IoT network Malicious Behaviour Profiling Based on Explainable AI Using LSTM and SHAP

Sepideh Niktabe

A Thesis Submitted to the Faculty of Graduate Studies
In Partial Fulfillment of the Requirements
for the Degree of Master of Applied Science

Graduate Program in Computer Science

Supervisor: Arash Habibi Lashkari, Ph.D.

York University
Toronto, Ontario

September 4, 2024

Abstract

The proliferation of IoT devices has enhanced connectivity but exposed networks to new cyber threats, particularly from botnets. Detecting and identifying malicious data is critical for early threat detection, understanding botnet attack patterns, and deploying countermeasures. This research proposes an IoT Bot detection and identification profiling model using XAI. The proposed model introduces a novel feature selection technique with the XGBoost algorithm and a correlation-based feature selection technique to enhance efficiency. An optimized LSTM neural network enables accurate bot detection and identification, with hyperparameters selected using the Bayesian Optimization algorithm. SHAP analysis provides insightful individual and collective bot characteristic profiles. The model's performance was evaluated using the augmented BCCC-Aposemat-Bot-IoT-24 dataset, built upon the Aposemat-Bot-IoT-23 dataset, and compared against established models assessed primarily on the same dataset in previous research. The results showed that the proposed model performed comparably to these models, with distinct advantages, including handling sequential and time-series data, managing imbalanced datasets, and providing explainable insights into botnet behavior. The model's design also emphasizes computational efficiency, making it potentially suitable for deployment in resource-constrained environments.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Professor Arash Habibi Lashkari, whose invaluable guidance, insightful advice, and unwavering support have been instrumental throughout the entire process of completing this Master's thesis. His expertise in the field, combined with his dedication to his students, has made him an exceptional mentor. I am truly grateful for his patience, encouragement, and the countless hours he dedicated to helping me refine my research and develop my academic skills.

I am also immensely grateful to my parents, whose constant encouragement and belief in me made this achievement possible. Their unwavering support has been a driving force throughout my academic journey.

I am deeply appreciative of the love and encouragement from the rest of my family and my wonderful friends, who have been my pillars of strength and have stood by me every step of the way. Their support has been invaluable and has helped me overcome challenges and stay motivated.

Finally, I extend my thanks to the Department of Graduate Studies for their valuable feedback and support throughout this journey. Their guidance and assistance have been instrumental in my academic development and have contributed significantly to the successful completion of this thesis.

Contents

List of Tables

List of Figures

Table 1: List of Abbreviations

Abbreviation	Definition
Adam	Adaptive Moment Estimation
Adagrad	Adaptive Gradient Algorithm
AIDS	Anomaly-based Intrusion Detection System
ANN	Artificial Neural Networks
AC	Aho-Corasick
ARC	Argonaut RISC Core
AWS	Amazon Web Services
BCCC	Behaviour-Centric Cybersecurity Center
BiLSTM	Bidirectional Long-Short Term Memory
BRNN	Bidirectional Recurrent Neural Network
C&C	Command and Control
CBUS	Clustering-Based Under-sampling
CBOUS	Clustering-Based Over-sampling and Under-sampling
cGAN	Conditional Generative Adversarial Network
CNN	Convolutional Neural Network
CSSAE	Cost Sensitive Stacked Auto Encoder
DBN	Deep Belief Network
DBRF	Descriptive Back Propagated Radial Basis Function
DCGAN	Deep Convolutional Generative Adversarial Network
DCNN	Deep Convolutional Neural Network
DoS	Denial of Service
DDoS	Distributed Denial of Service
DFC	Direct Filter Classification
DGA	Domain Generation Algorithm
DNN	Deep Neural Network
DRF	Decisive Red Fox
DT	Decision Tree
ET	Extra Trees
XGBoost	Extreme Gradient Boosting
FL	Federated Learning
FFNN	Feed Forward Neural Network
FP	False Positive
FN	False Negative
FPR	False Positive Rate
FNR	False Negative Rate
GA	Genetic Algorithm
GB	Gradient Boosting

Abbreviation	Definition
GAN	Generative Adversarial Network
GOA	Grasshopper Optimization Algorithm
GRU	Gated Recurrent Unit
HS	Hybrid Sampling
HIDS	Hybrid Intrusion Detection System
ID	Intrusion Detection
IDE	integrated development environment
IDS	Intrusion Detection System
IIoT	Industrial Internet of Things
IoT	Internet of Things
KNN	K-Nearest Neighbors
LM	Language Model
LSTM	Long Short-Term Memory
MLP	Multi-Layer Perceptron
NB	Naive Bayes
NIDS	Network-based Intrusion Detection Systems
NSL-KDD	Network Security Laboratory-Knowledge Discovery and Data Mining
OS	Over Sampling
OSS	One-sided Selection
PCA	Principal Component Analysis
PFAC	Parallel Failureless AC
RBM	Restricted Boltzmann Machine
RF	Random Forest
RMSProp	Root Mean Square Propagation
RNN	Recurrent Neural Network
ROSE	Random Over-Sampling Examples
S3	Simple Storage Service
SDN	Software Defined Networking
SGD	Stochastic Gradient Descent
SHAP	SHapley Additive exPlanations
SIDS	Signature-based Intrusion Detection System
SOM	Self-Organizing Map
SVM	Support Vector Machine
TNR	True Negative Rate
TP	True Positive
TPR	True Positive Rate
TN	True Negative
TCP	Transmission Control Protocol

Abbreviation	Definition
US	Under Sampling
UDP	User Datagram Protocol
UNSW-NB15	University of New South Wales Network-based 2015 dataset
XAI	Explainable Artificial Intelligence
6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks

1 Introduction

The advent of new technologies, including sensors, broadband internet, 5G, radio frequency identification, smartphones, portable computers, industrial automation, semi-autonomous vehicles, satellites, and cloud computing, has led to the creation of the IoT [21]. IoT represents a vast network of interconnected smart devices and computational units, allowing seamless communication and interaction. However, this integration introduces an expanded attack surface, making IoT systems vulnerable to novel cyberattacks that exploit these devices as entry points or direct targets. IoT systems' complexity and unique architectures challenge the effectiveness of traditional cybersecurity measures [22, 23].

The diversity of IoT devices, produced by numerous companies with varying protocols and standards, complicates the implementation of uniform security measures. Additionally, these devices' limited computational and power capabilities hinder the adoption of advanced security solutions [24, 25]. This situation is exacerbated by IoT devices being attractive targets for DDoS attackers due to their sheer numbers and inherent security weaknesses [25]. It is estimated that 25% of all cyberattacks are aimed at IoT devices, a figure expected to rise [26]. The critical role of IoT in various aspects of human activity underscores the importance of addressing IoT security concerns [27, 28]. IDSs are vital in identifying unauthorized intrusions and malicious activities within networks. The transition to IoT and distributed networks necessitates a shift from traditional knowledge-based IDS to intelligent, data-driven systems capable of learning from data and identifying anomalies [23]. Profiling and analyzing the behavior of malicious network traffic are crucial for enhancing the effectiveness of IDS. Deep learning, with its capacity for intensive learning, offers promising solutions for IoT security, capable of processing large datasets and outperforming traditional systems in accuracy. However, deep learning systems are often considered 'black boxes' due to the opaque nature of their feature extraction and decision-making processes [23]. XAI has emerged as a solution to enhance the explainability of deep learning-based systems in cybersecurity, addressing the need for transparency and trust in these systems [29]. XAI aims to make the decision-making processes of 'black box' systems understandable, particularly in the context of IoT networks where complexity and heterogeneity of data are prevalent. Furthermore, the effectiveness of machine learning and deep learning systems in IoT IDS heavily depends on the availability of high-quality, labeled datasets. The scarcity of such datasets poses significant challenges for developing new and effective detection methods [30]. The need for datasets that accurately reflect current network architectures, include data on new attacks, and are balanced and free from redundancies is critical. This research tackles IoT security challenges by introducing an IoT Bot detection and identification profiling model using XAI. The model employs an innovative feature selection method using XGBoost and correlation-based feature selection. LSTM is used for detection and identification. The hyperparameters are selected using the Bayesian Optimization algorithm. SHAP is applied for behavioral profiling of malicious bot samples individually and collectively. The performance of the proposed model is evaluated using the augmented BCCC-Aposemat-Bot-IoT-24 dataset, which is based on the Aposemat-Bot-IoT-23 dataset. This dataset includes six different malicious bot samples and approximately 315 features. This study significantly enhances IoT security measures and understanding of the behavior of malicious bot samples, thereby improving the effectiveness and reliability of IDS in IoT environments.

The major contributions of this research work can be stated as follows:

- Proposing a novel feature selection technique to eliminate redundancy and prevent multi-collinearity
- Proposing an optimized model suitable for resource-constrained IoT edge devices
- Generating a bot family unique behavior profiling model
- Generating an augmented dataset namely BCCC-Aposemat-Bot-IoT-24

This research is organized as follows. Section 2 reviews existing IDSs for detecting malicious IoT network traffic. In section 3, we detail the proposed IoT Bot detection and identification profiling model using XAI. Section 4 outlines the implementation details, including dataset generation, sampling techniques, and the used tools and platforms. Following this section 5 explains the evaluation process of the proposed model on the BCCC-Aposemat-Bot-IoT-24 dataset, including feature selection, model optimization, generating a profile for each malicious bot sample samples individually and collectively, and finally, comparing the proposed system with traditional models commonly used for ID on the Aposemat-Bot-IoT-23 dataset. Finally, section 6 concludes the study.

2 Literature Review

This section reviews the previous works on IDS in IoT networks. Based on the previous studies, IDSs are categorized into four types according to their detection strategies: AIDS, SIDS, specification-based IDS, and HIDS [30, 31, 32, 22].

2.1 Anomaly-based IDSs (AIDSs)

AIDSs detect intrusion by comparing activities to known attack patterns. These systems create a user profile by examining system activities; Any activities that diverge from the profiles are flagged as intrusions [22]. AIDSs are particularly effective in detecting zero-day attacks and those associated with misuse of system resources. They operate on principles that any activity diverging from the expected behavior is potentially malicious [1]. Therefore, analyzing the user behavior profile is critical to AIDSs. Additionally, these systems have shown high positive rates. AIDSs can be categorized into learning and non-learning-based systems [33].

2.1.1 Learning-based AIDSs

Learning-based AIDSs utilize machine learning algorithms to identify and prevent unauthorized access or attacks on networks and systems. These systems involve training a system with a dataset comprising both normal and malicious activities, enabling the system to differentiate between benign and malicious behavior for threat detection [22, 1]. Learning-based AIDSs can be divided into classic machine learning and neural network systems. These systems can automatically discover key distinctions between normal and abnormal data with high accuracy [23]

2.1.1.1 Classic Machine learning

A classic machine learning system for behavior characterization in IoT environments involves using traditional machine learning approaches to identify and prevent unauthorized access or malicious activities. These systems utilize feature engineering to extract relevant features for attack classification [31].
Mohy-eddine *et al.* [1] presented an AIDS utilizing the KNN algorithm. Fig. 1 shows the scheme of their proposed system. It comprises typical components of an AIDS [34, 35]: data source module, pre-processing module, decision core module, and response module. They utilized feature selection methods to decrease the prediction time and resource consumption and enhance the system performance. They used the PCA, KBest as a univariate statistical test implemented, and GA separately for feature selection for improving data quality and selecting the ten best-performing features. The system was evaluated on the Bot-IoT dataset. Applying feature selection could reduce the prediction time from 51,182.22 seconds to under a minute. For all the mentioned feature selection techniques, the system could reach an accuracy of around 99.99%. Furthermore, the FPR of the system utilizing all feature selection techniques was less than 0.1. Moreover, they calculated the prediction time, as they considered it critical in building AIDS for IoT, and by applying feature selection, they reduced it significantly from 51,182.22 seconds to under a minute.

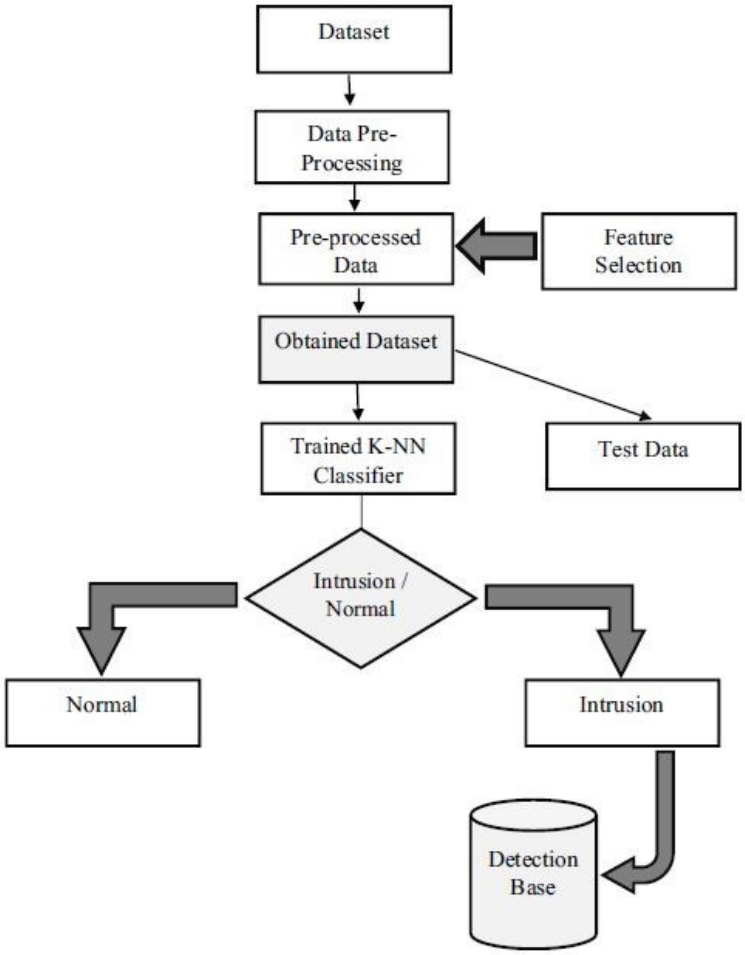


Figure 1: Proposed system by Mohy-eddine *et al.*[1]

Douiba *et al.* [2] introduced an enhanced AIDS utilizing GB and DT algorithms via the open-source Catboost, an efficient open-source package combining GB and DT algorithms specifically designed for IoT Security. They evaluated their system using the enhanced NSL-KDD, IoT-23, BoT-IoT, and Edge-IIoT datasets, leveraging GPU technology to optimize the experimental environment. Fig. 2 illustrates the architecture of their proposed system. Compared to existing AIDS solutions, this new approach demonstrated impressive performance, achieving nearly 99.09% accuracy in detection metrics such as accuracy, recall, and precision, as well as in computational efficiency. Based on the experiments, integrating GPU at the fog computing level can potentially minimize the ID time and assure responsiveness.

