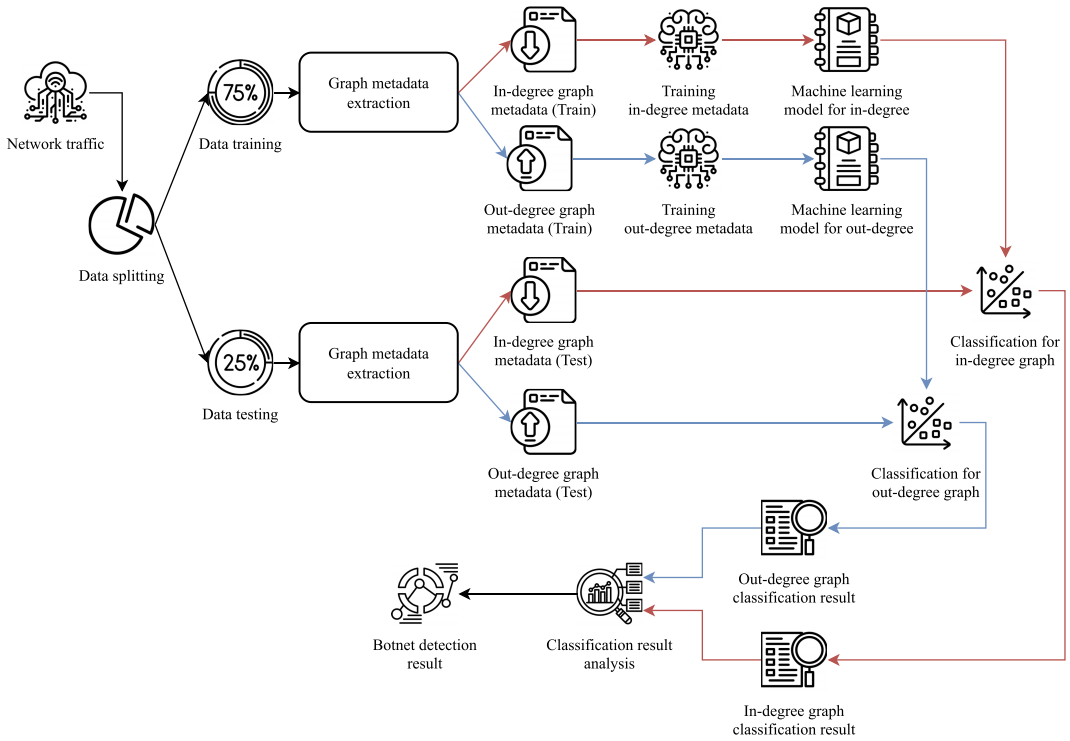

FIGURE 1. Proposed method.

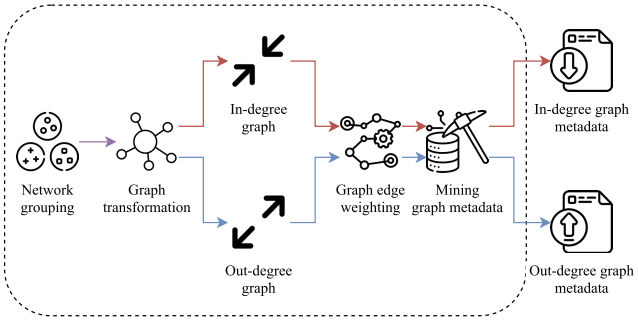

FIGURE 2. Detail of graph metadata extraction process.

of features within nf. Accordingly, the network flow can be formally described by (1).

$$nf_i = (x_{i1}, x_{i2}, \ldots, x_{in}) \tag{1}$$

Definition 2 (Collection of Network Flow): The collection of multiple network flows nf is represented as a vector, denoted as NF, where $NF = (nf_1, nf_2, \ldots, nf_m)$ with m representing the total number of network flows in the set NF. In this study, the network flow collection is analyzed in matrix form, as shown in (2).

$$NF = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} \tag{2}$$

Definition 3 (Features): The network flow (nf) collection contains diverse values across the same set of features, allowing for the formation of a feature-specific value set. The formula in (3) demonstrates how a set of values for a given feature is constructed. In this study, j represents the index or name of the feature within nf. For instance, the feature "StartTime," which corresponds to the first index in nf, can be denoted as f_1 or $f_{StartTime}$.

$$f_j = \begin{bmatrix} x_{1j} \\ x_{2j} \\ \vdots \\ x_{mj} \end{bmatrix} \tag{3}$$

A. DATA SPLITTING

The data splitting process commences by sorting the dataset based on the StartTime feature, ensuring that the entries are organized in chronological order. Once sorted, the dataset is divided into training and testing subsets according to a predetermined ratio, 75%:25%, with the index serving as the reference point. For example, in a dataset comprising 100 records, the first 75 entries are assigned to the training set, while the remaining 25 entries (from the 76th to the 100th) constitute the testing set. The training data from each subset is then combined to form a comprehensive set for the training phase. In contrast, the testing data is evaluated individually using machine learning techniques during the classification phase. Importantly, if the splitting process results in a test

set containing only one class, the respective subset is taken directly for testing purposes.

B. GRAPH METADATA EXTRACTION (FEATURE EXTRACTION)

This phase is dedicated to transforming the time-ordered network flow into a bidirectional graph. Initially, the network flow data is converted into a graph, from which metadata is subsequently extracted, incorporating 16 new features. The extraction of graph metadata involves four essential steps: activity grouping, graph transformation, edge weighting, and metadata mining. These processes operate in tandem to convert network activity into meaningful graph-based representations, thereby enabling more effective and comprehensive analysis.

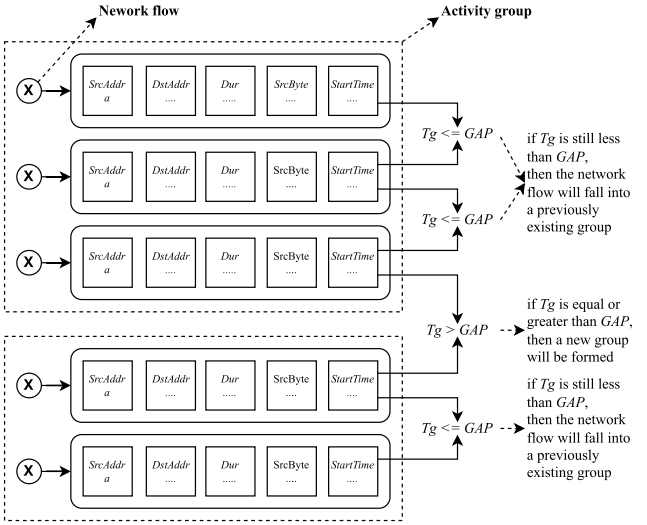

FIGURE 3. Illustration of the activity grouping process.

1) ACTIVITY GROUPING

Network flow recordings, encompassing millions of data points, exhibit a significant class imbalance due to the relatively low proportion of botnet activity, which constitutes less than 10% of the total data. Representing millions of network flows directly within a graph fails to adequately emphasize botnet attack activity. To address this issue, this research introduces a method that partitions the network flows into smaller subsets based on the source IP address and the temporal distance between network activities. In addition to creating these smaller subsets, this process is designed to capture the intensive and group-based nature of botnet attacks. By organizing network flows into activity groups, the primary characteristics of botnets can be more effectively analyzed. This approach ensures that the sparse botnet activity is not overshadowed by the vast majority of benign traffic, enabling more focused detection. Furthermore, by grouping network flows into activity groups, botnet attack patterns are represented in a more structured and organized manner. This grouping not only facilitates the analysis of botnet behaviors but also helps identify relationships between activities, such as distributed communication patterns or coordinated attack behaviors within a specific time frame, which are key characteristics of botnets.

The process begins by sorting the network flows according to their source IP addresses and the timestamps of each activity. A novel feature, termed Tg, is introduced to represent the temporal interval between consecutive network flow activities originating from the same IP address. Using this Tg, the network flows are grouped into subsets, referred to as activity groups, based on specific criteria. When Tg is greater than or equal to a predefined threshold GAP, a new activity group is created. Conversely, if Tg is smaller than this threshold, the network flow is assigned to an existing group. The value of GAP is determined through an analysis of the characteristics of botnet activity, as elaborated in Section IV-C. An illustration of the activity grouping process is presented in Fig. 3.

2) GRAPH TRANSFORMATION

Once the network flows are ordered and grouped into network groups, they are transformed into a directed graph $G = (V, E)$. In this representation, each IP address within a network group is modeled as a vertex v, such that V represents the set of IP addresses within the group. Formally, this is expressed as $V = v_1, v_2, \ldots, v_{nip}$, where nip denotes the total number of IP addresses in the group. A communication link between two IP addresses is represented by an edge e, and the set of all such edges within the group is denoted by E.

Moreover, each edge e may be assigned a weight w, where the weight of the edge from IP address A to IP address B is denoted by $w(e_{AB})$. This weight quantifies the communication relationship between v_A and v_B. Consequently, the graph G captures both the structural relationships and the weighted interactions among the IP addresses within the network group.

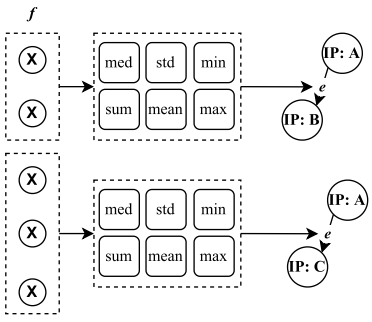

FIGURE 4. Detail of graph edge weighting process (for out degree graph).

3) GRAPH EDGE WEIGHTING

This process assigns a value w to each edge e in the previously constructed graph. The procedure for graph edge weighting is illustrated in Fig. 4. In this study, the 16 edge weights, which is taken by analyzing four distinct network

flow features. These features subjected to statistical analysis are: $feature_{SrcBytes}$ (the size of packets sent by the source), $feature_{Dur}$ (the duration of the activity), $feature_{Diff}$ (the time gap between consecutive activities), and $feature_{StartTime}$ (the time at which the activity occurs). Among the numerous features present in the network flow data, this study employed a manual feature selection process, resulting in the identification of four key features. These features—$feature_{SrcBytes}$, $feature_{Dur}$, $feature_{Diff}$, and $feature_{StartTime}$—were chosen for their suitability in capturing the primary characteristics of botnets, which rely on distributed attack intensity and become more distinguishable during activity grouping.

The edge weighting process, based on the 16 types of edge weighting, uses the statistical analysis of these four features to represent the communication characteristics between IP addresses in the graph more accurately.