S *et al.* [36] presented an integrated architecture of the IoT-Fog-Cloud computing system to reduce latency between IoT sensors and the cloud. He utilized an improved NB classifier, based on the PCA technique, for anomaly detection in AIDS. The conventional NB classifier does not consider that different features have diverse contributions to classification. Moreover, it overlooks the issue of feature weighting. To address this, he initially extracted primary features from the dataset using PCA to reduce data dimensionality and classification time. He also considered the PCA input rate as the weight of features in NB classification. The goal is to reflect the significance of each feature in the classification decision. To evaluate the attack

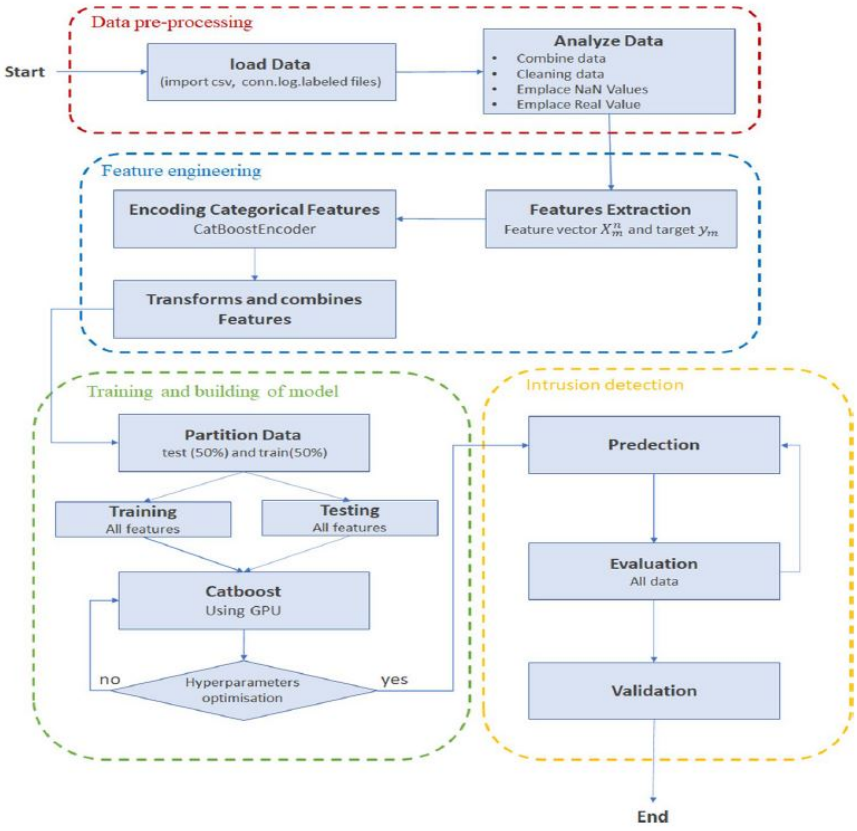


Figure 2: Proposed system by Douiba *et al.* [2]

detection system, he used the UNSW NB15 dataset. He compared his system with RF, KNN, and SVM. The proposed system outperformed other systems and achieved an accuracy of 92.48% and a detection rate of 95.35%.

Abbas *et al.* [3] proposed an ensemble-based system for ID. They preprocessed the CICIDS2017 dataset and applied six machine learning algorithms: DT, gaussian NB, multinomial NB, RF, logistic regression, linear SVM, and SGD Classifiers. Among these, RF achieved the highest average accuracy, reaching 99.67% in both classification types. The study also utilized four feature selection methods for regularization: Chi-2, ANOVA (from the filter method), RF, and linear SVM (from the embedded method), focusing on the top 15 features. However, the rationale behind selecting these 15 features was not explained. Fig. 3 exhibits the flowchart of the proposed ID method. In feature selection, linear SVM surpassed others in binary and multi-class scenarios, attaining average accuracies of 88.19% and 85.56%, respectively. Notably, in the multi-class format, three classifiers—NB, DT, and LR—showed accuracies of 83.93%, 93.50%, and 87.60%, respectively. In binary classification, these same algorithms performed best, with accuracies of 85.24%, 88.45%, and 88.58%, respectively. The proposed system incorporated logistic regression, NB, and DT algorithms in a hard voting ensemble method. This approach improved accuracy over existing systems in binary and multi-class scenarios.

Stivan *et al.* [37] conducted research on the recognition of ping flood attack patterns in IoT networks. Their study involved experiments using wireless communication under three distinct scenarios: normal traffic,

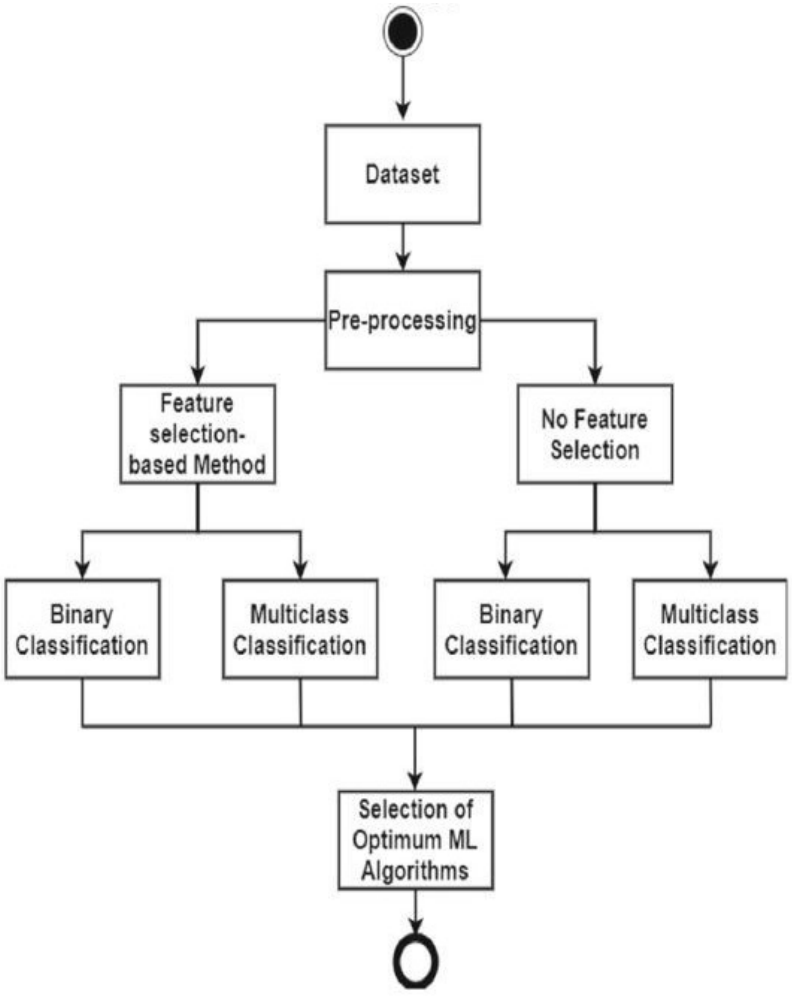


Figure 3: Proposed system by Abbas *et al.* [3]

attack traffic, and combined normal-attack traffic. This resulted in the creation of separate datasets for each scenario. The datasets were then categorized into two clusters: normal and attack. The average packet count in the attack cluster was 95,931, while 4,068 in the normal cluster. The K-Means algorithm was used to produce the clustering results. They also clustered normal and attack data packets based on ping flood attack pattern attributes by the random tree algorithm, which provided an algorithm that provided similar results as K-Means. The evaluation of the ping good attack detection results yielded to the accuracy levels of up to 99.94%. The performance metrics, derived from the confusion matrix, included a TNR of 98.62%, TPR of 100.00%, FNR of 0.00%, and FPR of 1.38%.

2.1.1.2 Neural Networks

Unlike traditional machine learning systems, neural network-based systems utilize various linear and non-linear processing layers to abstract discriminative or generative features for pattern analysis [38, 34]. These networks are typically categorized into supervised and unsupervised learning systems [33, 39]. Supervised neural networks include simple neural networks and deep learning approaches, while unsupervised neural

networks encompass a subset of deep learning algorithms. A simple neural network typically refers to a basic artificial neural network designed for learning tasks, with the single-layer perceptron being the most basic form [39]. Deep learning systems cover a range of deep networks, including supervised systems like DNNs, CNNs, and LMs. In contrast, unsupervised systems include autoencoders, RBMs, GANs, DBNs, and SOMs [33].

Javeed *et al.* [40] introduced a novel AIDS by CNN and BiLSTM within a horizontal FL framework. This system leverages the strengths of both architectures, with CNN adept at identifying local spatial patterns indicative of intrusions and BiLSTM adept at capturing sequential dependencies in data. To protect data privacy, this approach adheres to a zero-trust system. Training occurs locally on edge devices, and only the learned system weights are shared with a centralized FL server. This server aggregates updates from multiple devices to refine a global system. Evaluated on the CICIDS2017 and Edge-IIoTset datasets, this system outperformed centralized and other federated deep learning-based AIDS. Impressively, it achieved high accuracy rates of 99.32% and 99.97% respectively, even with 20 edge devices participating across 10 communication rounds. By keeping data local and sharing only system weights, this approach effectively safeguards sensitive information while still enabling collaborative learning for enhanced ID.

Kumar *et al.* [4] introduced a cutting-edge image-based AIDS that combines an SDN honeypot with a two-level autoencoder and CNN. Fig.4 exhibits the proposed system. The system converts binary programs into grayscale images and extracts textural features using deep CNN architectures, employing transfer learning for enhanced performance. A custom two-level autoencoder reduces the dimensions of these extracted features, which are input into six classifiers, including KNN, SVM, RF, MLP, ET, and Gaussian NB. The study optimizes hyperparameters for each classifier using automated grid search and random search algorithms. Evaluated on the public MalImg dataset, the system demonstrates superior performance compared to existing methods, achieving a remarkable test accuracy of 98.50% and weighted precision, recall, and F1-score of 98.60%. Moreover, the system boasts a rapid mean response time of 0.006 seconds, highlighting its efficacy in real-time anomaly detection scenarios.

J.Rabie *et al.* [5] developed a novel security system combining DRF optimization with DBRF classification. This approach utilizes the recently developed DRF optimization method incorporated with a machine learning algorithm, aiming to enhance the security of IoT systems. Fig.5 illustrates the workflow of this proposed security system. Initially, the system involves preprocessing and normalizing data to create a balanced IoT dataset, which enhances the detection accuracy of the classification process. Subsequently, the DRF optimization algorithm is employed to finely tune the features essential for precise ID and classification. This optimization not only accelerates the training process but also minimizes the error rate of the classifier. Following this, the DBRF classification system is utilized to distinguish between normal and malicious data flows, leveraging the optimized features. The effectiveness of the proposed DRF-DBRF security system is assessed using various datasets, including IoTID20, NetFlow-IoT-v2, NetFlow-ToN-IoT, NSL-KDD, and UNSW-NB 15. Comparative analysis reveals that the DRF-DBRF combination surpasses other anomaly detection techniques in terms of precision (99%), accuracy (99.2%), recall (99%), and F1-score (98.9%).

Altuny *et al.*[6] proposed three distinct systems aimed at detecting intrusions in the IIoT network by using deep learning architectures of CNN, LSTM, and CNN + LSTM, generated from a hybrid combination of

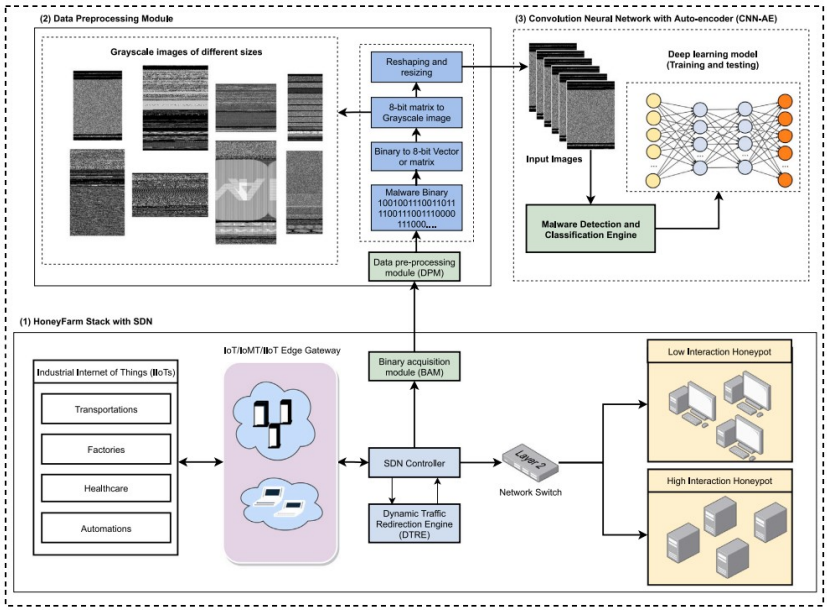


Figure 4: Proposed system by Kumar *et al.* [4]

these. Fig. 6 shows the architecture for the CNN+LSTM system. In order to evaluate the proposed systems, they utilized UNSW-NB15 and X-IIoTID datasets. Among the introduced systems, the hybrid CNN + LSTM system showcased superior accuracy in detecting intrusions across both datasets. Specifically, this system achieved a 93.21% accuracy rate in binary classification and 92.9% in multi-class classification on the UNSW-NB15 dataset. For the X-IIoTID dataset, the same system demonstrated a 99.84% accuracy in binary classification and 99.80% in multi-class classification.

Khan *et al.* [7] conducted an analysis of neural network algorithms and hybrid SOM on NSLKDD and IoT Botnet datasets, focusing on system efficiency. Fig.7 depicts the proposed system. Their research highlighted that the hybrid SOM required considerably less training time, indicating its suitability for IoT devices due to the significant reduction in training duration compared to traditional ANN. The study revealed a remarkable 948% reduction in training time and a 41.87% improvement in F1-score, demonstrating the proposed system's superior anomaly detection capabilities and efficient resource consumption. In comparisons on the Majran imbalanced dataset, the hybrid SOM surpassed the neural network in terms of precision, recall, and F1-score macro averages. The investigation also found that reducing the number of nodes in the system significantly cuts training time, and enhances inference performance. This was evidenced by a comparison of three hybrid SOM systems on the NSL-KDD dataset with different numbers of nodes, where the system with the fewest nodes exhibited a 27.28% boost in macro averages of precision, recall, and F1-score, and a 15.72% increase in weighted evaluation metrics, alongside a 955% reduction in training time when reducing nodes from 36 to 5.