1) **Degree:** number of connected devices. The in-degree refers to the number of devices that send messages to a given source address, while the out-degree represents the number of devices that receive messages from the source address.
2) **Intensity:** communication intensity. In-degree intensity refers to the level of communication directed from destination devices to the source within a group, whereas out-degree intensity indicates the level of communication originating from the source to destination devices within the group.
3) **Mean value of packet size:** average packet size. In the in-degree graph, the mean packet size represents the average size of the packets received, whereas in the out-degree graph, it reflects the average size of the packets sent.
4) **Standard deviation of packet size:** packet size data distribution value. A higher standard deviation in the in-degree graph indicates greater variability in the packet sizes received within a group. Similarly, in the out-degree graph, a larger standard deviation reflects greater diversity in the sizes of packets sent by sources within the group.
5) **Median value of packet size:** the median value of the packet size data. The median packet size in the in-degree graph signifies the middle value of the range of packet sizes received, whereas, in the out-degree graph, it represents the middle value of the packet sizes sent.
6) **Sum value of packet size:** total packet size. In the in-degree graph, the sum of packet sizes represents the total size of all packets received within the group. Correspondingly, in the out-degree graph, the sum denotes the total size of all packets sent by the sources within the group.
7) **Mean value of duration:** average activity duration. In the in-degree graph, the mean value of communication duration represents the average duration of incoming communication, whereas in the out-degree graph, it reflects the average communication time with the target.
8) **Standard deviation of duration:** activity duration distribution value. A high standard deviation of duration in the in-degree graph indicates significant variability in the duration of communication received within a group. In contrast, in the out-degree graph, the standard deviation of duration reflects the variations in communication duration with the target. Since botnet activity typically originates from the same source and has the same objective, the variation in communication duration between the bot and the target is expected to be minimal.
9) **Median value of duration:** the median value of the activity duration. In the in-degree graph, the median duration represents the middle value of all received communication durations. Similarly, the median duration in the out-degree graph indicates the middle value of all communication durations with the target.
10) **Sum value of duration:** the total duration of the communication. The in-degree graph illustrates the total time during which communication was received, while the out-degree graph represents the total duration of communication initiated with the target.
11) **Mean value of time gap:** average time gap. The in-degree graph reflects the average time gap between nodes and their respective listeners, while the out-degree graph represents the average time gap when nodes are acting as senders.
12) **Standard deviation of time gap:** time gap distribution value. To assess the variations in time gaps, the in-degree provides insight into the time differences between nodes when they are listeners, whereas the out-degree highlights the variations when nodes are senders.
13) **Median value of time gap:** the median value of the time gap. The median value of the time gap in a network group is shown in the in-degree graph when the node is a listener and in the out-degree graph when the node is a sender.
14) **Sum value of time gap:** total time gap. The sum of all communication time gaps within a network group is represented in the in-degree graph when nodes are listeners, and in the out-degree graph when they are senders.
15) **Activity group start time:** describes the activity start time of a network group when it is a listener and sender.
16) **Activity group end time:** describes the end time of activity from a network group when it is a listener and sender.

The four features analyzed for graph edge weighting are computed using statistical methods. If each feature in the network flow (*SrcBytes*, *Dur*, *Diff*, *StartTime*) is represented by j, the statistical calculations for the sum ($S(f_j)$), mean ($\bar{f}_j$), standard deviation ($\sigma(f_j)$), median ($\tilde{f}_j$), minimum ($min(f_j)$), and maximum ($max(f_j)$) values are described

in (4-9), respectively.

$$S(f_j) = \sum_{i=1}^{m} x_{ij} \tag{4}$$

$$\bar{f_j} = \frac{1}{m} S(f_j) \tag{5}$$

$$\sigma(f_j) = \sqrt{\frac{1}{m-1} \sum_{i=1}^{m} (x_{ij} - (\bar{f_j})^2} \tag{6}$$

$$\tilde{f_j} = \begin{cases} x_{(\frac{m}{2})j} & \text{if } m \text{ is odd} \\ \frac{1}{2}\left(x_{(\frac{m}{2})j} + x_{(\frac{m}{2}+1)j}\right) & \text{if } m \text{ is even} \end{cases} \tag{7}$$

$$min(f_j) = \min(x_{1j}, x_{2j}, \ldots, x_{mj}) \tag{8}$$

$$max(f_j) = \max(x_{1j}, x_{2j}, \ldots, x_{mj}) \tag{9}$$

4) MINING GRAPH METADATA

The previous processes result in the construction of a directed graph with 16 weights assigned to the edges. From this graph, directed graph metadata are extracted, encompassing both the 16 edge weights and the associated IP addresses. The metadata extraction process yields tabular data, which is subsequently used for classification in machine learning models. The output of this process consists of two distinct types of metadata: in-degree graph metadata and out-degree graph metadata. This research separates the in-degree and out-degree metadata because not all nodes in the graph possess both in-degree and out-degree information, leading to differences in the characteristics of the data. This separation is essential for analyzing distinct communication patterns between botnet activities that act as attackers and those that receive commands. Additionally, it simplifies the classification process in machine learning models by providing two specific and relevant datasets for further analysis.

C. ACTIVITY GROUP CLASSIFICATION WITH MACHINE LEARNING ALGORITHM

This phase is responsible for classifying the data based on the output from the previous step, which consists of in-degree and out-degree activity group metadata graphs. A total of 8 machine learning algorithms are implemented in this process: Decision Tree (DT), Logistic Regression (LR), Naive Bayes (NB), Random Forest (RF), AdaBoost (AB), Extra Tree (ET), XGBoost (XGB), and k-Nearest Neighbors (k-NN). The classification phase begins with the training of the machine learning models using the processed training data from the previous phase. The training is performed separately for the in-degree and out-degree metadata, resulting in two training models for each algorithm.

Next, the testing data is classified based on the features extracted from the graph metadata, utilizing the appropriate training model for each algorithm, depending on whether the graph is in-degree or out-degree. This graph-based classification generates predictions for each IP address within the graph according to its respective group. The classification results are summarized into a dataset that includes a confidence score for the predictions. The confidence score represents the ratio of predicted botnet IP addresses. If the total number of data points associated with IP address A is $count(A)$, and the number of data points predicted as botnet-related is $predictedBot(A)$, the confidence score (C) is calculated as described in (10).

This study utilizes machine learning during the classification phase for several reasons, supported by the alignment of data characteristics and the optimal performance observed in prior research [14], [38]. The features extracted from network flow metadata are relatively small in scale and complexity, making traditional machine learning algorithms particularly well-suited for classification tasks. Models such as Decision Trees, Logistic Regression, and Random Forest effectively manage structured tabular data and provide interpretable results, rendering them appropriate for this problem. The goal of this study is to develop a detection model characterized by low complexity and lightweight features, which will enable more efficient forensic analysis of past attacks. Since the data is processed not as individual network flows but as aggregated groups, this approach promotes a more scalable and practical solution for real-time analysis.

$$C(A) = predictedBot(A)/count(A) \tag{10}$$

D. CLASSIFICATION RESULT ANALYSIS

In this phase, the confidence scores from the classification of the in-degree and out-degree graphs are combined. Botnets primarily function as attackers, and their attack characteristics can often be identified through out-degree metadata. However, a thorough analysis requires examination of both in-degree and out-degree graphs. This dual approach is essential, as botnets frequently engage in receiving communications, such as during TCP handshakes or when commands are dispatched by botmasters via command and control (C&C) channels [3], [14]. By integrating both in-degree and out-degree metadata, the analysis becomes more comprehensive, especially in pinpointing bots that actively serve as intermediaries for C&C communication.

Before the combination, a weighting process is necessary because the in-degree and out-degree C values have different levels of influence on determining the final prediction. The out-degree data typically provides a clearer indication of botnet behavior, as its primary characteristic is intense outgoing traffic, particularly in the context of attacks [37]. Nevertheless, in-degree data also contributes to the prediction, so the results of the in-degree graph metadata classification are included with a different weight.

The determination of the weights, w_{in} and w_{out}, is done adaptively based on the comparison of the average C values from the in-degree and out-degree classification results. If the average C value for the in-degree data is $\bar{C_{in}}$ and the average C value for the out-degree data is $\bar{C_{out}}$, the weights w_{in} and w_{out} are assigned in proportion to $\bar{C_{in}} : \bar{C_{out}}$. Each IP address is then analyzed based on its in-degree and out-degree

confidence scores, which are combined to produce the final confidence score (C_{fin}). If w_{in} and w_{out} are the weights for in-degree and out-degree data, respectively, the final confidence score for an IP address A is calculated as shown in Algorithm 1.

The results of the C_{fin} calculation are further analyzed to determine whether the IP address is classified as a botnet or normal. This determination is based on a threshold value T: if C_{fin} exceeds T, the IP address is identified as a botnet; otherwise, it is classified as normal. The optimal threshold value T is established through iterative testing, with T values ranging from 40 to 60, in increments of 5. The results of these threshold tests are presented in Section IV-E2.

$$\text{Inference}= \begin{cases} \text{Botnet} & \text{if } C_{fin}(A) \geq T \\ \text{Normal} & \text{otherwise} \end{cases} \tag{11}$$

Algorithm 1 Final Confidence Score Calculation

Input: Activity A
Output: Final Confidence Score $C_{fin}(A)$

$predictedBot_{in}(A)$: predicted number of inbound bot actions for A
$predictedBot_{out}(A)$: predicted number of outbound bot actions for A
w_{in} and w_{out}: weights for inbound and outbound actions, respectively
$count_{in}(A)$ and $count_{out}(A)$: total number of inbound and outbound actions for A

$numerator \leftarrow (predictedBot_{in}(A) \times w_{in}) + (predictedBot_{out}(A) \times w_{out})$

$denominator \leftarrow (count_{in}(A) \times w_{in}) + (count_{out}(A) \times w_{out})$

if $denominator = 0$ **then**
  return 0
end if

$C_{fin}(A) \leftarrow numerator / denominator$

return $C_{fin}(A)$

IV. RESULT AND DISCUSSION

The experiments were conducted using applications developed in Python 3.10, executed on a system equipped with a 12th Gen Intel(R) Core(TM) i7-12700 processor (20 CPUs, 2.1GHz) and 32 GB of RAM. This section presents the experimental results, beginning with a description of the dataset, followed by the data splitting process, time-gap analysis, metadata graph classification results (in-degree and out-degree), botnet detection outcomes, ablation study, execution time analysis, and a comparative analysis of the proposed approach. The model's performance is assessed using five evaluation metrics: accuracy (acc), precision (pre), recall (rec), F1-score ($f1$), and execution time.

A. DATASET

This experiment utilizes three distinct types of datasets: CTU-13 [35], NCC [36], and NCC-2 [37]. These datasets, provided in the bidirectional network flow (.binetflow) format, represent different attack characteristics: sporadic (CTU-13), periodic (NCC), and simultaneous (NCC-2). Each dataset contains a varying number of sub-datasets: CTU-13 and NCC both contain 13 sub-datasets, referred to as scenarios, while NCC-2 comprises 3 sub-datasets, referred to as sensors. In total, 29 sub-datasets were tested in this study. Details of each dataset, including the total recording duration and network flow, are provided in Table 1.

Each sub-dataset includes a different number of attacking bot IPs, ranging from one to ten bot IPs. The calculation of bot IPs disregards the specific botnet type. Unlike CTU-13 and NCC, which feature a single botnet type per sub-dataset, NCC-2 includes multiple types of botnet attacks within a single sub-dataset. For example, in NCC-2 sensor 1, five types of botnets utilize the same Bot IP, '147.32.84.165.' If bot IPs were further differentiated based on the type of botnet, NCC-2 sensor 1 would contain 27 attackers [37]. However, this study focuses exclusively on bot IPs, resulting in a total of ten bot IPs analyzed across the datasets. The experiments are conducted by analyzing each sub-dataset containing recorded network flows instead of processing each incoming network flow in real-time. Decisions are made only after evaluating all data within the sub-dataset, rather than for each network flow as it arrives. This method aligns with the primary objective of the detection model, which is designed for post-attack forensic analysis. It focuses on examining the characteristics of botnet attacks to enhance the anticipation of future threats.

B. DATA SPLITTING

In this research, data labeling was conducted prior to the data splitting phase to ensure that the proportion of botnet and normal data remained balanced in both the training and testing datasets. Each network flow associated with an IP address listed in Tables 2-4 was labeled as an IP Botnet. The data splitting phase was performed based on the time at which the network flow occurred. This time-based splitting approach was taken because the grouping phase relies on the temporal appearance of network flows. Therefore, the time interval between network activities serves as the foundation for forming network traffic activity groups.

However, splitting the data based on the time of activity increases the likelihood of situations where a bot IP may not be present in the testing data. As shown in Tables 2 and 3, in CTU-13 and NCC scenario 11, there are only two bot IPs involved in the attack, as bot IP '147.32.84.192' appears only at the beginning of the botnet activity.

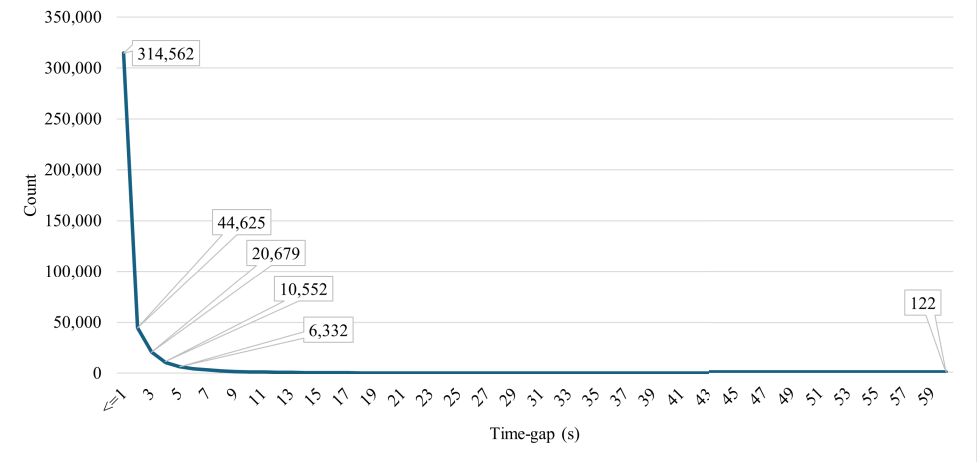

FIGURE 5. The amount of botnet network flow data in each time-gap.

TABLE 1. Detail of dataset.

Dataset	Sub Dataset	Dur (h)	Total NF	Botnet NF	Normal NF
D1	1	6.15	2,824,636	40,961	2,783,675
	2	4.21	1,808,122	20,941	1,787,181
	3	66.85	4,710,638	26,822	4,683,816
	4	4.21	1,121,076	2,580	1,118,496
	5	11.63	129,832	901	128,931
	6	2.18	558,919	4,630	554,289
	7	0.38	114,077	63	114,014
	8	19.50	2,954,230	6,127	2,948,103
	9	5.18	2,087,508	184,987	1,902,521
	10	4.75	1,309,791	106,352	1,203,439
	11	0.26	107,251	8,164	99,087
	12	1.21	325,471	2,168	323,303
	13	16.36	1,925,149	40,003	1,885,146
D2	1	8.00	2,112,224	23,000	2,089,224
	2	8.00	1,465,182	24,000	1,441,182
	3	8.00	2,905,611	2,000	2,903,611
	4	8.00	724,388	11,000	713,388
	5	8.00	92,917	19,000	73,917
	6	8.00	512,021	6,000	506,021
	7	8.00	83,473	9,000	74,473
	8	8.00	2,871,217	14,000	2,857,217
	9	8.00	1,573,304	220,000	1,353,304
	10	8.00	984,369	60,000	924,369
	11	8.00	30,964	12,000	18,964
	12	8.00	274,186	9,000	265,186
	13	8.00	1,876,489	19,000	1,857,489
D3	1	8.00	4,895,158	146,000	4,749,158
	2	8.00	5,998,133	364,000	5,634,133
	3	8.00	3,885,792	294,000	3,591,792

D1: CTU-13 (sporadic); D2: NCC (periodic); D3: NCC-2 (simultaneous)

TABLE 2. Data splitting result (CTU-13).

Sub	Botnet IP		Botnet IP Addresses
	Train	Test	
1	1	1	147.32.84.165
2	1	1	147.32.84.166
3	1	1	147.32.84.167
4	1	1	147.32.84.168
5	1	1	147.32.84.169
6	1	1	147.32.84.170
7	1	1	147.32.84.171
8	1	1	147.32.84.172
9	10	10	147.32.84.165, 147.32.84.191, 147.32.84.192, 147.32.84.193, 147.32.84.204, 147.32.84.205, 147.32.84.206, 147.32.84.207, 147.32.84.208, 147.32.84.209
10	10	10	147.32.84.165, 147.32.84.191, 147.32.84.192, 147.32.84.193, 147.32.84.204, 147.32.84.205, 147.32.84.206, 147.32.84.207, 147.32.84.208, 147.32.84.210
11	3	2	147.32.84.165, 147.32.84.191, 147.32.84.192
12	3	3	147.32.84.165, 147.32.84.191, 147.32.84.193
13	1	1	147.32.84.165

C. NETWORK FLOW TIME-GAP ANALYSIS

A time gap refers to the time difference between two sequentially connected network traffic events. This research analyzes the characteristics of botnet network flows in the training data to determine the appropriate *GAP* value. The analysis collects unique *Tg* values from botnet-specific network flows (*NF*) in seconds. Fig. 5 presents the results, illustrating variations in *Tg* values ranging from $Tg \leq 1$ to $Tg = 60$. Beyond $Tg > 60$, the trend in the graph remains constant, as it continues to decline.

As depicted in Fig. 5, the majority of *Tg* values between botnet attacks are "≤ 1," with a total of 314,562 occurrences across all datasets (CTU-13, NCC, and NCC-2). Based on this data, it can be concluded that the interval between botnet attacks is typically no more than 1 second. Therefore, the *GAP* value is set to 1. This means that if the time difference between activities in a network flow (*NF*) exceeds 1 second, it will be classified as belonging to a different activity group.

D. ACTIVITY GROUP CLASSIFICATION RESULT

The output of the time-gap analysis is used to group each network flow based on the identified time gaps. Tables 5 (training) and 6 (testing) provide the number of activity groups in each dataset used in this study for both in-degree and out-degree graphs. Since botnets launch attacks via communication channels, their activity can be detected by analyzing both out-degree and in-degree metadata.

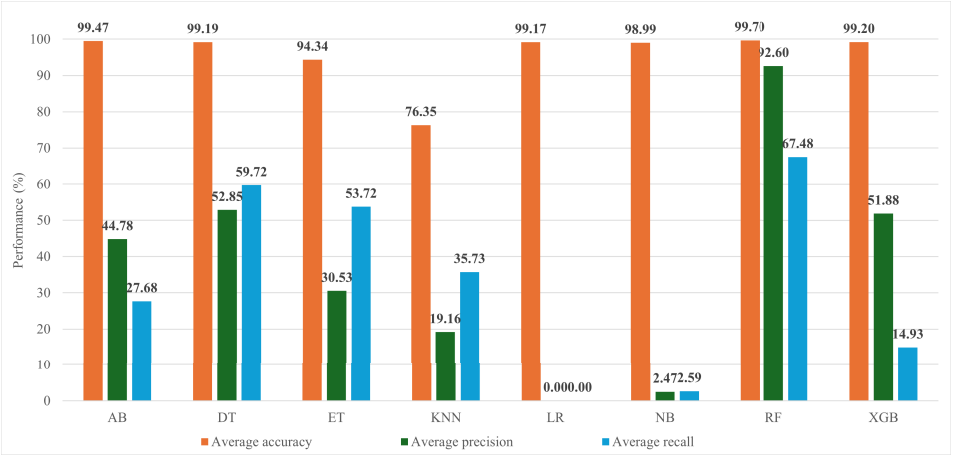

FIGURE 6. Out-degree activity group metadata classification result with CTU-13.

TABLE 3. Data splitting result (NCC).

Sub	Botnet IP		Botnet IP Addresses
	Train	Test	
1	1	1	147.32.84.165
2	1	1	147.32.84.166
3	1	1	147.32.84.167
4	1	1	147.32.84.168
5	1	1	147.32.84.169
6	1	1	147.32.84.170
7	1	1	147.32.84.171
8	1	1	147.32.84.172
9	10	10	147.32.84.165, 147.32.84.191, 147.32.84.192, 147.32.84.193, 147.32.84.204, 147.32.84.205, 147.32.84.206, 147.32.84.207, 147.32.84.208, 147.32.84.209
10	10	10	147.32.84.165, 147.32.84.191, 147.32.84.192, 147.32.84.193, 147.32.84.204, 147.32.84.205, 147.32.84.206, 147.32.84.207, 147.32.84.208, 147.32.84.210
11	3	2	147.32.84.165, 147.32.84.191, 147.32.84.192
12	3	3	147.32.84.165, 147.32.84.191, 147.32.84.193
13	1	1	147.32.84.165

TABLE 4. Data splitting result (NCC-2).

Sub	Botnet IP		Botnet IP Addresses
	Train	Test	
1	10	10	147.32.84.165, 147.32.84.191, 147.32.84.192, 147.32.84.193, 147.32.84.204, 147.32.84.205, 147.32.84.206, 147.32.84.207, 147.32.84.208, 147.32.84.209
2	10	10	147.32.84.165, 147.32.84.191, 147.32.84.192, 147.32.84.193, 147.32.84.204, 147.32.84.205, 147.32.84.206, 147.32.84.207, 147.32.84.208, 147.32.84.209
3	10	10	147.32.84.165, 147.32.84.191, 147.32.84.192, 147.32.84.193, 147.32.84.204, 147.32.84.205, 147.32.84.206, 147.32.84.207, 147.32.84.208, 147.32.84.209

TABLE 5. Number of activity group in data training.

Dataset	Sub-Dataset	In-degree		Out-degree	
		Normal	Botnet	Normal	Botnet
D1	1	1,185,581	163	911,726	2,396
	2	782,262	59	602,516	1,197
	3	1,464,730	3,049	1,107,178	8,393
	4	385,005	184	233,483	511
	5	49,412	6	31,792	158
	6	177,757	33	108,434	531
	7	44,092	4	25,618	9
	8	1,052,720	222	762,134	1,368
	9	808,063	430	530,799	9,971
	10	429,020	247	272,418	973
	11	43,782	11	29,284	25
	12	120,548	2,773	74,575	134
	13	837,490	294	627,057	13,219
D2	1	1,101,553	219	896,614	4,934
	2	758,770	55	625,208	4,813
	3	1,118,488	3,815	898,728	697
	4	342,122	579	255,936	4,349
	5	60,691	7	42,534	4,884
	6	245,328	60	188,945	3,078
	7	54,942	-	43,776	3,997
	8	1,114,076	277	876,919	4,659
	9	754,406	422	486,684	49,110
	10	487,458	5,290	340,645	31,496
	11	21,685	438	13,362	7,111
	12	149,499	7,385	122,202	2,142
	13	849,036	277	678,838	4,859
D3	1	1,806,854	7,925	1,306,262	30,686
	2	2,394,325	684	1,721,439	43,007
	3	1,365,611	5,625	822,675	45,756

D1: CTU-13 (sporadic); D2: NCC (periodic); D3: NCC-2 (simultaneous);

As previously discussed, out-degree metadata is derived from visualizing network activity groups into a directed graph, where out-degree represents communication activity from botnet nodes to external targets. This enables the model to identify bots based on the intensity of their communication, which can reveal their role in attacks.

In addition to analyzing outgoing communication, this study also investigates the behavior of bots when receiving communication through in-degree graph analysis. Directed graph metadata within activity groups is classified using eight different machine learning algorithms to distinguish suspected attack activities from normal ones. This section presents an analysis of the classification results for both

TABLE 6. Number of activity group in data testing.