Zhao *et al.* [8] addressed the issue of class imbalance in datasets for IoT ID by employing the DCGAN-SCB algorithm to increase the limited sample data, alongside developing an ID system that integrates GRU, and ResNet. Fig 8 indicates their proposed system. Initially, a DCGAN is utilized to generate a few sample data in the class imbalance data to make the sample data reach balance. Subsequently, the system employs

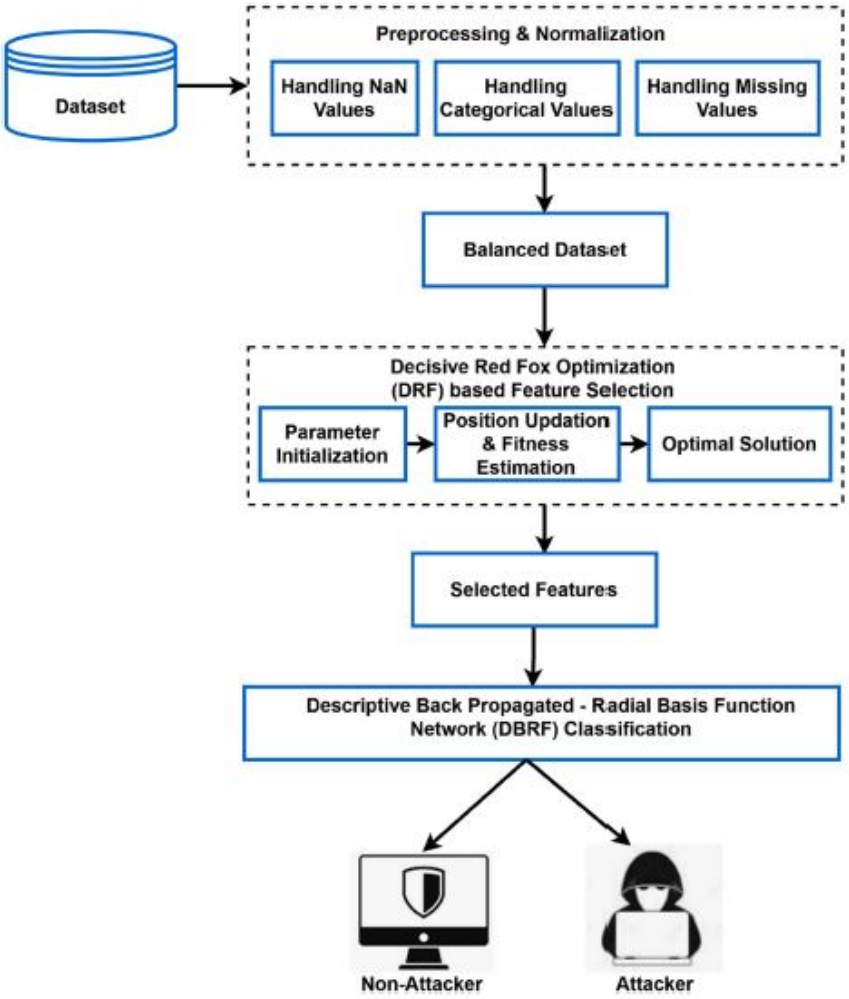


Figure 5: Proposed method by J. Rabie *et al.* [5]

GRU to learn the data features, extract time series features of the sample data, and classify the sample data features with the ResNet. Finally, the classification outcomes are normalized using the softmax function. The system's effectiveness was tested on the NSL-KDD dataset, showing superior performance in both single, and multi-class detection tasks when compared to systems based on LSTM, LSTM-ResNet, GRU, and SVM. Specifically, the proposed system achieved an accuracy of 96.12% and a detection rate of 97.85%, surpassing the LSTM-ResNet system by 1.86% and 2.59%, respectively. The results confirm the system's superiority over GRU, LSTM, and SVM-based approaches.

Dushimimana *et al.* [9] employed a BRNN to develop a robust security solution for IoT network protection. Their proposed system's flowchart is depicted in Fig.9. The system leverages an RF classifier algorithm to select the important features. The effectiveness of this system was assessed using the KDD Cup 99 ID dataset. The findings demonstrated that the BRNN system surpasses traditional RNNs, and GRU networks, achieving an accuracy rate of 99.04%. The superiority of BRNN is attributed to its enhanced architecture, which incorporates additional cells and hidden neurons to process information from both past and future states. This feature addresses the limitations observed in RNNs, such as the vanishing gradient problem and

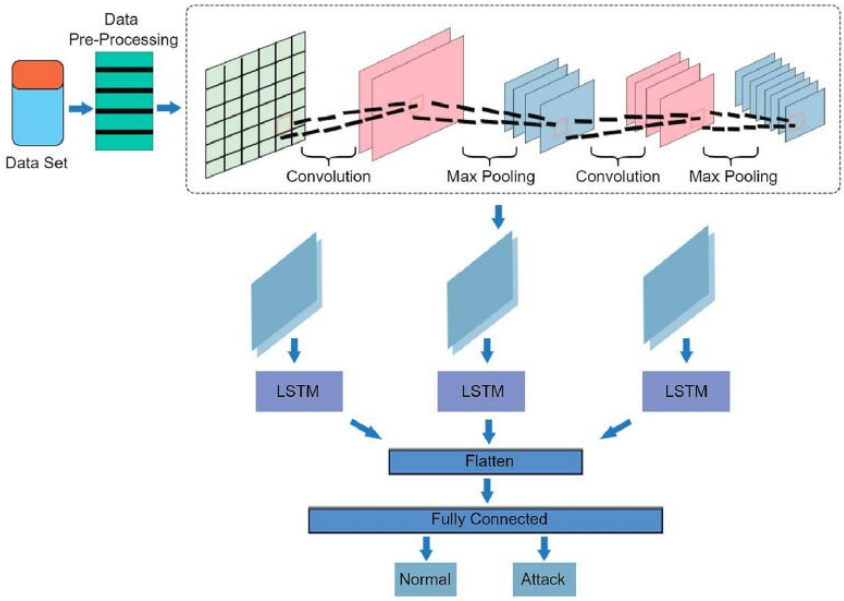


Figure 6: Proposed system by Altuny *et al.*[6]

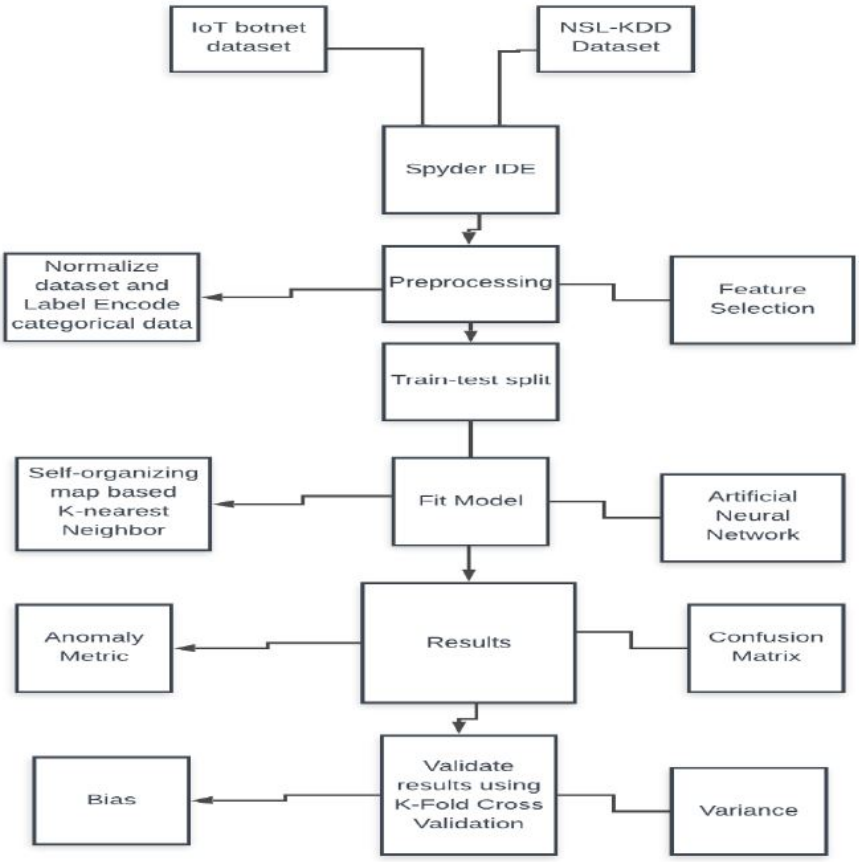


Figure 7: Proposed system by Khan *et al.* [7]

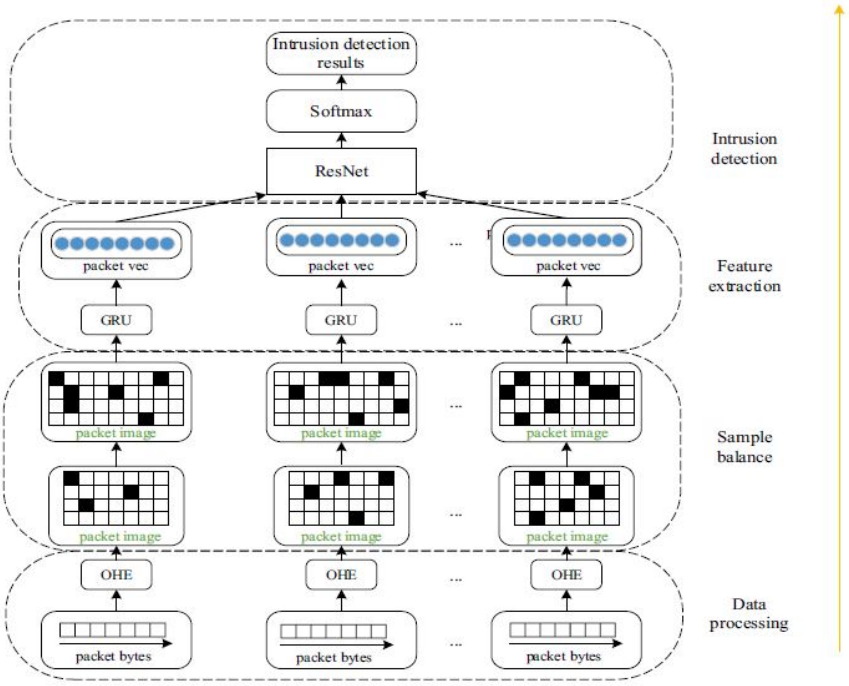


Figure 8: Proposed system by Zhao *et al.*[8]

the GRUs, which are constrained by a fewer number of cells in comparison to BRNNs.

Ullah *et al.* [10] utilized cGANs to generate realistic data distributions for given feature sets, addressing the challenge of data imbalance. Fig 10 shows their proposed system. They employed a one-class cGAN system to learn from the minority class, thereby balancing the dataset. Subsequently, a binary class cGAN system was used to create augmented data for the binary balanced dataset. The effectiveness of the one-class cGAN and binary-class cGAN systems was assessed in both binary and multiclass classification settings. They used a FFNN for evaluation on two network-based, and five IoT network-based anomaly detection datasets. The systems proved to be versatile, achieving high detection rates across various testing environments. Specifically, the binary class cGAN system outperformed existing GAN-based methods in nearly all evaluation metrics, with a notable 98.10% accuracy on the KDD99 dataset. Additionally, a multiclass classification framework utilizing the binary class cGAN system was developed and tested, further showcasing the one-class cGAN and binary class cGAN systems' consistency and effectiveness in anomaly detection within IoT networks. However, it was observed that the systems' detection rates were significantly lower when the training dataset contained fewer than 1000 samples, indicating the need for a minimum sample size to ensure system accuracy. The binary class cGAN system's performance was particularly strong on balanced datasets, highlighting the importance of equal representation of classes during training.

Moghanian *et al.* [11] developed an AIDS utilizing data mining, and machine learning to identify network intrusions. This system utilizes an MLP, which is a multilayer ANN, as a learning technique in ID. To enhance the detection accuracy, and reduce error rates, the system incorporates a meta-heuristic algorithm based on swarm intelligence. The GOA is specifically employed to refine the learning process of the MLP, aiming to minimize ID errors by optimizing parameters like weight, and bias. The system's performance was

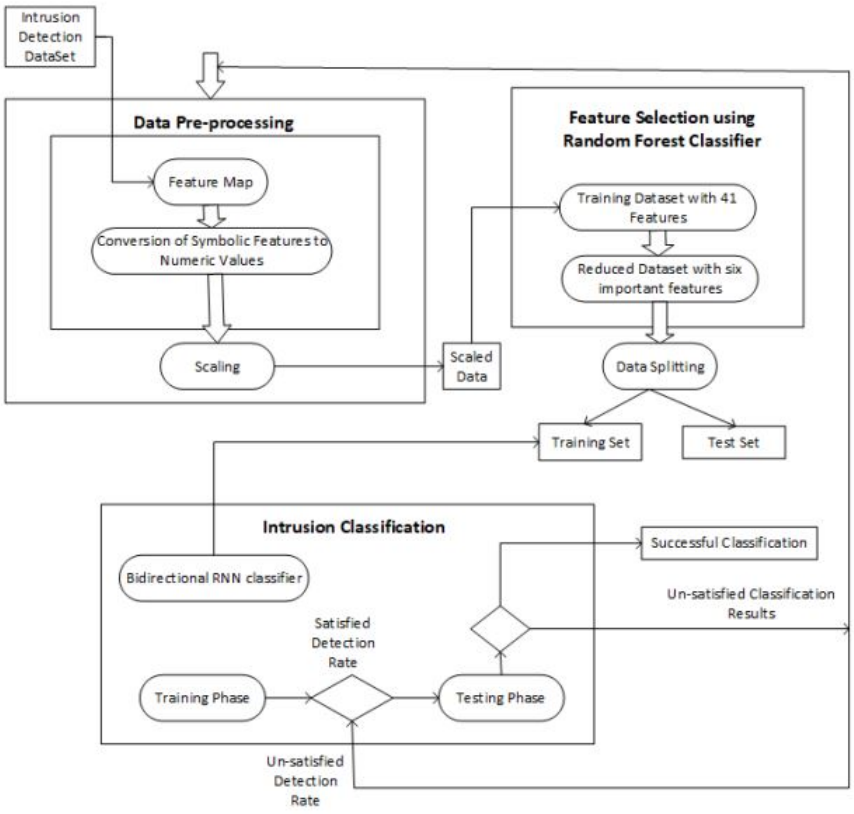


Figure 9: Proposed system by Dushimimana *et al.* [9]

assessed using the KDD, and UNSW datasets. As depicted in Fig.11, the ANN's accuracy is significantly improved through GOA optimization for NIDS. The system demonstrated high efficiency in detecting abnormal and malicious network activities, achieving accuracy rates of 95.41% and 98.88% on the KDD, and UNSW datasets, respectively. The GOAMLP approach surpassed other contemporary state-of-the-art techniques, including RF, XGBoost, and embedded learning with BOA, HHO, and BWO algorithms, in network ID accuracy.