Dataset	Sub-Dataset	In-degree		Out-degree	
		Normal	Botnet	Normal	Botnet
D1	1	424,538	108	324,636	807
	2	261,038	20	206,144	852
	3	653,224	1,972	500,932	3,085
	4	126,088	16	78,970	105
	5	14,899	5	9,639	153
	6	61,364	22	37,573	210
	7	12,732	-	7,861	6
	8	376,137	77	288,167	445
	9	219,345	262	150,709	2,963
	10	150,156	216	96,273	584
	11	14,241	3	9,106	16
	12	39,058	1,469	25,191	135
	13	271,762	37	203,516	7,977
D2	1	369,594	61	301,867	1,696
	2	254,561	19	210,794	1,675
	3	379,385	1,309	304,973	247
	4	115,554	163	87,126	1,459
	5	20,470	2	14,504	1,668
	6	83,119	20	64,763	1,081
	7	18,582	-	14,934	1,308
	8	375,528	109	296,206	1,589
	9	255,022	135	164,656	16,290
	10	164,835	1,785	116,219	10,506
	11	7,269	141	4,472	2,346
	12	50,717	2,526	41,854	697
	13	286,420	77	229,002	1,695
D3	1	672,649	3,849	502,711	10,384
	2	947,499	408	704,237	14,344
	3	447,467	2,003	294,300	15,312

D1: CTU-13 (sporadic); D2: NCC (periodic); D3: NCC-2 (simultaneous)

out-degree and in-degree metadata. The classification performance is evaluated using three key metrics: accuracy (*acc*), precision (*pre*), and recall (*rec*).

1) OUT-DEGREE CLASSIFICATION RESULT

The performance of the eight algorithms in classifying out-degree metadata on the CTU-13, NCC, and NCC-2 datasets is presented in Figs. 6 through 8, respectively. Overall, two algorithms—Logistic Regression (LR) and Naive Bayes (NB)—underperform compared to the others. While these algorithms achieve optimal *acc* values, they fail to effectively detect botnet attack activity groups. The classification results indicate that LR only successfully detected botnet activity in sub-datasets characterized by high attack intensity, specifically in NCC (scenario 10) and NCC-2 (sensors 1 and 2). In these sub-datasets, multiple bots (ten in total) launched Distributed Denial of Service (DDoS) attacks. DDoS attacks, which aim to overwhelm targets with traffic, tend to increase the intensity of the attacks, particularly when carried out by many attacking bots [24], [39].

Regarding the recall (*rec*) metric, two algorithms—*k*-NN and Extra Trees (ET)—demonstrate superior performance, achieving maximum performance on both the NCC and NCC-2 datasets. With a well-designed grouping process, the proposed method effectively identified the intensity of attacks on these datasets, though there is a higher likelihood of false alarms. On average, the four decision tree-based algorithms (Random Forest (RF), Decision Tree (DT), XGBoost (XGB), and AdaBoost (AB)) perform consistently well. Among these, RF achieves the most optimal average performance across all datasets, regardless of their different characteristics.

However, CTU-13 (scenario 7) proves to be the most challenging sub-dataset to classify. Several algorithms—XGB, LR, RF, *k*-NN, ET, and AB—fail to detect botnet activity groups in this scenario. The difficulty in detecting botnet activity groups in this sub-dataset is likely due to the low intensity of the attack, which involved only a single bot.

2) IN-DEGREE CLASSIFICATION RESULT

The classification results for the in-degree graph data are presented in Fig.s 9 through 11. Experiments using the three different datasets demonstrate that the model performs well in terms of the accuracy (*acc*) metric. On average, except for Naive Bayes (NB), the models achieve an accuracy of 99%. However, it is important to note that *acc* alone does not provide a comprehensive measure of model performance due to the imbalanced nature of the data.

In contrast, as shown in Figs. 9 through 11, the precision (*pre*) and recall (*rec*) metrics do not perform as well as *acc*. The performance of NB, for instance, is influenced by the characteristics of botnet activity groups within the in-degree metadata, achieving an average *rec* value of over 90%, but with lower *acc* and *pre* scores. A comparison of *rec* with *acc* and *pre* reveals that NB is highly sensitive, but prone to generating a high number of false alarms.

Random Forest (RF) maintains consistent performance in detecting botnet activity within the in-degree graph data, though its performance declines in scenarios 1, 11, 2, and 7. These results suggest that in-degree graph data can effectively characterize botnet attack activity groups. Several machine learning models tested in this study yielded optimal recall values when classifying in-degree data. This indicates that botnet activity can also be detected based on the behavior of the botnet when acting as a listener.

The combination of in-degree and out-degree detection results is particularly compelling, as each algorithm exhibits varying performance when classifying the two types of metadata. This variation underscores the potential advantage of integrating in-degree and out-degree classification to improve the overall detection of botnet activity.

E. BOTNET DETECTION RESULT

This research focuses on detecting IP addresses suspected of being part of a botnet. The classification results of the activity groups serve as the foundation for calculating the confidence score (C) for each IP address in both the in-degree and out-degree graphs, using (10). The confidence scores from the two directed graphs are then combined to derive the final detection results by calculating the final confidence score (C_{fin}) through (1). A key component in calculating C_{fin} is the determination of appropriate weights for in-degree and out-degree scores. The process for determining the optimal weights is discussed in sub-section IV-E1.

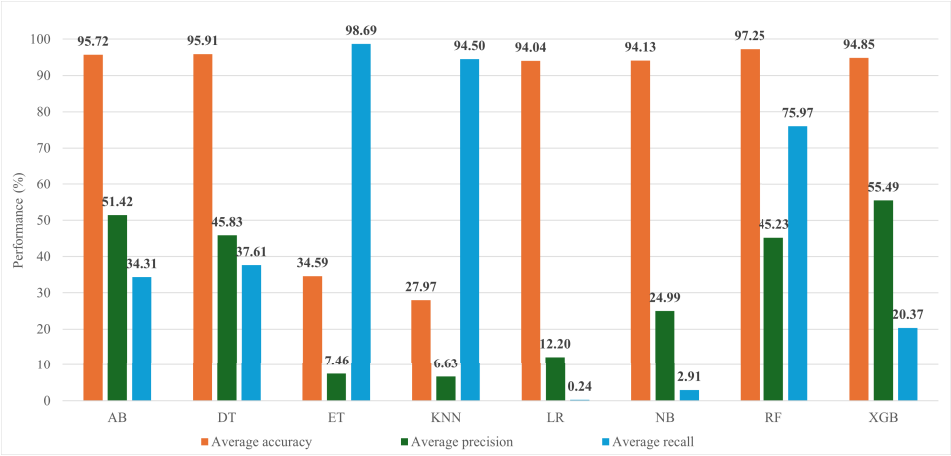

FIGURE 7. Out-degree activity group metadata classification result with NCC.

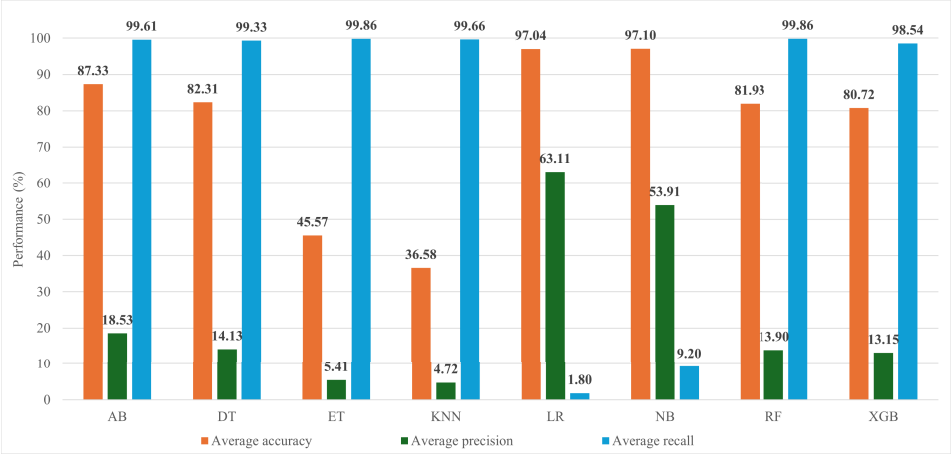

FIGURE 8. Out-degree activity group metadata classification result with NCC-2.

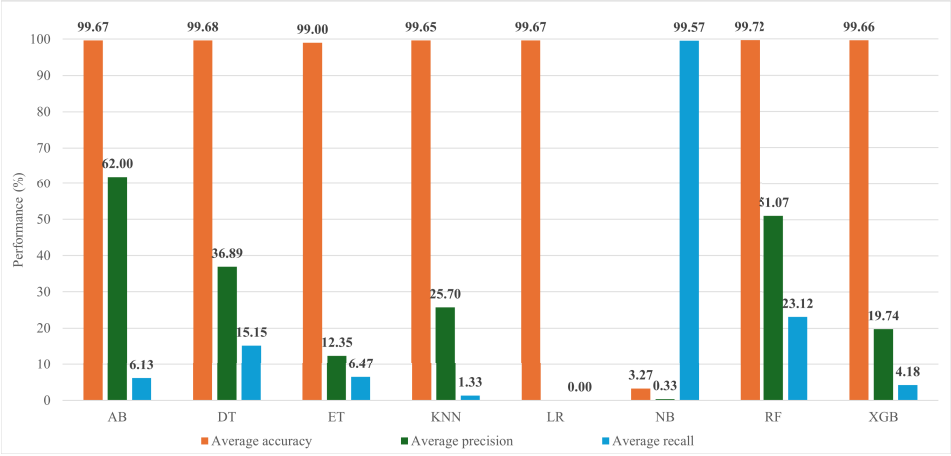

FIGURE 9. In-degree activity group metadata classification result with CTU-13.

Additionally, the analysis of the threshold value (T), which is used to make final detection decisions based on (11), is outlined in Sub-section IV-E2. The following subsections present a detailed analysis of the detection

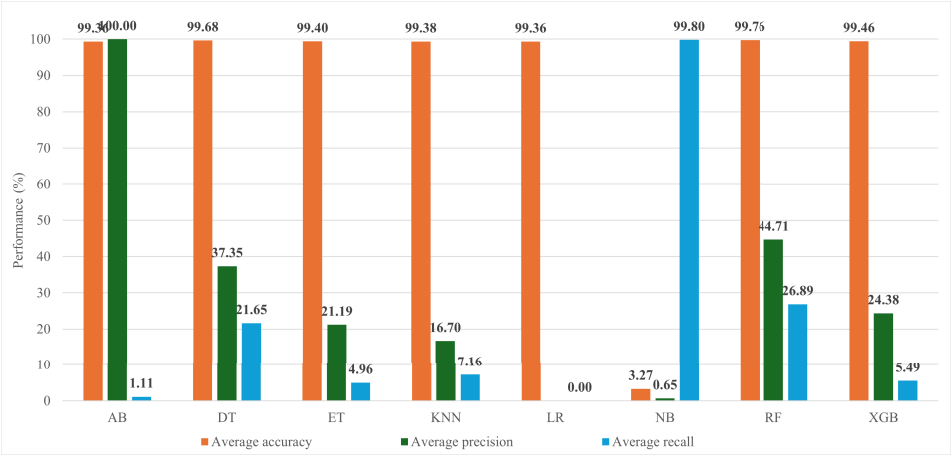

FIGURE 10. In-degree activity group metadata classification result with NCC.

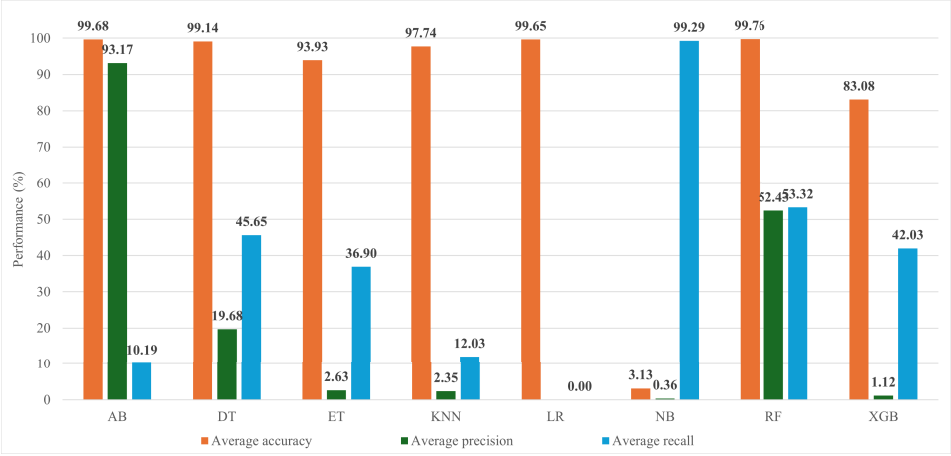

FIGURE 11. In-degree activity group metadata classification result with NCC-2.