Ullah *et al.*[12] introduced an AIDS based on a DCNN. Fig. 12 depicts the architecture of the proposed system. A DCNN consists of two convolutional layers and three densely connected layers. This system is designed to enhance performance while reducing computational demands. It was tested using the IoTID20 dataset for binary, multi-class category, and subcategory classifications. Different layers of the CNN algorithm were evaluated to determine the optimal configuration. The system's performance was thoroughly assessed using the Adam, Nadam, and AdaMax optimizers. For binary, multi-class category, and multi-class subcategory classifications, the Nadam optimizer yielded the best results with batch sizes of 128, 32, and 64, respectively. Additionally, the system was benchmarked against contemporary deep learning techniques and conventional machine learning algorithms to determine its efficacy and robustness. The experimental results demonstrate that the proposed system achieved superior performance, as evidenced by its accuracy, precision, recall, and F1-score.

Almiani *et al.* [13] introduced a fully automated AIDS tailored for enhancing fog security against cyber threats. The system's workflow, depicted in Fig. 13, outlines the proposed IoT ID framework. This system

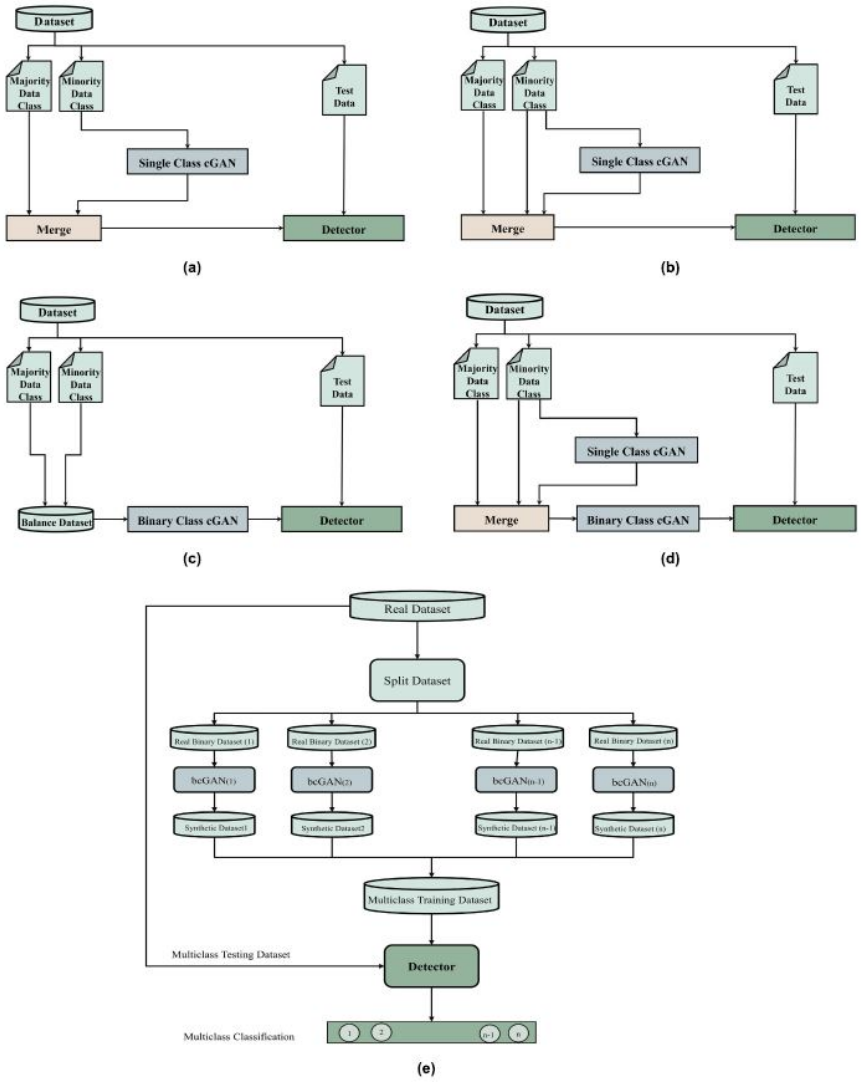


Figure 10: Proposed system by Ullah *et al.* [10]

incorporates multiple layers of RNNs, specifically designed for fog computing security, bringing protection closer to end-users, and IoT devices. The effectiveness of this system was evaluated using a balanced iteration of the NSL-KDD dataset, a well-known benchmark in the field. Performance was assessed using standard metrics, with the addition of the Matthew correlation and Cohen's Kappa coefficients for a more comprehensive analysis. The experimental outcomes and simulations underscored the system's stability and efficacy across various performance indicators.

Elsaeidy *et al.* [14] proposed a smart city ID framework based on RBMs. As Fig.14 shows, the proposed system consists of two main parts: the RBM system for feature learning and the classifier system for attack detection. Their approach involved experimenting with varying depths of RBM by adjusting the number of layers to optimize performance. They assessed the effectiveness of this method using simulated data from a smart water distribution network. To enhance the robustness of their system, they modified the original dataset to include multiple types of attacks, whereas it initially contained only one. The study explored several classification techniques, including standard FFNNs, an automated version of FFNNs that selects op-

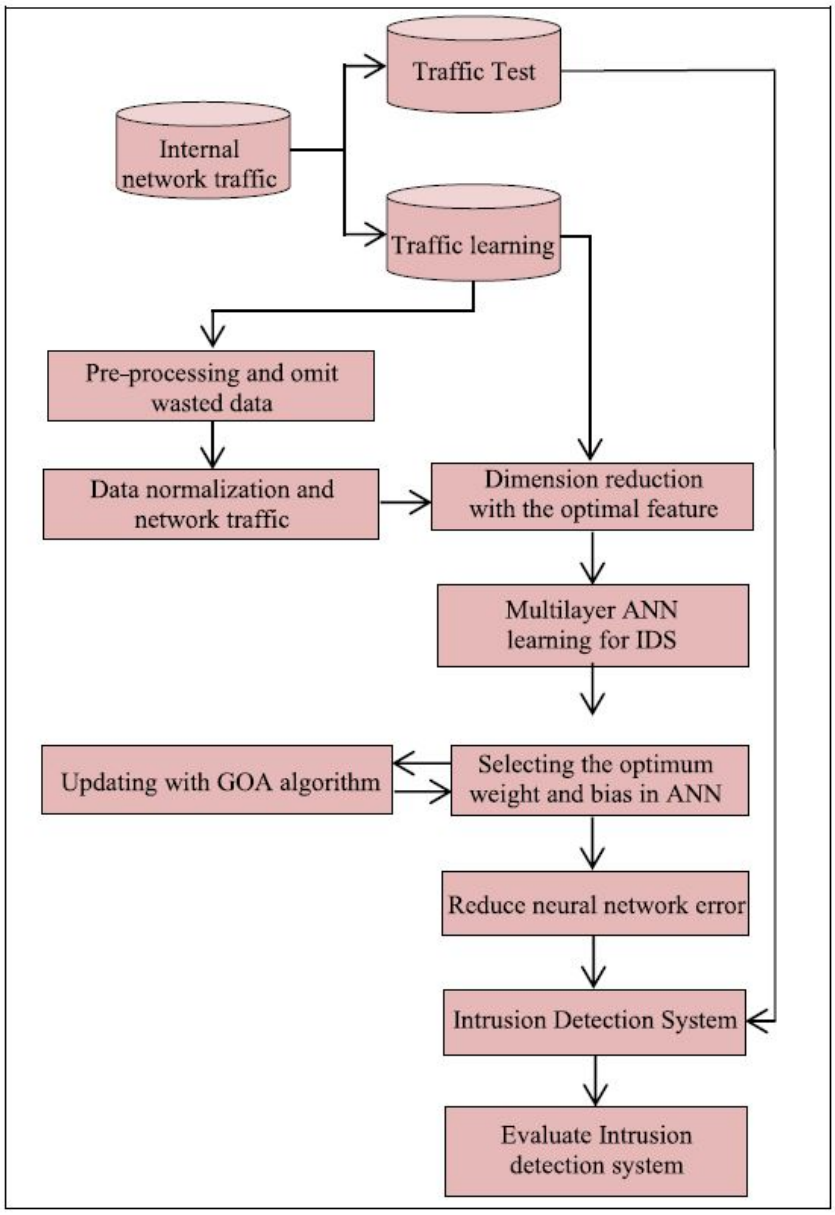


Figure 11: Proposed system by Moghanian *et al.* [11]

timal hyperparameters automatically, RF, and SVM. The findings revealed that the automated FFNNs were particularly effective in detecting attacks, surpassing the performance of standard FFNNs, RF, and SVM systems. Notably, the most successful systems were those that integrated single-layer RBM systems. However, adding more RBM layers caused a decrease in accuracy for the learned systems. Moreover, increasing the number of clusters has an effect on the attack detection performance, which is decreased by increasing the number of classes generated by the clustering algorithm.

Telikani *et al.* [15] proposed a CSSAE specifically designed to address the class imbalance issue in IDS. The CSSAE system generates a cost matrix where each class is assigned a distinct cost, reflecting the class distribution. This matrix is formulated during the initial stage of the CSSAE process. Subsequently, in the

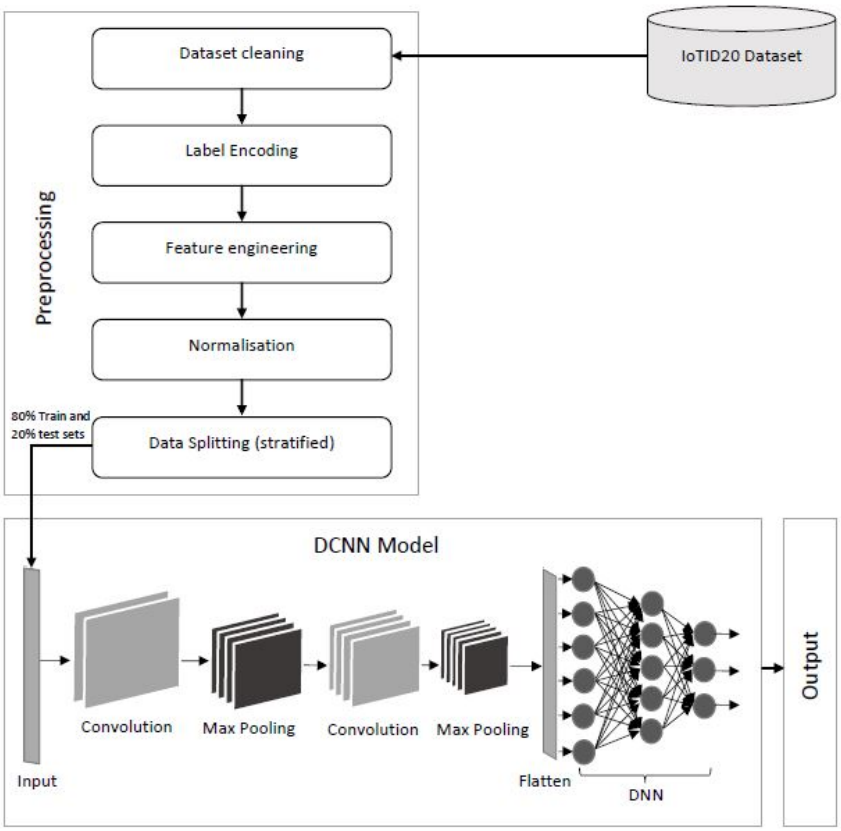


Figure 12: Proposed system by Ullah *et al.* [12]

second stage, a two-layer stacked auto-encoder is employed to enhance the feature discrimination between minority and majority classes. The costs derived from the matrix are incorporated into the deep learning feature extraction process, adjusting the neural network parameters within the cost function layers to reflect these costs. Fig.15 depicts the CSSAE system. As the figure shows, it utilizes a softmax classifier in its output layer for the classification task, showcasing the application of the SAE algorithm with two hidden layers. The enhancements to the cost function, guided by a cost-sensitive approach, further refine its efficacy in handling class imbalances. This improved system is compatible with both binary and multiclass datasets. To validate the effectiveness of CSSAE, it was tested using the renowned KDD Cup 99 and NSL-KDD datasets. The performance evaluation indicates that CSSAE outperforms other AIDSs that do not account for class imbalance, particularly in detecting low-frequency attacks.

Balakrishnan *et al.* [16] introduced an advanced system for enhancing cybersecurity using deep learning techniques, specifically through the application of DBNs. They illustrated the systematic process of DBN in ID through Fig.16. This sophisticated approach for detecting intrusions monitors harmful activities within the network, focusing on unauthorized attempts to gain access. The effectiveness of the DBN-enhanced security measures was evaluated against conventional DGAs and AIDSs strategies. The comparative analysis revealed areas where the proposed method requires further refinement to effectively address both standard and unforeseen security threats.

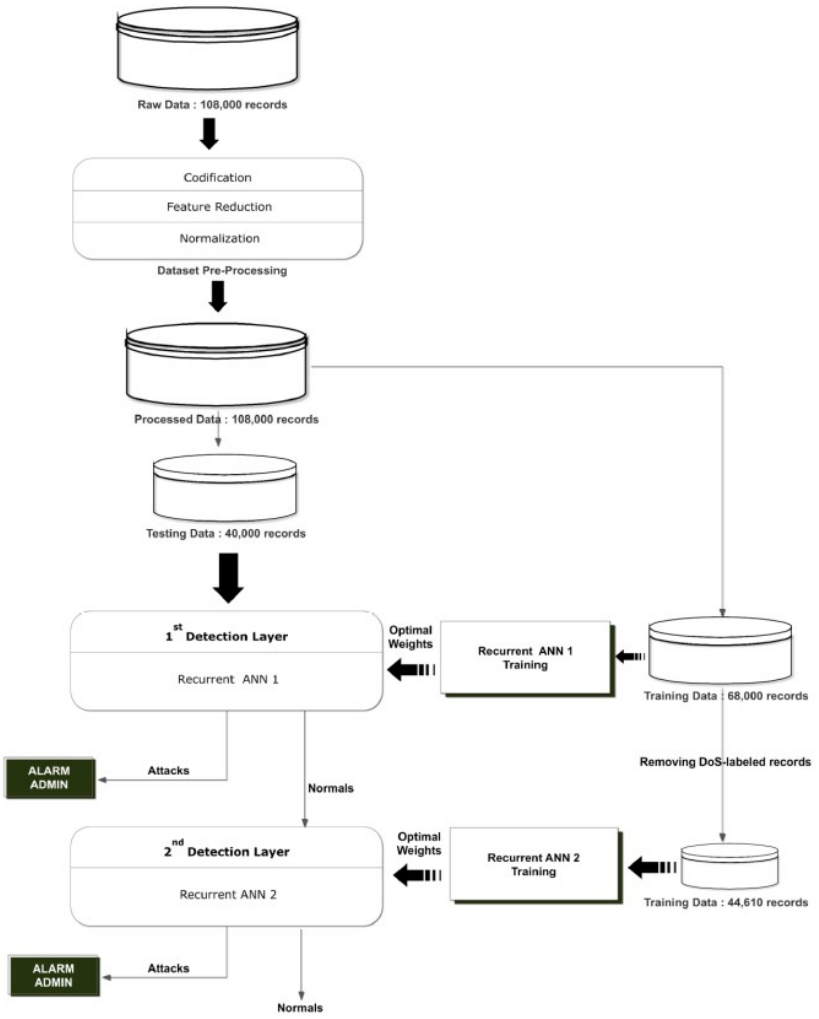


Figure 13: Proposed method by Khan *et al.* [13]

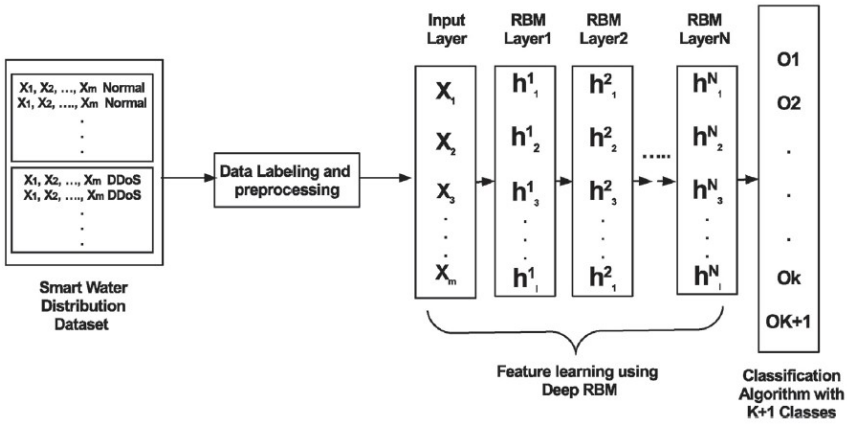


Figure 14: Proposed system by Elsaeidy*et al.* [14]

2.1.2 Non-Learning-based

Non-learning-based AIDSs do not use machine-learning approaches to detect unauthorized activities. They are essential for IoT contexts with limited computational resources. These systems are grouped into three

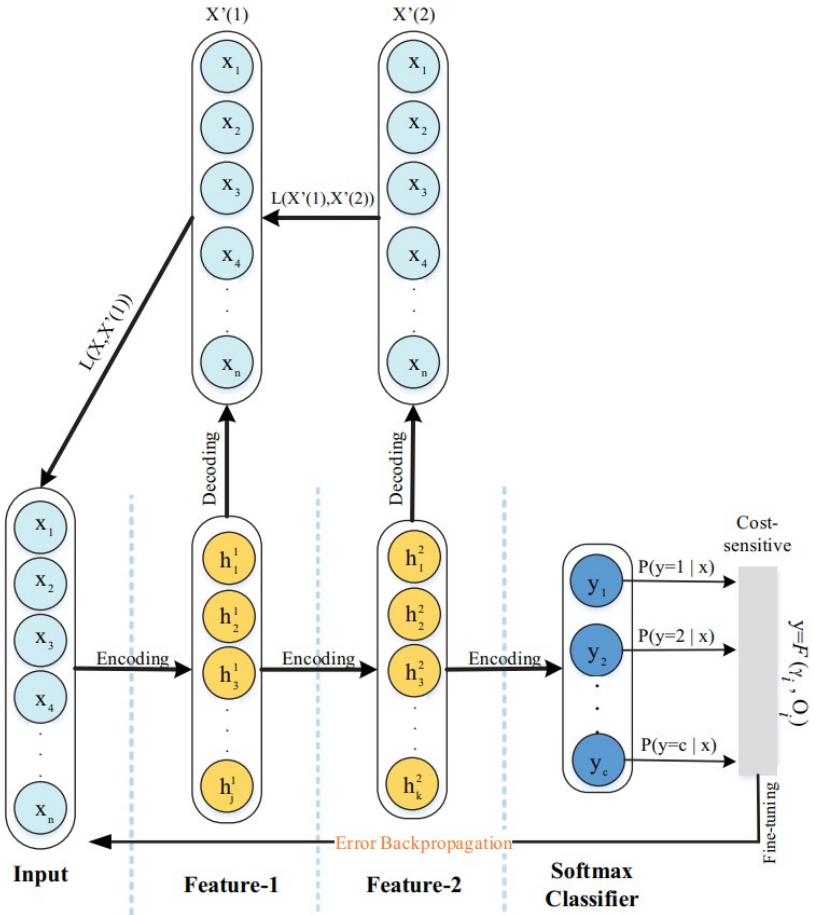


Figure 15: Proposed system by Telikani*et al.* [15]

categories: statistical-based, rule-based, and data mining [22, 41, 17, 42].

2.1.2.1 Statistical system

A statistical-based AIDS involves using statistical approaches to analyze network traffic, system logs, or other relevant data to identify patterns and anomalies that could indicate security breaches or malicious activities [43]. .

Stiawan *et al.* [41] researched recognizing ping flood attack patterns within IoT networks. The study involved wireless communication experiments under three distinct scenarios: normal traffic, attack traffic, and a mix of both. These experiments generated datasets corresponding to each scenario, which were subsequently divided into two categories: normal and attack. The K-Means clustering algorithm was employed to analyze the data, and its effectiveness was evaluated using a confusion matrix. The clustering accuracy achieved by applying the K-Means algorithm was remarkably high at 99.94%. The detailed outcomes from the confusion matrix showed a true negative rate of 98.62%, a true positive rate of 100%, no FNR, and an FPR of 1.38%.

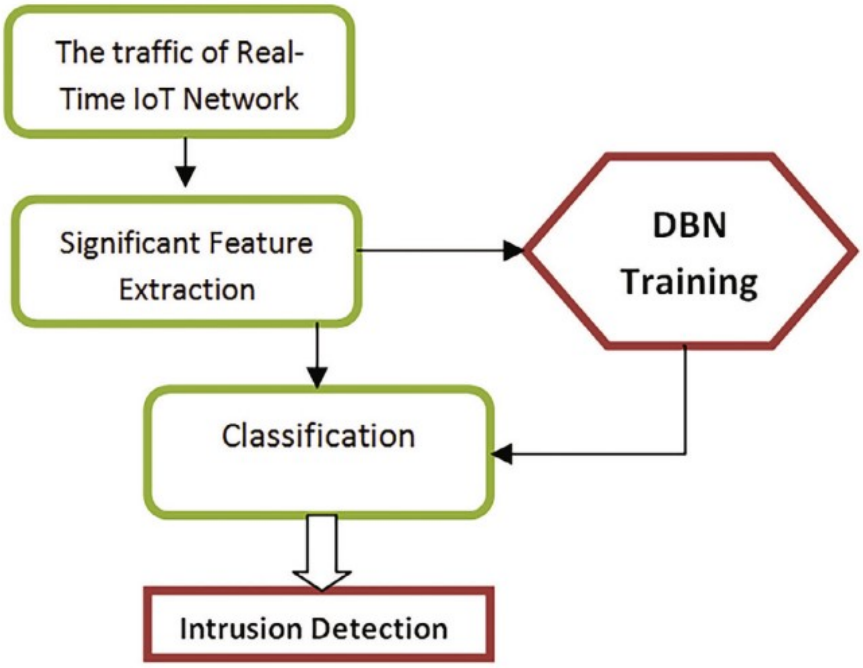


Figure 16: Proposed system by Balakrishnan *et al.* [16]

2.1.2.2 Rule-based

A rule-based AIDS involves defining rules or patterns that network traffic and device behavior must follow. When these rules are violated, it indicates a potential security breach or malicious activity [44].
Ferrag *et al.* [17] proposed a hierarchical AIDS illustrated in Fig. 17. This system integrates three distinct classifiers: REP Tree, JRip algorithm, and Forest PA. It uniquely employs two classifiers in parallel to enhance a third classifier, effectively mitigating data imbalance impacts. The system's innovation lies in its simultaneous use of binary and multi-class classifiers that feed a third classifier in parallel, thus minimizing the effect of data imbalance on the outcome. They evaluated the proposed system on real-world datasets "CICIDS2017" and "BoT-IoT" datasets. The result showed that the hierarchical system outperformed different well-known and recent machine learning systems, giving the highest TNR and highest DR for seven types of attacks. It achieved remarkable detection rates of 94.47%, and 95.175%, accuracy rates of 96.66%, and 96.99%, and maintained the lowest false alarm rates at 1.14%, and 1.12%, respectively.

2.1.2.3 Data mining

Data mining approaches in AIDS are employed to detect unusual patterns in network traffic or system activities, which may indicate a security breach. These patterns can describe the behavior data from users or networks within a computing environment [22].
Hu *et al.* [42] mentioned that the lack of sampling data hinders using multi-kernel clustering algorithms. In this order, they proposed an AIDS for 5G and IoT networks based on multiple kernel k-means with incomplete kernels. They evaluated their proposed system utilizing SLKDD, UNSW, and AWID datasets. The experimental results show that the proposed system can achieve high-accuracy clustering when the

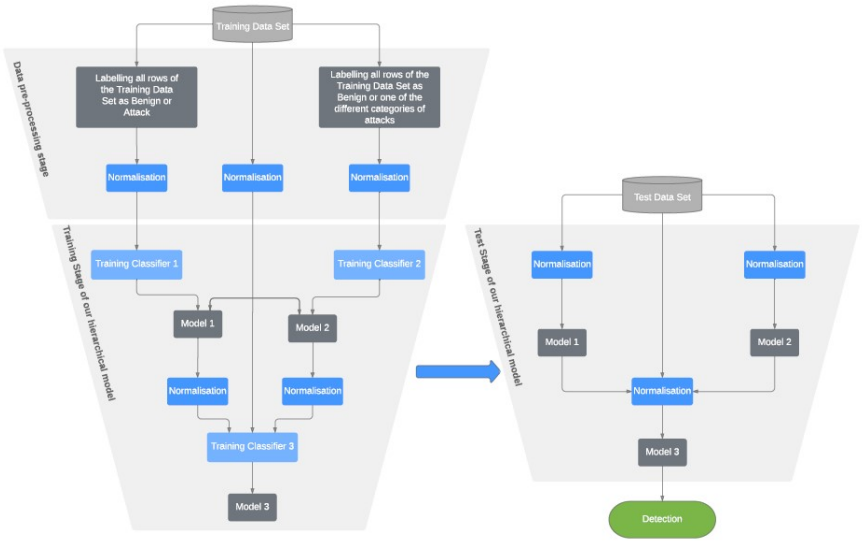


Figure 17: Proposed system by Ferrag*et al.* [17]

sampled data is incomplete. However, the proposed system has some problems in processing performance and processing massive amounts of data.

2.2 Signature-based IDSs (SIDSs)

SIDSs identify attacks by comparing network traffic signatures to a database of known signatures [1]. It triggers an alarm when a match is found. This system is effective for known attacks but can not detect novel or variant attacks not in its database [31, 45, 45]. It is classified into three main categories: pattern matching, expert system, and finite-state-machine.

Yang *et al.* [46] introduced a lightweight SIDS that integrates deep learning, and knowledge graphs. Fig. 18 presents the architecture of the proposed system. Initially, the system extracts semantic relationships and key features by knowledge graph and statistical analysis. Subsequently, requests within the IoT network are transformed into word vectors via multiview feature fusion and alignment. Finally, the attention-based CNN-BiLSTM system is employed to detect malicious request attacks, effectively capturing long-range dependencies and contextual semantics. To evaluate their proposed system, they utilized the NSL-KDD dataset. Experiment results show that the proposed system significantly outperforms the existing solution in terms of robustness. Moreover, it can select more critical features for ID to achieve better accuracy and lower the false alarm rate. Compared with the state-of-the-art systems, the proposed IDS achieves a higher detection accuracy of 90.01%. In addition, the system can detect various stealthy attack types (including DoS, Probe, R2L, and U2L) and extract semantic relationships among features.

Stylianopoulos *et al.* [47] conducted a comprehensive comparative analysis of pattern matching algorithms, such as AC, DFC, PFAC, and HYBRID (a combination of DFC, and PFAC), specifically for NID on embedded devices with GPUs. Their research aimed to methodically explore how these algorithms perform in relation to the architectural features of medium to high-level devices commonly used in IoT deployments. Their findings suggest that utilizing GPUs in embedded devices for pattern-matching tasks in SIDSs offers

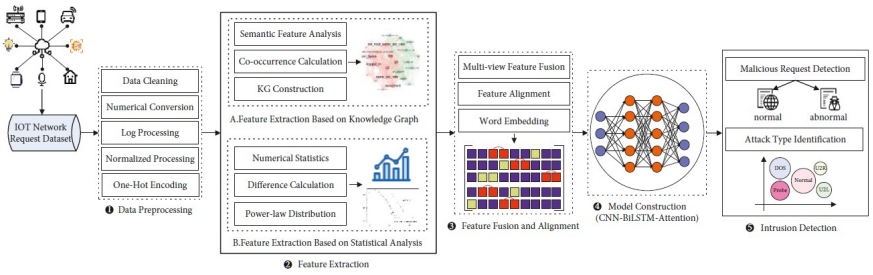


Figure 18: Proposed system by Yang *et al.* [17]

significant advantages over traditional CPU-based approaches, notably in reducing execution time and energy consumption by up to 50%. The study also delved into the impact of algorithmic engineering strategies tailored to the device's architectural characteristics, like shared memory between GPU and CPU, on the performance of these algorithms. Notably, the HYBRID algorithm emerged as particularly effective when implemented on GPUs, achieving a performance boost of up to 40% over the best-performing GPU-based standard.

Fu *et al.*[48] proposed a uniform SIDS for diverse IoT networks. The proposed system uses an extension of labelled transition systems to propose a uniform description of IoT systems and can detect intrusions by comparing the abstracted action flows. They designed the ID approach, built the Event Databases, and implemented the Event Analyzer to achieve the SIDS approaches. The effectiveness of the proposed SIDS was demonstrated by its ability to identify three distinct IoT attack types: jamming, falsification, and replay attacks. Furthermore, they established an experimental setup to validate the IDS methodology and assess its performance against attacks targeting the RADIUS application.

2.3 Specification-based IDSs

Specification-based IDSs operate on a set of predefined rules to create behavior profiles for network devices like nodes, routers, and servers [49]. These profiles include information about protocols and routing tables. This system generates alerts when there is a deviation from the specified behavior [45]. These systems function similarly to AIDS, as both trigger alerts when the network profile deviates from normal behavior. However, specification-based IDS differ by manually constructing device profiles through defining rules by security experts [50]. These system reports lower FPs, and do not need prior learning to understand network device behavior or network flow [31, 51]. However, signature databases require constant updates, which demands human intervention [1]. Since these systems define specifications manually, adjusting to network environmental changes takes more time, and they are more prone to errors [31].

2.4 Hybrid-based IDSs (HIDSs)

Hybrid IDSs integrate the concepts and benefits of SIDS, AIDS, and specification-based IDS to achieve high levels of classification accuracy and detection rates [45, 1].