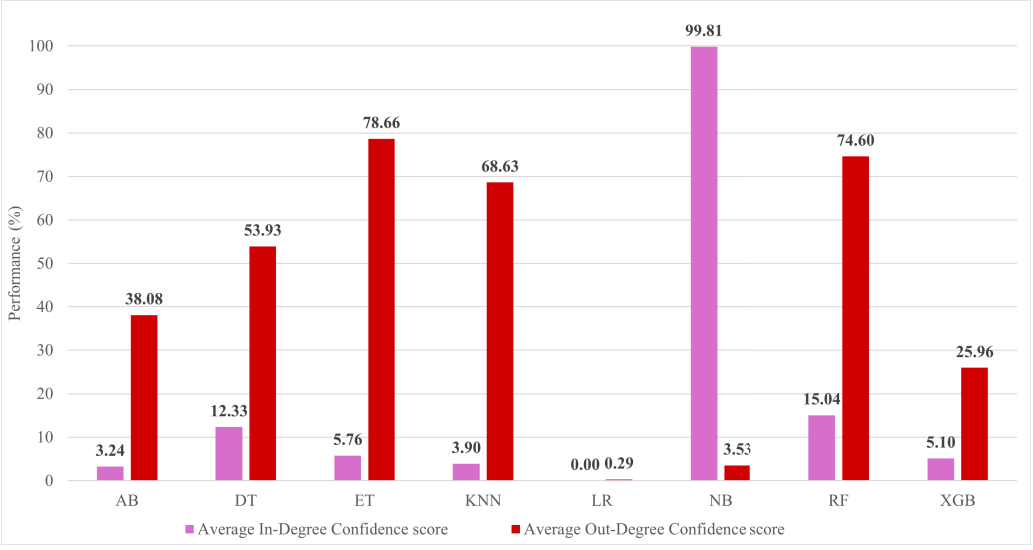

FIGURE 12. Average of in-degree & out-degree confidence score in every machine learning algorithms.

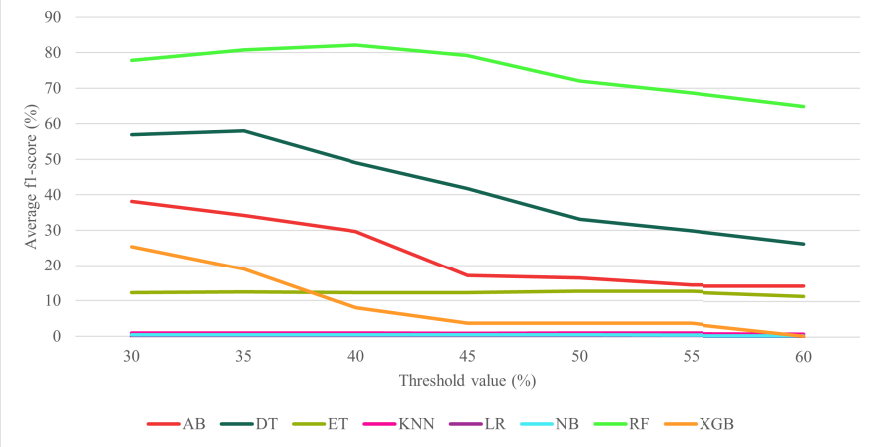

FIGURE 13. Detection performance in every machine learning algorithm with different T.

model's performance, considering the chosen weights and T values, and evaluating the results across four key metrics: accuracy (*acc*), precision (*pre*), recall (*rec*), and F1-score ($f1$).

1) WEIGHT ANALYSIS

Figs. 6 through 11 demonstrate that the performance of the algorithms in detecting in-degree and out-degree metadata varies significantly. Therefore, rather than relying on a fixed pair of weight values for each algorithm, a more effective approach involves using dynamic weighting, based on the detection performance of each model on in-degree and out-degree metadata.

In this study, the weights for in-degree and out-degree scores were determined by comparing the average confidence scores for each algorithm in both in-degree and out-degree graphs. This comparison is illustrated in Fig. 12, which shows that the average confidence score for out-degree is not consistently higher than for in-degree. Notably, NB is the only algorithm that performs better at detecting botnets using in-degree rather than out-degree metadata. Based on the average value of C and w values, it can be concluded that the in-degree metadata exhibits independent features. This conclusion aligns with the primary characteristic of Naive Bayes (NB), which is most optimal for detecting data with independent features compared to other algorithms used in this study [40]. NB is the only algorithm with the most optimal average C value compared to other algorithms, excluding LR, which exhibits below-average performance. By simplifying the comparison of the values in Fig. 12, the weights w_{in} and w_{out} are derived. These values are presented in Table 7 and are subsequently used to calculate the final confidence score (C_{fin}).

2) THRESHOLD

The result of the C_{fin} calculation is an integer ranging from 0 to 100, representing the model's confidence in classifying an IP address as a botnet. To finalize the detection, the model uses a threshold value (T) determined through heuristic testing, starting with values between 40 and 60, in increments of 5. Each IP address is then processed with the corresponding T value and the weights w_{in} and w_{out}, as specified in Table 7. The prediction results are compared with the actual labels, and the F1-score ($f1$) is calculated for each T value across different algorithms. The detection results are then presented as an $f1$ graph to illustrate the performance trend for each T value.

TABLE 7. Value of w_{in} and w_{out} in each algorithms.

Algorithm	w_{in}	w_{out}
AB	3	38
DT	12	53
ET	5	78
k-NN	3	68
LR	1	1
NB	33	1
RF	15	74
XGB	1	5

In the initial testing phase, most algorithms achieved their highest $f1$ performance at $T = 40$. Based on these findings, further tests were conducted with $T = 30$ and $T = 35$ to pinpoint where peak performance occurred. The final tests indicated a decline in performance for the Random Forest (RF) algorithm when T was set below 40, for Decision Tree (DT) at T below 35, and for AdaBoost (AB) at T below 35. The experimental results, as shown in Fig. 13, demonstrate that a lower T value does not necessarily lead to improved performance.

Analyzing the performance trend of each model across varying T values revealed that the optimal threshold was $T = 40$. This threshold was selected as RF achieved the highest performance at $T = 40$, although not significantly higher than the other algorithms. Thus, $T = 40$ was chosen as the most suitable threshold for botnet IP address detection, following (11). The detection results were then compared

with actual data, and the model's performance was evaluated using four key metrics: accuracy (*acc*), precision (*pre*), recall (*rec*), and F1-score ($f1$).

3) AVERAGE PERFORMANCE OF EACH MACHINE LEARNING ALGORITHM

Each algorithm yields varying results when detecting botnet IP addresses with the parameters $w_{in} = 15$, $w_{out} = 74$, and $T = 40$. The RF, DT, ET, k-Nearest Neighbors (k-NN), and NB algorithms all demonstrated optimal performance, with average recall (*rec*) values exceeding 70%. However, DT, ET, k-NN, and NB still struggled with the issue of false alarms. Only RF effectively addressed the false alarm problem, achieving an average precision (*pre*) of 80.90%.

Furthermore, all algorithms produced strong accuracy (*acc*) values above 95% (see Table 14). However, given the imbalanced nature of the botnet dataset, a high *acc* value alone is not a reliable indicator of the model's effectiveness in detecting botnets. Based on a comprehensive analysis of the performance metrics—including accuracy, precision, recall, and F1-score—RF emerges as the most optimal algorithm when using the combination of $w_{in} = 15$, $w_{out} = 74$, and $T = 40$.

4) BEST PERFORMANCE OF PROPOSED METHOD

Based on the previous analysis, the Random Forest (RF) algorithm demonstrates the most stable performance compared to other algorithms when using the parameters $w_{in} = 15$, $w_{out} = 74$, and $T = 40$. Despite this stability, certain botnet IP addresses were not detected, specifically in scenarios 7 of the CTU-13 and NCC datasets (see Table 8). Both of these scenarios involve HTTP attack activity executed by the Sogou botnet [35]. The botnet activity in scenario 7 of CTU-13 and NCC is the least active compared to other sub-datasets. In general, the number of botnet activities in a sub-dataset correlates with the intensity of the attack. Since the proposed method relies heavily on analyzing the intensity of activity groups, low-intensity attacks are more challenging to detect.

The proposed model also performs well in terms of precision (*pre*). Of the 29 sub-datasets tested, 22 achieved a precision value of 100%, indicating that the model effectively reduces the likelihood of false alarms. However, the method struggles more with simultaneous botnet attacks, such as those in NCC-2, which is reflected in its lower *pre* values. Although precision in NCC-2 is below average, the number of false alarms generated is relatively low in percentage terms. For instance, in NCC-2 sensor 1, the total number of false positives (FP) is 388, which represents only 0.11% of the total normal IP addresses. The precision remains low because the ratio of false positives is significantly higher than that of true positives (TP).

The model's performance is significantly affected by the imbalanced nature of the dataset, as illustrated in Table 8. For instance, the class comparison can reach a ratio of 10:361,566 in sub-dataset 1 NCC-2; nonetheless, the model

TABLE 8. Proposed method best performance (RF, $w_{in} = 15$, $w_{out} = 74$ and $T = 40$) with CTU-13, NCC and NCC-2.

D	Sub	TN	FP	FN	TP	$acc(\%)$	$pre(\%)$	$rec(\%)$	$f1(\%)$
D1	1	268,214	0	0	1	100	100	100	100
	2	163,417	1	0	1	100	50.00	100	66.67
	3	220,353	0	0	1	100	100	100	100
	4	66,020	0	0	1	100	100	100	100
	5	10,432	0	0	1	100	100	100	100
	6	38,504	0	0	1	100	100	100	100
	7	9,567	0	1	0	99.99	0.00	0.00	0.00
	8	141,914	0	0	1	100	100	100	100
	9	134,484	2	0	10	100	83.33	100	90.91
	10	84,929	0	0	10	100	100	100	100
	11	11,622	0	0	2	100	100	100	100
	12	26,661	0	0	3	100	100	100	100
	13	113,632	0	0	1	100	100	100	100
D2	1	183,493	0	0	1	100	100	100	100
	2	129,464	0	0	1	100	100	100	100
	3	139,893	0	0	1	100	100	100	100
	4	43,713	0	0	1	100	100	100	100
	5	6,351	0	0	1	100	100	100	100
	6	30,865	0	0	1	100	100	100	100
	7	6,358	0	1	0	99.98	0.00	0.00	0.00
	8	154,543	0	0	1	100	100	100	100
	9	100,881	0	0	10	100	100	100	100
	10	58,263	0	0	10	100	100	100	100
	11	2,324	0	0	3	100	100	100	100
	12	22,136	0	0	3	100	100	100	100
	13	123,285	0	0	1	100	100	100	100
D3	1	361,178	388	0	10	99.89	2.51	100	4.90
	2	382,871	206	0	10	99.95	4.63	100	8.85
	3	201,818	160	0	10	99.92	5.88	100	11.11

D = Dataset; D1: CTU-13 (sporadic); D2: NCC (periodic); D3: NCC-2 (simultaneous); TN = True Negative; FP = False Positive; FN = False Negative; TP = True Positive

generally performs well. In scenarios involving imbalanced data, even minor errors in detecting the majority class are critical, leading to a direct drop in *pre*. This occurs because the ratio of the numerator and denominator (TP + FP) becomes considerably large. As mentioned previously, a mere 0.11% misidentification of the majority class can reduce the precision value by 2.51% (see Table 8 D3; sub-1) since the numerator remains only at 10, despite the model's capability to accurately detect all instances in the minority class.