Awotune *et al.* [18] developed a multilevel RF algorithm for ID. They enhanced the system by utilizing a fuzzy inference system. Fig.19 shows the architecture of their proposed system. They proposed an advanced

multi-level feature selection technique by combining strengths of the filter, and wrapper approaches. In order to select essential features, they used the filter method using a correlation-based feature selection based on the multi-collinearity in the data. The correlation-based feature selection utilized a genetic search method to choose the best features from the feature set. The GA evaluated the merit of each attribute, subsequently selecting those with the highest fitness value. A rule assessment has also been deployed to determine if two feature subsets have the same fitness values. In such cases, the system prefers the subset with the fewer number of features. In the second stage, a wrapper method utilizing the sequential feature selection technique is employed to further refine the selection of top features. This is based on the performance of the baseline classifier, specifically focusing on its accuracy. The selected features through these features serve as input into the RF algorithm, which is designed for detecting intrusions. Ultimately, the fuzzy logic categorized intrusions into one of four classifications: normal, low, medium, or high. This approach is intended to decrease the rate of misclassification. Comparison of the proposed method with the existing systems using the same dataset demonstrated a higher accuracy, precision, sensitivity, specificity, and F1-score of 99.46%, 99.46%, 99.46%, 93.86%, and 99.46%, respectively.

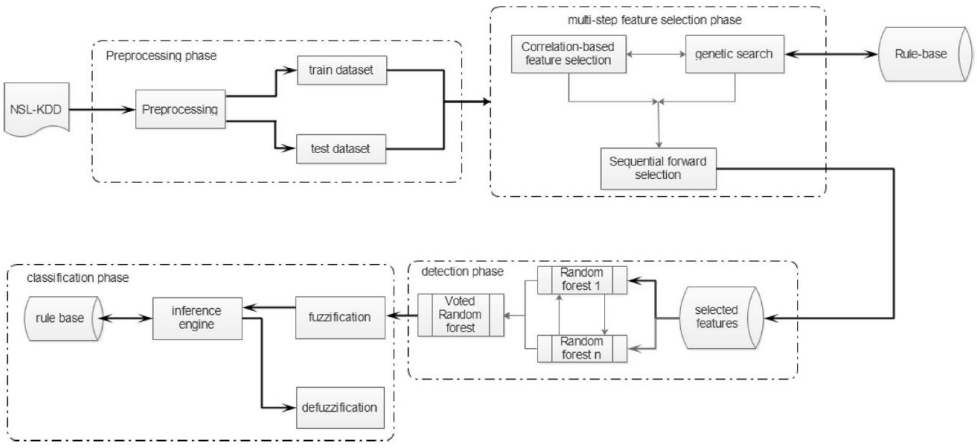


Figure 19: Proposed system by Awotune *et al.* [18]

Khraisat *et al.* [19] proposed a novel ensemble HIDS by combining a C5 classifier, and one class SVM classifier. Fig.20 shows the architecture of the proposed method. HIDS integrates the advantages of SIDS and AIDS. SIDS effectively identifies known attacks, whereas AIDS is adept at detecting zero-day attacks. In the SIDS phase, once a DT is created, it can be applied to detect other samples with varying success depending on how well it systems the dataset. The tree can then be applied as a rule set for detecting whether a test sample is malware or benign software. Unknown traffic is analyzed through pattern matching to ascertain whether it signifies normal or malicious behavior. Alerts are triggered for samples matching known attack signatures in the database, indicating a potential threat. Conversely, samples not matching any signature proceed to the AIDS phase for further evaluation. AIDS, trained with benign samples and informed by SIDS's output, excels in recognizing typical activities. This phase focuses on identifying standard behavior and labeling deviations as potential zero-day attacks. The primary goal of this framework is to achieve high accuracy in detecting both well-known intrusions and zero-day attacks while maintaining low false alarm rates. The proposed HIDS is evaluated using the Bot-IoT dataset, which includes legitimate IoT network

traffic, and several types of attack. The authors did not compare the performance of the proposed system with the state of the art papers. Table 2.1 summarizes the recent related works in terms of published year, category, IDS method, utilized dataset, and limitations of the proposed methods.

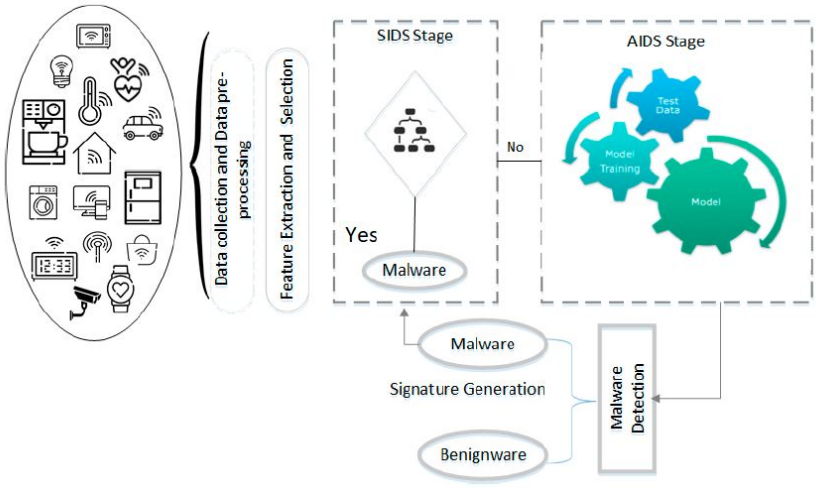


Figure 20: Proposed method by Khraisat *et al.* [19]

Table 2.1: Related works

Reference	Year	Category	IDS method	Dataset	Limitations
Javeed *et al.* [40]	2024	AIDS	CNN and BiLSTM	CICIDS2017 and EdgeIIoTset	CICIDS2017 is not an IoT dataset, Unbalanced dataset, Not profiled the system
Kumar *et al.* [4]	2024	AIDS	CNN	MalImg	The utilized dataset is not an IoT dataset, Limited number of samples in the dataset, types of malware is not detected, Not profiled the system
J.Rabie *et al.* [5]	2024	AIDS	Descriptive Back Propagated Radial Basis Function (DBRF)	IoTID20, Netflow-ToN-IoT, NSL-KDD, and UNSW-NB 15	Some of the utilized datasets are not IoT datasets, Limited number of data in each dataset, Not profiled the system
Awotune *et al.* [18]	2024	Hybrid	RF	NSL-KDD	Unbalanced dataset,Not utilized an IoT dataset
Mohy-eddine*et al.* [1]	2023	AIDS	KNN	Bot-IoT	Unbalanced datasets, Did the experiments with Limited number of records, Not profiled the system

Continued on next page

Table 2.1 – *Continued from previous page*

Reference	Year	Category	IDS method	Dataset	Limitations
Altuny *et al.*[6]	2023	AIDS	CNN, LSTM	UNSW-NB15, X-IIoTID	One of the utilized dataset is not a balanced dataset, Not profiled the system
Douiba *et al.* [2]	2022	AIDS	CatBoost	NSL- KDD, IoT-23, BoT-IoT, and Edge-IIoT	Not profiled the system
Ullah *et al.* [12]	2022	AIDS	DCNN	IoTID20	Not profiled the system
Khan *et al.* [7]	2022	AIDS	SOM	NSL-KDD, N-BaIoT	Unbalance dataset, One of the datasets is not an IoT dataset, Not profiled the system
Yang *et al.* [46]	2022	SIDS	CNN, BiLSTM	NSL-KDD	Not utilized an IoT dataset, Unbalance dataset, Not profiled the system
S *et al.*[36]	2021	AIDS	NB	UNSW-NB15	Not utilized an IoT dataset, Unbalanced dataset, Not profiled the system
Abbas *et al.* [3]	2021	AIDS	Logistic Regression,NB, and DT	CICIDS2017	Not utilized an IoT dataset, Unbalanced dataset, Not profiled the system
Zhao *et al.* [8]	2021	AIDS	GRU, and ResNet	NSL-KDD	Not utilized an IoT dataset, Not profiled the system
Ullah *et al.* [10]	2021	AIDS	GANs	KDD 99,NSL-KDD, BoT-IoT, IoT Network Intrusion, MQTT-IoT-IDS2020, MQTTset, IoT-23	Not profiled the system
Stivan *et al.*[37]	2021	AIDS	K-Means, Random Tree	Generated their own dataset	Limited size of the dataset, Unbalanced dataset, Limited number of features, Not profiled the system
Hu *et al.*[42]	2021	AIDS	Clustring	SLKDD, UNSW, AWID	Not utilized IoT datasets, Not profiled the system
Moghanian *et al.* [11]	2020	AIDS	Multilayer Perceptron(MLP)	KDD, UNSW	Not utilize an IoT dataset, Unbalanced dataset, Not profiled the system

Continued on next page

Table 2.1 – *Continued from previous page*

Reference	Year	Category	IDS method	Dataset	Limitations
Dushimimana*et al.* [9]	2020	AIDS	BRNN	KDD Cup 99	The utilized dataset is not an IoT dataset, Unbalanced dataset, Not profiled the system
Ferrag*et al.*[17]	2020	AIDS	REP Tree, JRip algorithm, and Forest PA	CICIDS2017, BoT-IoT	Not profiled the system
Khraisat *et al.* [19]	2019	Hybrid	DT,SVM	BoT-IoT	Not profiled the system
Elsaeidy*et al.*[14]	2019	AIDS	RBMs, FFNN automated FFNN, RF, and SVM	smart water distribution system dataset	Limited number of features, Not mention the number of records in the dataset, Not profiled the system
Telikani *et al.* [15]	2019	AIDS	auto-encoder	KDD CUP 99, NSL-KDD	Not utilized an IoT dataset, Not profiled the system
Balakrishnan *et al.* [16]	2019	AIDS	DBN	Not mentioned	Not profiled the system
Almiani *et al.* [13]	2019	AIDS	DNN	NSL-KDD	Not utilized an IoT dataset, Not profiled the system
Stylianopoulos *et al.* [47]	2019	SIDS	AC,DFC, PFAC, DFC, HYBRID	DARPA 2000	Not utilized an IoT dataset,Not profile the system
Fu *et al.*[48]	2017	SIDS	Finite State Automata	Event Database (Private database)	Not compare the proposed system performance with other systems, Not profiled the system

Based on this review, state-of-the-art IoT IDSs can be grouped into four major categories: AIDS, SIDS, Specification-based IDS, and HIDS. Fig. 21 presents a comprehensive taxonomy of IDSs in IoT networks, detailing the subcategories of each system. However, despite their effectiveness, we still face several challenges and issues, including:

- The lack of established systems for profiling malicious behaviors in IoT networks.
- The limitation of IoT computational and power resources, makes it challenging to implement sophisticated security solutions, including conventional IDSs.
- The vulnerability of existing IDSs to zero-day attacks.
- The Lack of diverse and high-quality labeled datasets covering a wide range of attacks and malware in binary and multi-class formats hinders effective system training.
- The lack of using balanced datasets for training systems, especially those sensitive to class imbalance (like KNN), can lead to biased results and poor performance in minority classes.

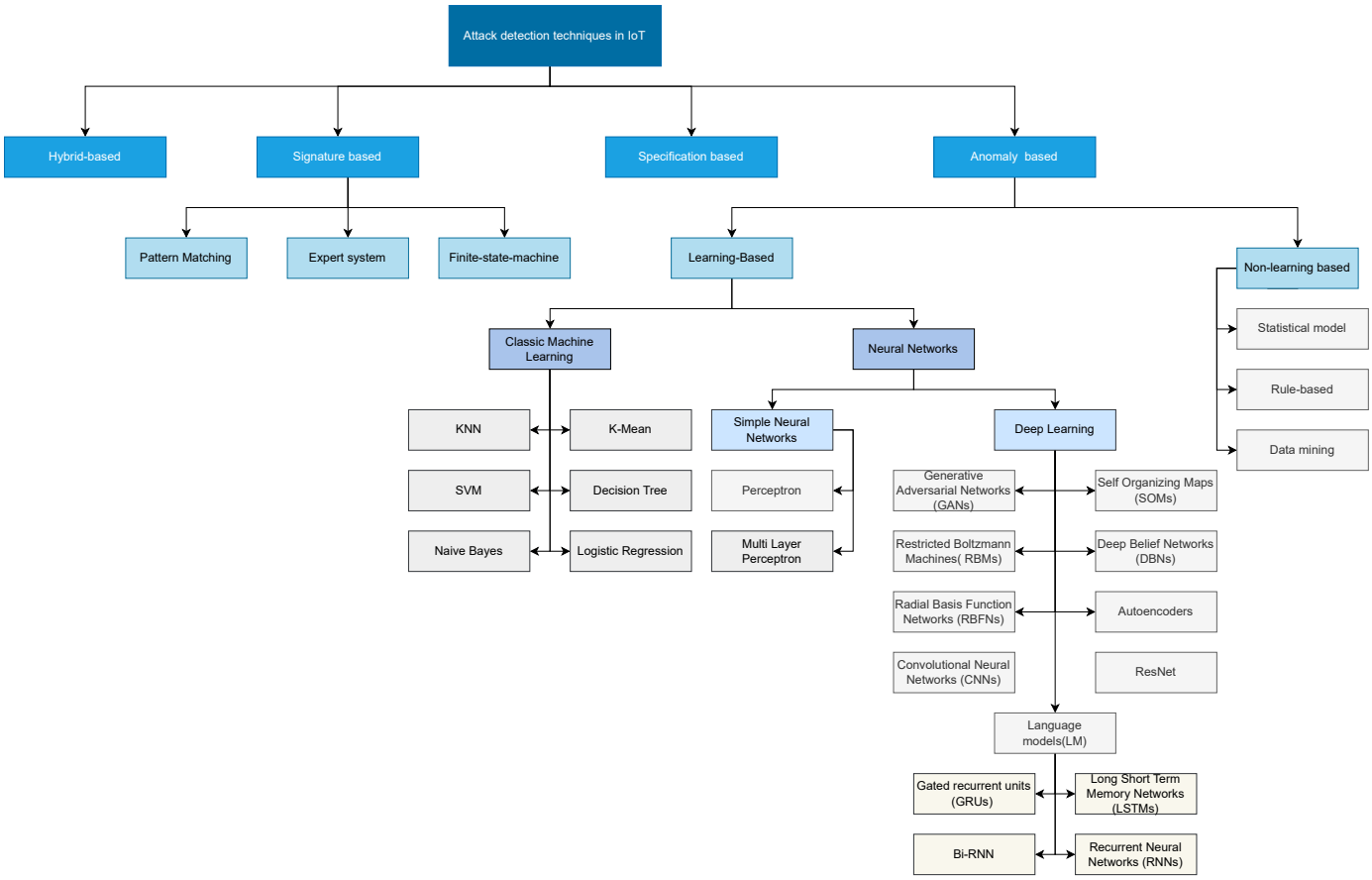


Figure 21: Proposed Taxonomy for IoT IDS

This study addresses the first, second, fourth, and fifth limitations by proposing an explainable AI-based model using LSTM and SHAP. Given the widespread impact of botnets as common intruders on IoT networks, the study focuses exclusively on the analysis, detection, and identification of botnet network traffic.