F. ABLATION STUDY

This section aims to examine the impact of each stage of the proposed method through an ablation study, which involves comparing the model's performance with and without certain steps. In this research, the ablation study is conducted under four different conditions. It provides insights into how each component of the proposed method contributes to its overall effectiveness.

The first condition assesses the model's performance when only the in-degree metadata from the group activity classification is used. The second condition evaluates the model's performance when only the out-degree metadata is utilized for group activity classification. The final condition investigates the effect of removing the weighting and threshold analysis, comparing the model's performance when no weights are applied and the threshold value (T) is set to 50.

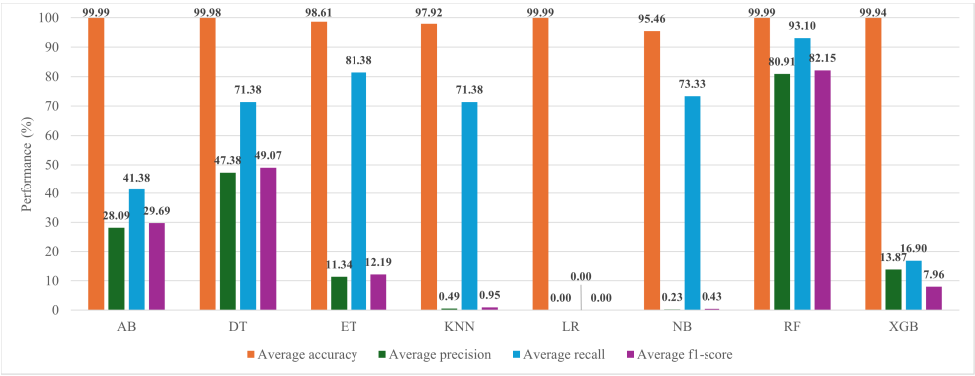

FIGURE 14. Average performance of each machine learning algorithms.

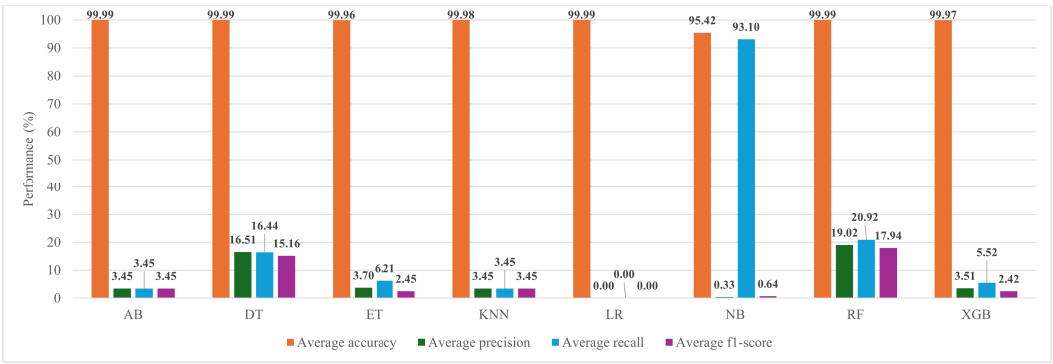

FIGURE 15. Detection performance with in-degree metadata only.

1) PERFORMANCE ONLY WITH IN-DEGREE METADATA CLASSIFICATION RESULT

The proposed model is designed to detect botnet IP addresses by analyzing the intensity of attacks derived from the network flow represented as a graph. Although in-degree graph metadata plays a significant role in the analysis process, relying solely on this metadata may not yield optimal results. Fig. 15 illustrates the model's performance when only in-degree metadata is used, with $T = 40$. While the average accuracy (*acc*) appears to be optimal, there is a noticeable disparity between the *acc* and the other metrics—precision (*pre*), recall (*rec*), and F1-score ($f1$).

The NB algorithm achieved the highest average recall performance, with a value of 93.10%. However, its average precision was significantly lower, indicating a high likelihood of false alarms. On the other hand, RF demonstrated the most stable performance overall, but still only achieved an average recall of 20.92% and an average precision of 19.02%. In summary, if the final decision relies exclusively on in-degree metadata, there is a substantial risk of false alarms, as evidenced by an average precision value that does not exceed 19%. This ablation study confirms the hypothesis that botnets can often be identified through out-degree metadata. However, in-degree metadata also proves to be highly informative, providing valuable insights into how bots establish communication and receive commands. This complementary perspective underscores the significance of incorporating both types of metadata for a more comprehensive understanding of botnet behavior.

2) PERFORMANCE ONLY WITH OUT-DEGREE METADATA CLASSIFICATION RESULT

This subsection provides an analysis of the model's performance when using only out-degree metadata with a threshold of $T = 40$. As shown in Fig. 16, the model demonstrates optimal average accuracy (*acc*) values. However, there remain significant issues with the average recall and precision metrics for several algorithms.

LR and NB perform the worst compared to other algorithms, even though NB achieved the highest average recall when classifying with in-degree metadata. False alarms continue to pose a challenge, with average precision values falling below 50% for most algorithms, except for RF, which consistently delivers the most stable performance, achieving an average recall of 87.36% and an average precision of 75.44%. Nevertheless, while RF performs well, its performance is still lower than that of the proposed model overall. The performance results indicate that relying solely on out-degree metadata is not a prudent approach. In-degree metadata offers equally important insights, particularly in recognizing communication patterns during TCP handshakes

TABLE 9. Execution time of proposed method with CTU-13, NCC and NCC-2.

D	Sub	Total NF	P1 (s)	P2 (s)	P3 (s)	P4 (s)	Total (s)	Speed (NF/s)
D1	1	706,160	16.97	17.33	17.36	1.29	52.96	13,335.09
	2	452,032	10.64	11.25	13.64	0.84	36.37	12,429.94
	3	1,177,661	27.83	31.20	24.65	1.17	84.85	13,879.76
	4	280,269	5.81	6.92	9.86	0.33	22.93	12,225.23
	5	32,459	0.67	2.56	7.22	0.06	10.51	3,087.64
	6	139,731	2.83	4.44	8.22	0.19	15.67	8,914.85
	7	28,521	0.58	2.45	7.14	0.05	10.22	2,789.66
	8	738,559	16.75	16.98	17.38	0.97	52.08	14,180.29
	9	521,878	10.75	10.61	12.64	0.99	34.99	14,916.58
	10	327,448	6.99	8.02	11.42	0.53	26.95	12,149.98
	11	26,814	0.58	2.53	7.36	0.07	10.53	2,545.33
	12	81,368	1.75	4.33	8.08	0.14	14.30	5,691.23
	13	481,288	11.22	12.77	14.36	0.58	38.93	12,363.99
D2	1	528,057	13.56	19.92	22.70	1.24	57.43	9,194.91
	2	366,296	9.86	15.30	18.46	0.92	44.53	8,225.03
	3	726,404	17.19	21.97	24.19	0.99	64.33	11,291.02
	4	181,098	4.52	7.97	13.47	0.27	26.22	6,905.81
	5	23,230	0.72	2.95	8.31	0.06	12.05	1,928.44
	6	128,006	3.19	6.19	11.89	0.23	21.50	5,954.74
	7	20,869	0.67	2.86	8.49	0.07	12.08	1,727.23
	8	717,805	17.08	21.95	23.87	1.06	63.97	11,220.98
	9	393,326	9.95	14.80	19.33	0.68	44.76	8,787.90
	10	246,093	6.08	10.55	16.67	0.40	33.70	7,302.94
	11	7,742	0.23	2.31	7.41	0.04	10.00	774.40
	12	68,548	1.78	5.00	10.17	0.20	17.15	3,995.89
	13	469,123	11.38	17.27	18.75	0.81	48.20	9,732.49
D3	1	1,223,790	29.71	35.33	21.67	2.18	88.88	13,768.37
	2	1,499,534	39.80	50.00	34.89	1.96	126.65	11,839.99
	3	971,449	21.78	23.30	18.11	1.08	64.26	15,116.76

D : Dataset; D1: CTU-13; D2: NCC; D3: NCC-2; P1 : Graph metadata extraction execution time (s); P2 : In-degree metadata classification execution time (s); P3 : Out-degree metadata classification execution time (s); P4 : Ensembling execution time (s); Speed : processed network flows per second (NF/s); NF : network flow; s : second

with targets, interactions with command and control (C&C) channels, and instances in which bots receive commands. By integrating both perspectives, we can achieve a more balanced and comprehensive analysis of botnet behavior.

3) PERFORMANCE WITHOUT WEIGHT AND THRESHOLD ANALYSIS

Weighting and threshold analysis are crucial components of the proposed method. Fig. 17 shows the model's performance when ensemble classification is performed without weighting and with a threshold value of $T = 50$. Similar to the proposed model, the accuracy metric shows strong performance, with an average value exceeding 95%. However, when considering the recall metric, ET and RF deliver the best performance, both exceeding 80%. Despite this, ET struggles with false alarms, resulting in low precision values.

Similarly, k-NN and NB exhibit the same difficulties in managing false alarms. Although their recall values are above 60%, their precision values, which do not even reach 1%, negatively impact their F1-scores. Overall, with a threshold of $T = 50$, RF remains the most stable algorithm, achieving an F1-score of 71.91%, followed by DT with 33.10%. This aligns with the trends shown in Fig. 13. However, using $T = 50$ still results in false alarm issues, as no algorithm other than RF achieves a precision value above 50%.

G. EXECUTION TIME ANALYSIS

The detailed execution times for each process in the proposed method are presented in Table 9. Execution time was recorded for each step, from graph metadata extraction to the ensemble process. Each sub-dataset has unique characteristics and varying amounts of network traffic. On average, processing one sub-dataset takes 39.55 seconds. Some datasets require more than one minute due to the large number of network flows and high activity intensity. On average, the proposed model processes 10,955.09 network flows per second.

Fig. 18 shows that the longest execution time occurs during the metadata classification phase (in-degree/out-degree). These two classification processes are conducted sequentially, which accounts for 72% of the total detection process. The model could be improved in future work by parallelizing the classification of the two metadata sets, potentially reducing execution time by approximately 30%.

This research also analyzes the training time required by each algorithm to process the in-degree and out-degree metadata (see Table 10). The best-performing model, RF, took 35,231.55 seconds to train the in-degree metadata and 24,638.04 seconds to train the out-degree metadata. These times may vary depending on the specifications of the device used.

Users can select the most appropriate model based on the analysis of training time and performance. For users prioritizing faster training times and a smaller model size with reasonably competitive performance, the DT algorithm may be a suitable option. On the other hand, users seeking the best detection results might opt for the RF algorithm, despite its longer training time and larger model size.

The challenges of extended training times and large model sizes can be mitigated by sharing pre-trained models, allowing users to avoid repeating the training process. This approach offers a practical solution for reducing resource consumption while maintaining model effectiveness. The analysis of execution and training times aligns with the primary objective of this study, which is to develop a detection model characterized by low complexity and lightweight features. While the model demonstrates commendable execution times, its application is limited to forensic analysis. The representation of network flow in the graph must occur only after all network flows have been recorded, rather than on a real-time basis as they arrive. This forensic analysis is crucial for network administrators, as it allows them to identify the characteristics of botnet attacks, thereby enhancing their ability to anticipate and prepare for future threats.

H. COMPARATIVE ANALYSIS

This research compares the proposed method with existing graph-based detection models that uses the same dataset, CTU-13. The previous studies that took CTU-13 for model testing are [19], [20], and [30]. Various studies utilize the CTU-13 dataset with a graph-based approach, including the work by Chowdhury et al. [29]. In that research,

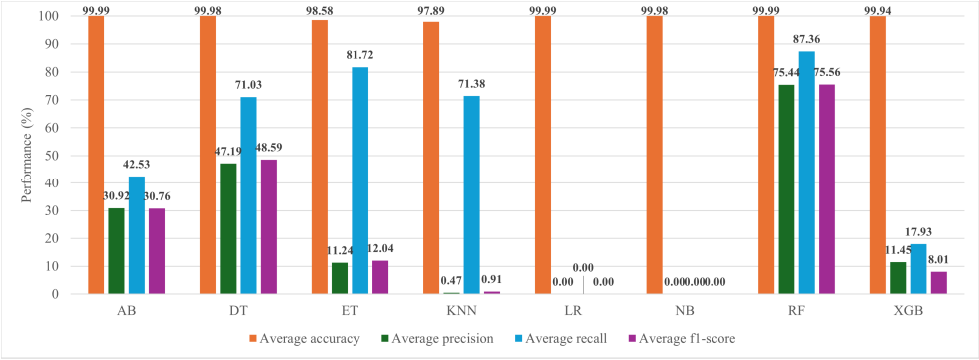

FIGURE 16. Detection performance with out-degree metadata only with CTU-13, NCC and NCC-2.

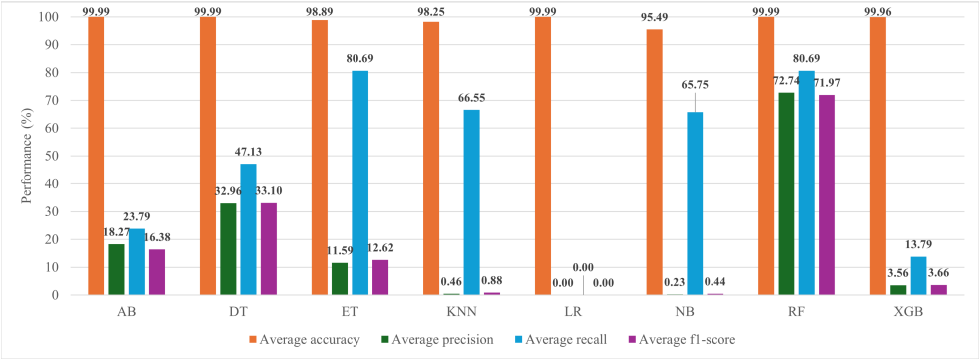

FIGURE 17. Detection performance without weighting and threshold analysis with CTU-13, NCC and NCC-2.

Chowdhury et al. [29] reported achieving a 100% detection rate for bot detection, indicating a perfect recall. However, additional metrics, like precision, were not provided, which restricts the ability to analyze and compare the results effectively. Different evaluation metrics are essential for understanding issues such as overfitting or the model's sensitivity to specific data, as previously discussed in our earlier research [38].

Several studies have adopted a graph-based approach to analyzing IoT botnets [32]. Nguyen et al. [32] employed an IoT botnet dataset that differs from traditional botnets, highlighting the IoTPOT dataset's balanced distribution of benign samples (7,199) and botnet samples (4,001). After removing non-binary and non-readable files, the ratio adjusted to 6,165 benign samples and 3,845 botnet samples. Despite an impressive accuracy of 98.7%, with a false negative rate (FNR) of 1.83% and a false positive rate (FPR) of 0.78%, the data imbalance noted differs from other studies.

Lefoane et al. [31] utilized the IoT-23 dataset, which investigates IoT botnets. However, this dataset primarily focuses on controlled environments, rendering it unsuitable for analyzing botnet behaviors in real-world scenarios. Traditional botnet data typically emerges from actual environments [35], whereas IoT-23 is derived from a controlled setup [34]. These differences can result in increased variability in background traffic, making detection of botnet attack patterns more challenging [35], [36], [37]. Furthermore, traditional botnets often employ sophisticated techniques to conceal activities among benign traffic, complicating detection efforts [41]. A significant factor is that traditional botnets usually exhibit a considerably higher ratio of normal traffic to attack traffic [35], [36], [37], further complicating detection challenges. In contrast, the IoT-23 dataset presents a more balanced ratio of attacks to normal traffic due to its controlled nature [31]. Additionally, while Lefoane et al. [31] focus on detecting network flow, this study targets Host/IP data, resulting in a more pronounced imbalance. Despite these challenges, the research demonstrates relatively stable performance.

TABLE 10. Each machine learning algorithm training time and model size with CTU-13, NCC and NCC-2.

Algorithm	In-degree Model size (KB)	In-degree Training time (s)	Out-degree Model size (KB)	Out-degree Training time (s)
AB	311	16,605.67	311	16,193.21
DT	227	228.18	252	155.15
ET	10,680,404	6,256.54	20,114,677	3,691.53
k-NN	-	-	-	-
LR	2	48.36	2	55.24
NB	2	12.84	2	7.37
RF	2,925,778	35,231.55	9,658,804	24,638.04
XGB	122	6,573.99	122	7,855.72

TABLE 11. Performance evaluation of proposed method vs previous research using CTU-13 dataset.

Metric	Method	S1	S2	S6	S8	S9	Average
acc	[20]	0.98	0.97	0.94	0.84	0.99	0.94
	[30]	0.98	**1.00**	**1.00**	0.97	0.99	0.99
	[19]	0.99	0.99	0.99	0.99	**1.00**	0.99
	Proposed	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
pre	[20]	-	-	-	-	-	-
	[30]	0.25	**1.00**	**1.00**	0.20	0.91	0.67
	[19]	0.50	0.34	0.50	**1.00**	**1.00**	0.67
	Proposed	**1.00**	0.50	**1.00**	**1.00**	0.83	**0.87**
rec	[20]	-	-	-	-	-	-
	[30]	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
	[19]	**1.00**	**1.00**	**1.00**	0.00	**1.00**	0.80
	Proposed	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**

S1: Sub dataset (scenario) 1; S2: Sub dataset (scenario) 2; S6: Sub dataset (scenario) 6; S8: Sub dataset (scenario) 8; S9: Sub dataset (scenario) 9

Thus, Daya et al. [20], Blaise et al. [30], and Alharbi and Alsubhi [19] studies evaluated their detection models using five CTU-13 sub-datasets (scenarios 1, 2, 6, 8, and 9). Table 11 presents a comparison of the performance of each existing method and the proposed method using three metrics: accuracy, precision, and *rec*. The proposed method achieves the highest average performance across all three metrics. It also performs well on each individual CTU-13 sub-dataset, although it exhibits lower precision in sub-datasets 2 and 9. Despite this, the proposed method remains competitive with other approaches, demonstrating promising results even with the reduced precision in those specific sub-datasets.

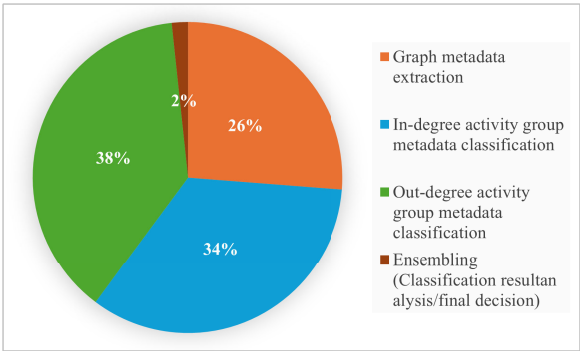

FIGURE 18. Each process execution time proportion of proposed model.

V. CONCLUSION

This research proposes a graph-based model for detecting botnet IP addresses in a network through adaptive weighting and feature extraction. It is specifically designed for post-attack forensic analysis to assess attack characteristics and anticipate potential future attacks. The approach involves grouping network flows based on IP addresses and their time of appearance to form activity groups. These groups are represented in a graph, where IP addresses serve as nodes (or vertices) and communication links between IP addresses are represented as edges. The directed graphs of the activity groups are then split into in-degree and out-degree graphs.

The next step involves applying 16 different weighting methods, derived from four network flow features, to each edge in the in-degree and out-degree graphs. The weighted graphs are then processed to extract two types of metadata: in-degree and out-degree graph metadata. These two metadata sets are classified using eight machine learning algorithms to generate a confidence score for each IP address. The final phase involves analyzing each IP address's confidence score from the in-degree and out-degree graphs using adaptive weighting and threshold values. The results from both the in-degree and out-degree graphs are combined in an ensemble approach to determine whether an IP address is classified as a botnet or a normal host.

The proposed model was tested on three datasets with varying botnet characteristics: CTU-13 (sporadic), NCC (periodic), and NCC-2 (simultaneous). The experimental results demonstrate that the model can effectively detect botnet IP addresses, achieving an average accuracy of 99.99%, precision of 80.91%, recall of 93.10%, and F1-score of 82.15%. The model successfully detected 100% of botnet IP addresses in 27 out of the 29 sub-datasets tested. However, the model encountered challenges in detecting low-intensity attacks. In terms of precision, the proposed method achieved a perfect score in 22 of the 29 sub-datasets, though performance in the NCC-2 dataset was limited, with precision as low as 6%. This lower precision in NCC-2 is attributed to data imbalance, as the false alarm rate remains below 0.11%. Additionally, the model's average execution time is 39.55 second.

A comparative analysis revealed that the proposed model outperforms existing graph-based detection models across three performance metrics: accuracy, precision, and recall. However, the primary drawbacks of the proposed method are the lengthy metadata training time and the large model size required for the best-performing algorithm, Random Forest. This issue could be addressed by selecting algorithms with faster training times and smaller model sizes, such as Decision Tree. Alternatively, sharing pre-trained models could eliminate the need for users to repeat the training process, allowing them to achieve optimal performance without incurring additional training time.

Future work will focus on detecting low-intensity attack activities by deepening network flow feature analysis and weighting techniques. Additionally, feature importance analysis and parallel processing will be explored to improve model efficiency. This study will also be extended by incorporating a wider range of open-source datasets to enhance the generalizability and robustness of the proposed approach.

ACKNOWLEDGMENT

The authors would like to thanks all research group members who have supported this research.

COMPETING INTERESTS

All authors have no competing interests.

DATA AVAILABILITY

- CTU-13 Dataset: https://www.stratosphereips.org/datasets-ctu13
- NCC (Botnet Group Activity Dataset): https://doi.org/10.17632/4vftxh97m8.1
- NCC-2 (Simultaneous Botnet Dataset): https://doi.org/10.17632/8dpt85jrhp.2

REFERENCES

[1] A. Affinito, S. Zinno, G. Stanco, A. Botta, and G. Ventre, "The evolution of Mirai botnet scans over a six-year period," *J. Inf. Secur. Appl.*, vol. 79, Dec. 2023, Art. no. 103629.

[2] O. I. Falowo, M. Ozer, C. Li, and J. B. Abdo, "Evolving malware and DDoS attacks: Decadal longitudinal study," *IEEE Access*, vol. 12, pp. 39221–39237, 2024.

[3] M. Al-Fawa'Reh, J. Abu-Khalaf, P. Szewczyk, and J. J. Kang, "MalBoT-DRL: Malware botnet detection using deep reinforcement learning in IoT networks," *IEEE Internet Things J.*, vol. 11, no. 6, pp. 9610–9629, Mar. 2024.

[4] D. C. Muñoz and A. D.-C. Valiente, "A novel botnet attack detection for IoT networks based on communication graphs," *Cybersecurity*, vol. 6, no. 1, p. 33, Dec. 2023.

[5] G. Wu, X. Wang, Q. Lu, and H. Zhang, "Bot-DM: A dual-modal botnet detection method based on the combination of implicit semantic expression and graphical expression," *Expert Syst. Appl.*, vol. 248, Aug. 2024, Art. no. 123384.

[6] J. B. Borges, J. P. S. Medeiros, L. P. A. Barbosa, H. S. Ramos, and A. A. Loureiro, "IoT botnet detection based on anomalies of multiscale time series dynamics," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 12, pp. 12282–12294, Dec. 2023.