3 Proposed Model

LSTMs are particularly well-suited for processing time series data due to their ability to capture long-term dependencies within the sequence, often leading to improved prediction or recognition accuracy [52]. In this order this research designed a model based on LSTM for IoT Bot intrusion detection and identification. To provide a comprehensive understanding, we first offer an overview of LSTM in subsection 3.1. Subsequently, we introduce SHAP in subsection 3.2, which is used to examine the behavior of botnet samples both individually and collectively. Finally, we detail the proposed model architecture in subsection 3.3.

3.1 LSTM

LSTM, a special type of RNN, designed to solve the problem of "long-term dependencies" by incorporating a "gate" mechanism, enabling the retention of memory over extended periods [53]. A key component of an LSTM is the cell state, which acts as a memory that carries information across different time steps in the network. This cell state is crucial for preserving long-term dependencies in sequential data. LSTM typically includes three gates: the forget gate, the input gate, and the output gate, which together manage and safeguard the cell state. Fig. 22 shows the LSTM architecture diagram. The architecture of an LSTM cell state consists of the input vector x_t, the previous hidden state h_{t-1}, and the cell state C_t. The forget gate layer, which is a sigmoid layer, outputs values between 0 and 1, controlling the updated cell state C_t, derived from the old cell state C_{t-1}. This update process involves the input i_t and forget f_t gates [54, 53].

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

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{2}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{3}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{4}$$

$$h_t = o_t * \tanh(C_t) \tag{5}$$

Figure 22: LSTM Architecture Diagram

3.1.1 Parameters

Parameter selection in LSTM architectures profoundly impacts performance, especially accuracy [55]. However, there is scarce guidance on evaluating or choosing these parameters. Consequently, achieving optimal performance often relies on experience or trial and error. Key parameters include learning rate, dropout rate, optimizer, activation function, epoch, unit, batch size, loss function, hidden layers, classifier, encoder, and number of time steps [54, 56, 52].

(a) **Learning Rate**

The learning rate controls the step size for updating weights during training. A high learning rate speeds up convergence but risks instability, while a low rate ensures stable convergence but may slow training or cause it to get stuck in local minima. For LSTM systems, an optimal learning rate between 0.01 and 0.1 is crucial for good results [57].

(b) **Dropout**

Dropout is a regularization technique in DNNs that randomly deactivates neurons during training with a specified probability (dropout rate). This creates an ensemble of subnetworks, preventing overfitting and improving system generalization and robustness to unseen data. The dropout rate typically ranges from 0 to 1 [58, 56].

(c) **Optimizer**

Optimizer algorithms enhance deep learning system performance by updating parameters like weights and biases during training, guiding systems towards convergence and minimizing the loss function [59].They significantly affect training speed, convergence quality, and generalization. Common optimizers include SGD, Adam, RMSProp, Adagrad, Adadelta, and Nadam [57, 56, 60].

(d) **Activation function**

Activation functions in neural networks transform a neuron's output to the next layer's input, crucial for shaping outputs[61]. They decide neuron activation based on input relevance for the system's prediction [62], introducing non-linearity to system complex data relationships. Popular activation functions include Logistic Sigmoid, Tanh, and ReLU[61, 62].

(e) **Epoch**

An epoch refers to a complete iteration through the entire training dataset. During each epoch, the neural network processes all training examples once, updating its weights based on the computed gradients. Multiple epochs allow the system to learn from the data and refine its internal representations over time [63].

(f) **Units**

Neural units in neural networks are fundamental components that process and transform input data. These units form layers that constitute the network's architecture [64]. Each unit computes a weighted sum of inputs, applies an activation function, and produces an output, crucial for learning representations and making predictions or classifications [65].

(g) **Batch size**

In neural networks, the batch size is the number of samples processed before updating the system's weights and biases. During each training step, a batch of samples goes through the network, and gradients are calculated via backpropagation [66]. For example, with 1050 training samples and a batch size of 100, the network is trained on batches of 100 samples until all samples are used. Smaller batch sizes require less memory and allow faster training with more frequent weight updates but less accurate gradients. Larger batch sizes use more memory but provide more stable gradient estimates [67].

(h) **Loss function**

A loss function in neural networks measures the difference between the predicted output and the actual target value, guiding the optimization process to minimize this discrepancy. The goal is to find the weights and biases that result in the lowest average loss across the training data. Different loss functions are used for tasks like regression or classification [68].

(i) **Hidden layers**

A hidden layer in an artificial neural network resides between the input and output layers. Unlike the input and output layers, where direct interaction with them is possible, hidden layers remain concealed during usage. These layers consist of artificial neurons that process weighted inputs and produce outputs through activation functions [69].

(j) **Encoder**

In neural networks, an encoder refers to a specific part of the architecture responsible for taking input data and transforming it into a compressed, encoded representation. This encoded data often captures the most important features of the input [70].

(k) **Time steps**

A timestep in a neural network is the sequential step or time interval considered when processing input data. In an LSTM network, a timestep is crucial for systeming sequential data. An LSTM, a type of RNN, captures long-term dependencies in time series data, making it suitable for tasks like ID [71, 72].

3.2 Shapley Additive exPlanation (SHAP)

SHAP is a powerful approach that calculates the impact of each feature on the target variable. It treats each feature as a "player" and the entire dataset as a "team" [73]. The contributions of individual features are evaluated by considering all possible combinations of features and retraining the model accordingly. The resulting SHAP values represent the impact of each feature on the model's predictions [73]. One of the key advantages of SHAP is that it is model-agnostic. It can be applied to any machine learning approach, including neural networks. Unlike some model-specific techniques, SHAP provides a consistent way to explain feature importance across different models [74, 73].

When deploying deep learning approaches in critical applications, understanding the underlying reasons for predictions is essential for accountability and decision-making [75]. Neural networks, as one of the most common deep learning algorithms, while powerful, often act as "black boxes" for the lack of inherent feature importance explanations [76]. SHAP bridges this gap by allowing us to interpret neural network predictions. By calculating SHAP values, we can understand which features contribute most significantly to the network's output [74].

3.3 Proposed Model Architecture

The proposed model employs a novel feature selection technique combining XGBoost and a correlation-based method. Additionally, the model utilizes LSTM with Bayesian optimization on hyperparameter tuning

to enhance performance. Moreover, a novel behavioral profiling approach using SHAP is introduced, examining botnet samples' behavior individually and collectively. The model is evaluated on a balanced augmented IoT bots dataset to ensure fair and unbiased performance assessment. Fig. 23 illustrates the overall diagram of the proposed model, organized into four sequential phases: data preprocessing, feature selection, tuned LSTM based detection and identification, and behavior profiling. Each phase of the proposed model is discussed in subsequent subsections.

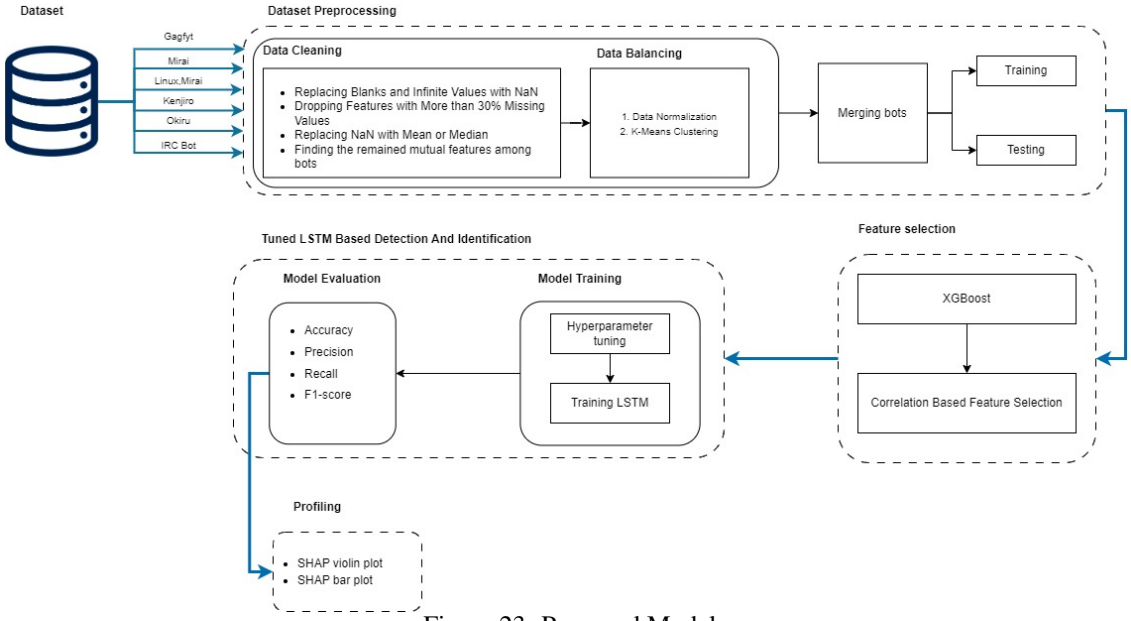


Figure 23: Proposed Model

3.3.1 Data Preprocessing

Data preprocessing removes noise and ensures data consistency, which is essential for improving data quality [77]. The proposed model handles NaN values by dropping features with over 30% missing data. It also ensures mutual features among all subsets. Data balancing is performed using Min-Max normalization and K-means clustering. Subsets of botnet samples are merged into a unified dataset, which is then split into training and testing sets for model evaluation.

3.3.2 Feature Selection

Feature selection is the process of selecting a subset of relevant features, minimizing computational costs and improving system performance [78]. In neural networks, it mitigates overfitting by retaining essential information and discarding irrelevant details [79, 80]. The proposed model uses the XGBoost algorithm to prioritize significant features and utilizes correlation coefficients between features to further enhance efficiency and reduce computational overhead. We will provide an overview of XGBoost and the concept of correlation coefficients, then detail the proposed feature selection technique.

- **XGBoost**: XGBoost employs L1 (Lasso) and L2 (Ridge) regularization, enhancing model general-

izability. It prunes trees during training to retain only relevant splits, improving both efficiency and accuracy [81]. XGBoost also parallelizes tree construction, offering significant speed advantages for large datasets [81, 82]. In XGBoost, feature importance is determined by how frequently each feature contributes to error reduction. This ranking aids in selecting relevant features for LSTM training, thereby improving performance and reducing noise [83, 82].

$$\text{Gain}(t) = \frac{1}{2}\left(\frac{\text{Gain}^2_{\text{left}}}{\text{Hessian}_{\text{left}}+\lambda} + \frac{\text{Gain}^2_{\text{right}}}{\text{Hessian}_{\text{right}}+\lambda} + \frac{\text{Gain}^2_{\text{node}}}{\text{Hessian}_{\text{node}}+\lambda}\right) - \gamma \tag{6}$$

  - $\text{Gain}_{\text{left}}$ and $\text{Gain}_{\text{right}}$ are the gains for the left and right child nodes, respectively.
  - $\text{Hessian}_{\text{left}}$ and $\text{Hessian}_{\text{right}}$ are the Hessians (second derivatives of the loss function) for the left and right child nodes, respectively.
  - $\text{Hessian}_{\text{node}}$ is the Hessian for the current node.
  - λ is the regularization term.
  - γ is the minimum loss reduction required for a split to occur.

- **Correlation coefficients**: Correlation measures the strength and direction of a linear relationship between two continuous variables. The correlation coefficient ranges from -1 (perfect negative correlation) to +1 (perfect positive correlation), with zero indicating no linear association [84]. A positive coefficient means both variables increase together, while a negative coefficient indicates that as one variable increases, the other decreases [85]. Collinearity occurs when predictor variables are highly correlated, potentially distorting the results of regression analyses. To avoid collinearity, pairwise correlations among independent variables should be below 0.9 [86, 87]. Common methods for calculating correlation include Pearson's and Spearman's coefficients [88, 89].

The proposed feature selection technique consists of two sequential steps. First, the XGBoost algorithm evaluates the significance of each feature and ranks them by importance. In the subsequent phase, correlation coefficients between features are computed. Features with a correlation coefficient exceeding 0.9 are compared by importance, and the less significant one is removed. As a result, a feature set composed of the most essential features, with pairwise correlations below 0.9, is derived. This research uses Pearson's coefficient to identify linear relationships in the data, as it determines the presence of a linear relationship [90, 20].

3.3.3 Tuned LSTM Based Detection And Identification

After selecting the most critical features, the proposed model employs a tuned-based detection method to identify intrusions. This phase consists of model training and evaluation. As shown in Fig. 23, during the model training phase, the LSTM model undergoes tuning. The tuning process involves selecting optimal values for learning rate and batch size parameters. After tuning, the LSTM model is trained and evaluated using metrics such as accuracy, precision, recall, and F1-score.

3.3.4 Behavior Profiling

The final phase of the proposed model focuses on behavior profiling, a critical component for enhancing network security by accurately identifying and understanding malicious activities. Profiling involves the systematic analysis of network traffic to identify patterns and behaviors indicative of potential threats. By establishing profiles of malicious and benign behavior, systems can better detect and respond to security breaches, unauthorized access, and other cyber threats[20].
To profile malicious network traffic behavior, we employed a defined profiling system based on attribute, relation, and entity extraction[20]. This approach allows for a nuanced understanding of how different factors contribute to network anomalies and potential intrusions:

- **Attribute Extraction**:This step involves identifying the key features or attributes of the model, which are crucial for interpretability and analysis. Attributes may include packet size, frequency of requests, source and destination IP addresses, and other relevant metrics that characterize network traffic. By focusing on these attributes, we can create a detailed profile that highlights deviations from expected behavior, which may indicate malicious activity.

- **Relation Extraction**:In this phase, we analyze the interactions between different features to understand their combined impact on the system's output. For instance, a high frequency of requests combined with unusual packet sizes might suggest a DoS attack. By examining these relationships, we can identify complex patterns that simple attribute analysis might miss, thereby improving the accuracy of threat detection.

- **Entity Extraction**:This process identifies the most influential features that distinguish different labeled data points. Entities could include specific types of malware or attack signatures that are particularly significant in differentiating between benign and malicious traffic. Understanding these entities helps in pinpointing the specific characteristics of threats, allowing for more targeted and effective responses.

This research utilized SHAP bar and violin plots to profile individual and common bot characteristics. SHAP values provide insights into how each feature contributes to the prediction outcome, enabling a more transparent understanding of the model's decision-making process:

- **SHAP Bar Plots**:These plots effectively compare the importance of different features by displaying their average impact on the model's output. By visually highlighting these differences, security analysts can prioritize the features that have the most significant influence on detecting malicious activity. This prioritization is crucial for refining detection algorithms and focusing resources on the most critical threat vectors.