[7] F. Alrowais, M. M. Eltahir, S. S. Aljameel, R. Marzouk, G. P. Mohammed, and A. S. Salama, "Modeling of botnet detection using chaotic binary pelican optimization algorithm with deep learning on Internet of Things environment," *IEEE Access*, vol. 11, pp. 130618–130626, 2023.

[8] A. M. Manasrah, T. Khdour, and R. Freehat, "DGA-based botnets detection using DNS traffic mining," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 34, no. 5, pp. 2045–2061, May 2022.

[9] A. Kurt, E. Erdin, K. Akkaya, S. Uluagac, and M. Cebe, "D-LNBot: A scalable, cost-free and covert hybrid botnet on Bitcoin's lightning network," *IEEE Trans. Dependable Secure Comput.*, vol. 21, no. 4, pp. 2162–2180, Jul. 2024.

[10] J. F. Balarezo, S. Wang, K. G. Chavez, A. Al-Hourani, and S. Kandeepan, "Dynamics of botnet propagation in software defined networks using epidemic models," *IEEE Access*, vol. 9, pp. 119406–119417, 2021.

[11] R. U. Khan, X. Zhang, R. Kumar, A. Sharif, N. A. Golilarz, and M. Alazab, "An adaptive multi-layer botnet detection technique using machine learning classifiers," *Appl. Sci.*, vol. 9, no. 11, p. 2375, Jun. 2019.

[12] M. Ali, M. Shahroz, M. F. Mushtaq, S. Alfarhood, M. Safran, and I. Ashraf, "Hybrid machine learning model for efficient botnet attack detection in IoT environment," *IEEE Access*, vol. 12, pp. 40682–40699, 2024.

[13] F. Sattari, A. H. Farooqi, Z. Qadir, B. Raza, H. Nazari, and M. Almutiry, "A hybrid deep learning approach for bottleneck detection in IoT," *IEEE Access*, vol. 10, pp. 77039–77053, 2022.

[14] D. P. Hostiadi and T. Ahmad, "Hybrid model for bot group activity detection using similarity and correlation approaches based on network traffic flows analysis," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 34, no. 7, pp. 4219–4232, Jul. 2022.

[15] I. Apostol, A.-D. Tica, and V.-V. Patriciu, "Design and implementation of a novel hybrid botnet," in *Proc. 14th Int. Conf. Electron., Comput. Artif. Intell. (ECAI)*, Jun. 2022, pp. 1–6.

[16] S. Li, Y. Cao, S. Liu, Y. Lai, Y. Zhu, and N. Ahmad, "HDA-IDS: A hybrid DoS attacks intrusion detection system for IoT by using semi-supervised CL-GAN," *Expert Syst. Appl.*, vol. 238, Mar. 2024, Art. no. 122198. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417423027008

[17] C. Joshi, R. K. Ranjan, and V. Bharti, "A fuzzy logic based feature engineering approach for botnet detection using ANN," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 34, no. 9, pp. 6872–6882, Oct. 2022.

[18] M. Putra, T. Ahmad, D. Hostiadi, R. Ijtihadie, and P. Maniriho, "EnglishBotnet attack analysis through graph visualization," *Int. J. Intell. Eng. Syst.*, vol. 17, pp. 913–927, Mar. 2024.

[19] A. Alharbi and K. Alsubhi, "Botnet detection approach using graph-based machine learning," *IEEE Access*, vol. 9, pp. 99166–99180, 2021.

[20] A. A. Daya, M. A. Salahuddin, N. Limam, and R. Boutaba, "BotChase: Graph-based bot detection using machine learning," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 1, pp. 15–29, Mar. 2020.

[21] W. W. Lo, G. Kulatilleke, M. Sarhan, S. Layeghy, and M. Portmann, "XG-BoT: An explainable deep graph neural network for botnet detection and forensics," *Internet Things*, vol. 22, Jul. 2023, Art. no. 100747.

[22] T. A. Tuan, H. V. Long, and D. Taniar, "On detecting and classifying DGA botnets and their families," *Comput. Secur.*, vol. 113, Feb. 2022, Art. no. 102549.

[23] M. Eslahi, W. Z. Abidin, and M. V. Naseri, "Correlation-based HTTP botnet detection using network communication histogram analysis," in *Proc. IEEE Conf. Appl., Inf. Netw. Secur. (AINS)*, Nov. 2017, pp. 7–12.

[24] W. Wang, Y. Shang, Y. He, Y. Li, and J. Liu, "BotMark: Automated botnet detection with hybrid analysis of flow-based and graph-based traffic behaviors," *Inf. Sci.*, vol. 511, pp. 284–296, Feb. 2020.

[25] M. Alshamkhany, W. Alshamkhany, M. Mansour, M. Khan, S. Dhou, and F. Aloul, "Botnet attack detection using machine learning," in *Proc. 14th Int. Conf. Innov. Inf. Technol. (IIT)*, Nov. 2020, pp. 203–208.

[26] H. Suryotrisongko and Y. Musashi, "Evaluating hybrid quantum-classical deep learning for cybersecurity botnet DGA detection," *Proc. Comput. Sci.*, vol. 197, pp. 223–229, May 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S187705092102359

[27] A. Pektaş and T. Acarman, "Deep learning to detect botnet via network flow summaries," *Neural Comput. Appl.*, vol. 31, no. 11, pp. 8021–8033, Nov. 2019, doi: 10.1007/s00521-018-3595-x.

[28] S. Hosseini, A. E. Nezhad, and H. Seilani, "Botnet detection using negative selection algorithm, convolution neural network and classification methods," *Evolving Syst.*, vol. 13, no. 1, pp. 101–115, Feb. 2022.

[29] S. Chowdhury, M. Khanzadeh, R. Akula, F. Zhang, S. Zhang, H. Medal, M. Marufuzzaman, and L. Bian, "Botnet detection using graph-based feature clustering," *J. Big Data*, vol. 4, no. 1, p. 14, Dec. 2017. [Online]. Available: http://journalofbigdata.springeropen.com/articles/10.1186/s40537-017-0074-7

[30] A. Blaise, M. Bouet, V. Conan, and S. Secci, "Botnet fingerprinting: A frequency distributions scheme for lightweight bot detection," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 3, pp. 1701–1714, Sep. 2020.

[31] M. Lefoane, I. Ghafir, S. Kabir, I.-U. Awan, K. E. Hindi, and A. Mahendran, "Latent semantic analysis and graph theory for alert correlation: A proposed approach for IoT botnet detection," *IEEE Open J. Commun. Soc.*, vol. 5, pp. 3904–3919, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10571994/

[32] H.-T. Nguyen, Q.-D. Ngo, and V.-H. Le, "A novel graph-based approach for IoT botnet detection," *Int. J. Inf. Secur.*, vol. 19, no. 5, pp. 567–577, Oct. 2020. [Online]. Available: http://link.springer.com/10.1007/s10207-019-00475-6

[33] X. D. Hoang and Q. C. Nguyen, "Botnet detection based on machine learning techniques using DNS query data," *Future Internet*, vol. 10, no. 5, p. 43, May 2018.

[34] S. Garcia, A. Parmisano, and M. J. Erquiaga, May 2021, "IoT-23: A labeled dataset with malicious and benign IoT network traffic," doi: 10.5281/zenodo.4743746.

[35] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Comput. Secur.*, vol. 45, pp. 100–123, Sep. 2014.

[36] D. P. Hostiadi and T. Ahmad, "Dataset for botnet group activity with adaptive generator," *Data Brief*, vol. 38, Oct. 2021, Art. no. 107334.

[37] M. A. R. Putra, D. P. Hostiadi, and T. Ahmad, "Botnet dataset with simultaneous attack activity," *Data Brief*, vol. 45, Dec. 2022, Art. no. 108628.

[38] M. A. R. Putra, T. Ahmad, D. P. Hostiadi, and R. M. Ijtihadie, "Botnet sequential activity detection with hybrid analysis," *Egyptian Informat. J.*, vol. 25, Mar. 2024, Art. no. 100440.

[39] M. G. Karthik and M. B. M. Krishnan, "Securing an Internet of Things from distributed denial of service and Mirai botnet attacks using a novel hybrid detection and mitigation mechanism," *Int. J. Intell. Eng. Syst.*, vol. 14, no. 1, pp. 113–123, Feb. 2021.

[40] Moch. A. Mawalid, A. Z. Khoirunnisa, M. H. Purnomo, and A. D. Wibawa, "Classification of EEG signal for detecting cybersickness through time domain feature extraction using NaÏve Bayes," in *Proc. Int. Conf. Comput. Eng., Netw. Intell. Multimedia (CENIM)*, Nov. 2018, pp. 29–34.
[41] F. F. Daneshgar and M. Abbaspour, "A two-phase sequential pattern mining framework to detect stealthy P2P botnets," *J. Inf. Secur. Appl.*, vol. 55, Dec. 2020, Art. no. 102645.

MUHAMMAD AIDIEL RACHMAN PUTRA was born in Medan, North Sumatra, Indonesia. He received the bachelor's degree in information technology from Universitas Sumatera Utara (USU), in 2019, and the master's degree in informatics engineering from the Institut Teknologi Sepuluh Nopember (ITS), Indonesia, in 2023, where he is currently pursuing the Ph.D. degree in cybersecurity and related disciplines.

Following his graduation, he began his career as a Backend Developer with PT Telkom Indonesia Tbk., where he involved on various projects related to logistics, fulfilment, sales force management, and order management systems. In September 2021, he joined the Net-Centric Computing Laboratory, as a Research Assistant, focusing on advancing knowledge in the fields of botnet detection and intrusion detection systems. His dedication to research has led to several published works in these areas, contributing to the broader understanding of cybersecurity challenges. His research interests include cybersecurity, botnet detection, intrusion detection systems, anomaly detection, and data mining.

Mr. Putra is an Active Member of IEEE and has been recognized as an awardee of the PMDSU scholarship, granted by the Ministry of Education, Culture, Research, and Technology of Republic of Indonesia.

TOHARI AHMAD (Member, IEEE) received the bachelor's degree in computer science from the Institut Teknologi Sepuluh Nopember (ITS), Indonesia, the master's degree in information technology from Monash University, Australia, and the Ph.D. degree in computer science from RMIT University, Australia, in 2012.

From 2001 to 2003, he was a consultant for some international companies. In 2003, he moved to ITS, where he is currently a Professor. His research interests include network security, information security, data hiding, and computer networks.

Prof. Ahmad is a member of ACM. His awards and honors include the Hitachi Research Fellowship and JICA Research Program to conduct research in Japan. He is a reviewer of a number of journals.

DANDY PRAMANA HOSTIADI (Member, IEEE) was born in Surabaya, Indonesia. He received the bachelor's degree in computer systems from STMIK STIKOM Bali, the master's degree in electrical engineering, focusing on information systems and computer management from Udayana University, and the Ph.D. degree from the Institut Teknologi Sepuluh Nopember (ITS), where he worked on finding ways to detect botnet activities in computer networks.

His Ph.D. research, titled "Development of Correlation Mechanisms and Detection of Botnet Activity in Computer Networks," looks at the growing risks from malware, especially botnets, that threaten network security. In his work, he suggested four new methods to spot botnet attacks and understand how these attacks are connected. Besides his research, he was an Associate Professor with the Institut Teknologi Dan Bisnis STIKOM Bali, where he shares his knowledge about network security and forensics with students. He is dedicated to improving awareness of cyber threats and contributes to the field through his publications.

ROYYANA MUSLIM IJTIHADIE received the Ph.D. degree from Kumamoto University, Japan, in 2013. He is currently a Senior Lecturer and a Researcher with the Department of Informatics, ITS Surabaya, Indonesia. He has experience in various cases of distributed systems and IT infrastructure.