- **SHAP Violin Plots**: These plots offer a more detailed distribution of the data, showing both the density and variability of feature impacts. The violin plots aid in understanding the spread and concentration of feature impacts across different instances. This visualization helps in identifying not just which

features are important, but also how consistent their impact is across various samples. This consistency is vital for developing robust detection mechanisms that are resilient to variations in attack patterns.

By using these plots, we identify the most significant features and their combined effects on the system. This comprehensive profiling of botnet samples' behavior, both individually and collectively, enhances our ability to detect and respond to threats.

4 Experimental setup

The advancement of learning-based IDSs depends significantly on the quality of datasets [91]. Since learning-based IoT IDSs are evaluated using these datasets, understanding their strengths and weaknesses is crucial for researchers. In this context, subsection 4.1 analyzes prominent publicly available IoT datasets used to evaluate IoT IDSs, examines their limitations, and selects the most comprehensive one, named Aposemat-Bot-IoT-23 [92], for an in-depth analysis. To address some of the limitations of the selected dataset, we generated an augmented dataset named BCCC-Aposemat-Bot-IoT-24. We thoroughly analyzed and explained the process of generating this dataset. Additionally, to avoid bias towards the majority class during training and evaluation of the proposed model, subsection 4.2 reviews different sampling techniques in the literature and selects the most suitable one based on the nature of the generated augmented dataset. Finally, subsection 4.3 details the platforms and tools used for model implementation.

4.1 Dataset

The proliferation of IoT IDSs has made them targets for malicious third parties. It is essential to develop effective protection and investigation countermeasures to address this challenge, such as IDSs and network forensic systems. A key aspect of this development is having a well-structured and representative dataset for training and validating these systems. The analysis of the available datasets used for evaluating IoT IDSs, detailed in subsection 4.1.1, has identified the Aposemat-Bot-IoT-23 dataset as a standout. This dataset, which is one of the most comprehensive publicly available IoT datasets, is a crucial resource for researchers in the field. It contains both malicious and benign network traffic, providing a well-rounded view of IoT network behavior. We delve into a detailed analysis of this dataset in subsection 4.1.2. Subsection 4.1.3 then introduces the augmented BCCC-Aposemat-Bot-IoT-24 dataset, which we have developed to address some issues found in the original Aposemat-Bot-IoT-23 dataset.

4.1.1 Available datasets

This section analyzes prominent publicly available IoT datasets used in IoT IDSs. While previous datasets have driven IoT IDSs research, they also have limitations. Addressing these limitations can help create better datasets, facilitating more effective IoT attack detection techniques. Table 4.1 overviews recent IoT datasets. Source IP, destination IP, source port, destination port, and timestamp were not considered in the total number of features. The limitations identified in previous datasets are as follows:

- **Lack of Realistic Network Traffic**: Accurate IoT IDS need training with realistic network traffic patterns.Without this, datasets do not represent real-world behavior, limiting detection methods' effectiveness.
- **Dataset Size and Attack Type Constraints**: Many IoT datasets are small and lack diversity in attack types, making it difficult to train and accurately evaluate detection systems.
- **Labeling Accuracy Shortcomings**: Inaccurate data labeling undermines dataset reliability and detection system effectiveness. Existing IoT datasets often lack essential details like IP addresses, ports,

MAC addresses, and precise attack timings, hindering dataset generation from PCAP files and further research.

- **Documentation Chaos**: Many IoT datasets lack organized documentation, missing crucial details such as IP addresses, ports, MAC addresses, and attack timings. This absence hampers dataset creation from PCAP files and impedes research.
- **Outdated Attack Scenarios**: Limited and outdated attack scenarios in datasets make detection systems ineffective against emerging threats. Continuously updated datasets are needed to reflect the evolving landscape of IoT attacks.
- **Underutilization of User Behavior**: User behavior is crucial for detecting IoT attacks but often missing from datasets. Including user behavior data enhances training realism and detection system effectiveness.
- **Addressing Modern Protocols Incompatibilities**: Modern protocols like Zigbee, Z-Wave, and MQTT are underrepresented in training data. Existing datasets are often small and limited in attack types, hindering robust detection system development. Comprehensive datasets covering modern protocols and diverse attacks are crucial for securing the growing IoT landscape.
- **Imbalanced Class Distribution**: Imbalanced attack classes in IoT datasets challenge effective IDS development. This bias can cause systems to miss rarer, more dangerous threats. Addressing this requires strategies like oversampling, cost-sensitive learning, and using metrics such as precision, recall, and AUC-ROC. Balancing classes is essential for robust IoT security.

Based on the analysis above, and the datasets presented in Table 4.1, we selected the Aposemat IoT-23 dataset [92] for further analysis in this research. This dataset addresses nearly half of the 10 issues mentioned previously. Subsection 4.1.2 provides a more detailed discussion of the Aposemat IoT-23 dataset [93].

Index	**Name**	**Features Count**	**Format**			**Organized Documentation**	**Size**	
			CSV	**Log**	**PCAP**		**Small**	**Large**
1	IoTID20 [94]	76	✓	✗	✗	✗	✓	✗
2	IoT Network Intrusion Dataset [95]	-	✗	✗	✓	✗	✓	✗
3	BoT-IoT [96]	42	✓	✗	✓	✗	✗	✓
4	ToN-IoT [97]	44	✓	✓	✓	✓	✗	✓
5	NF-ToN-IoT [98]	8	✓	✗	✗	✓	✓	✗
6	NF-ToN-IoT-v2 [99]	39	✓	✗	✗	✓	✗	✓

Index	Name	Features Count	Format			Organized Documentation	Size	
			CSV	Log	PCAP		Small	Large
7	NF-BoT-IoT [98]	8	✓	✗	✗	✓	✓	✗
8	NF-BoT-IoT-v2 [99]	39	✓	✗	✗	✓	✗	✓
9	CIC-ToN-IoT [100]	75	✓	✗	✗	✓	✗	✓
10	CIC-BoT-IoT [100]	75	✓	✗	✗	✓	✗	✓
11	X-IIoTID [101]	59	✓	✗	✓	✗	✓	✗
12	WUSTL-IIoT-2021 [102]	41	✓	✗	✗	✓	✓	✗
13	Aposemat IoT-23 dataset [92]	15	✗	✓	✓	✗	✓	✗
14	Edge-IIoT [103]	58	✓	✗	✓	✓	✗	✓
15	N-BaIoT [104]	115	✓	✗	✓	✗	✗	✓
16	MedBIoT [105]	20	✓	✗	✓	✗	✓	✗
17	IOT-ENVIRONMENT-DATASET [106]	-	✗	✗	✓	✗	✓	✗
18	MQTT-IoT-IDS2020 [107]	64	✓	✗	✓	✗	✓	✗
19	Federated ToN-IoT [108]	258	✓	✓	✗	✓	✓	✗
20	CIC IoT Dataset 2022 [109]	48	✓	✗	✓	✗	✓	✗
21	CIC IoT Dataset 2023 [110]	46	✓	✗	✓	✗	✓	✗
22	Dataset of legitimate IoT data VARIoT [111]	77	✓	✗	✓	✗	✓	✗
23	TCP FIN Flood and Zbassocflood Dataset [112]	146	✓	✗	✓	✗	✓	✗

Index	Name	Features Count	Format			Organized Documentation	Size	
			CSV	Log	PCAP		Small	Large
24	IoT Healthcare Security Dataset [113]	46	✓	✗	✗	✗	✓	✗
25	A Dataset Bundle for Building Automation and Control Systems Security Analysis [114]	20	✓	✗	✓	✗	✓	✗

Table 4.1: Overview of IoT Datasets and their limitations

4.1.2 Aposemat IoT-23

The Aposemat-Bot IoT-23 dataset is a resource for studying IoT network traffic, consisting of 23 scenarios representing various types of IoT device communication. These scenarios are categorized into malicious (20) and benign (3). The malicious scenarios capture traffic from compromised IoT devices generated by infecting Raspberry Pi devices with specific malware samples. The benign scenarios capture real traffic from everyday IoT devices, including a Philips Hue smart LED lamp, an Amazon Echo smart speaker, and a Somfy smart door lock [93].

Table 4.2 details each scenario's characteristics, including whether the traffic is malicious or benign, the associated malware family, specific malware sample name, the individual file size of each malware sample, and the total size of both benign and malware capture files. For example, one scenario captures traffic from a device infected with the Linux Hajime variant (472 MB), and another captures the Hide and Seek variant (140 MB), totaling approximately 612 MB [93].

Network traffic features were extracted from the original PCAP files using the Zeek network traffic analyzer, resulting in conn.log files with 15 features extracted from the data [93]. Despite its comprehensiveness, the Aposemat-Bot IoT-23 dataset has some limitations, including:

- **Limited number of extracted features by Zeek analyzer**:
  Only 15 features were extracted by Zeek network traffic analyzer.
- **Inclusion of Benign Network Traffic:**
  It is essential to recognize that the captured malicious network traffic does not solely consist of malicious network traffic. Benign network traffic is also present. Researchers and analysts should consider this mix when extracting features from the malware pcap files.
- **Proportion of Captured Malicious to Benign Network Traffic:**
  Table 4.2 highlights a significant gap between the proportion of captured malicious and benign network traffic. This disparity is crucial for understanding the prevalence of different types of network activity.

- **Size Discrepancies Among Malware Families:**
  When analyzing malicious samples, we observe substantial differences in file sizes across various families. Specifically:

  – For Worm malware, two scenarios were dedicated to capturing data. The total size of captured Worm traffic amounts to approximately 612 MB.

  – In contrast, 17 scenarios were dedicated to capturing Bots malware. The total size of captured Bots malware traffic is a staggering 58.4 GB. This substantial difference underscores the varying impact of different malware families.

  – Even within the Bots family, there are significant variations. For instance, the captured Hakai sample is relatively small at 2.1 MB, while the captured Mira sample is much larger, totaling 18.5 GB.

4.1.3 BCCC-Aposemat-Bot-IoT-24

In order to address some of the issues mentioned in the Aposemat-Bot-IoT-23 dataset [93], we generated an augmented BCCC-Aposemat-Bot-IoT-24 dataset. This new dataset addresses the first and second issues of the original Aposemat-Bot-IoT-23 dataset. Due to the limited size of captured Worm and Trojan malware compared to Bots and Benign network traffic, we excluded them from the generated dataset. Consequently, the generated dataset comprises only various bot samples and benign data. For generating the augmented BCCC-Aposemat-Bot-IoT-24 dataset, we employed the NTLFlowlyzer network traffic analyzer [115]. We used this analyzer to extract features from the PCAP files. NTLFlowlyzer is an open-source Python project for anomaly profiling in TCP-based network traffic. It generates bidirectional flows from the network and transport layers, allowing statistical time-related features to be calculated separately for forward (source to destination) and backward (destination to source) directions. The tool provides additional functionalities, such as feature selection, adding new features, and controlling flow timeout duration. NTLFlowlyzer extracts 315 features, categorized into packet-based, status-based, header-based, payload-based, flag-based, and side-based features(Table 4.4). The original captured malicious PCAP files of the Aposemat-Bot-IoT-23 dataset contain malicious and benign network packets. To accurately label the network traffic, we leveraged log files for each scenario, which labeled certain traffic flows as malicious or benign. Since Zeek considers both directions within a connection (including SYN, SYN-ACK, and ACK packets), it creates bidirectional TCP flows [116]. Given that NTLFlowlyzer also generated bidirectional flows [115], we were able to label the generated CSV files by NTLFlowlyzer based on the log files generated by the Zeek network traffic analyzer, utilizing the source port, destination port, source IP, and destination IP. The dataset generated by NTLFlowlyzer has two label columns: ”Binary Classification” and ”Sample Classification”. ”Binary classification” labels are ”Malicious” or ”Benign”. Sample classification labels are either blank (-), indicating ”Benign” data, or the specific bot sample name, such as Muhstik, Hakai, Okiru, IRCBot, Kenjiro, Torii, Linux Mira, or Mirai. Algorithm 1 further details the devised strategy, as outlined below.

(a) Data Preprocessing:

Index	File name	Binary	Family	Sample	Sample PCAP size	Family PCAP size	Binary PCAP size
1	CTU-Honeypot-Capture-4-1	Benign	-	-	4,549 KB	387.6 MB	387.6 MB
2	CTU-Honeypot-Capture-5-1				381 MB		
3	CTU-Honeypot-Capture-7-1				2,094 KB		
4	CTU-IoT-Malware-Capture-9-1	Malware	Worm	Linux, Hajime	472 MB	612 MB	59.0 GB
5	CTU-IoT-Malware-Capture-1-1			Hide and Seek	140 MB		
6	CTU-IoT-Malware-Capture-42-1		Trojan	Trojan	2.8 MB	2.8 MB	
7	CTU-IoT-Malware-Capture-3-1		Bots	Mushtik	56 MB	56 MB	
8	CTU-IoT-Malware-Capture-8-1			Hakai	2.1 MB	2.1 MB	
9	CTU-IoT-Malware-Capture-36-1			Okiru	992 MB	992 MB	
10	CTU-IoT-Malware-Capture-39-1			IRC Bot	5.3 GB	5.3 GB	
11	CTU-IoT-Malware-Capture-17-1			Kenjiro	7.8 GB	11.7 GB	
12	CTU-IoT-Malware-Capture-33-1				3.9 GB		
13	CTU-IoT-Malware-Capture-20-1			Torii	3.9 MB	7.8 MB	
14	CTU-IoT-Malware-Capture-21-1				3.9 MB		
15	CTU-IoT-Malware-Capture-7-1			Linux, Mirai	897 MB	897 MB	
16	CTU-IoT-Malware-Capture-60-1			Gagfyt	21 GB	21 GB	
17	CTU-IoT-Malware-Capture-34-1			Mirai	121 MB	18.5 GB	
18	CTU-IoT-Malware-Capture-35-1				3.6 GB		
19	CTU-IoT-Malware-Capture-43-1				6 GB		
20	CTU-IoT-Malware-Capture-44-1				1.7 GB		
21	CTU-IoT-Malware-Capture-48-1				1.2 GB		
22	CTU-IoT-Malware-Capture-49-1				1.3 GB		
23	CTU-IoT-Malware-Capture-52-1				4.6 GB		

Table 4.2: Summary of Aposemat-BotIoT-23 dataset scenarios

- Convert Zeek-generated log files for each scenario into CSV format.
- Filter the Zeek-generated CSV file to include only flows using the TCP protocol (as NTLFlowLyzer is TCP-based).

(b) Flow Matching and Labelling:

- Iterate through each flow extracted by NTLFlowLyzer.
- For each flow, check if its source port, destination port, source IP, and destination IP match any flow in the filtered Zeek CSV.
  - If the matched Zeek flows have a unique label (e.g., 'Malicious' or 'Benign'):
    - Assign the same label to the NTLFlowLyzer flow.
      - For "Malicious" label:
      (a) Set "Binary Classification" to "Malicious".
      (b) Set "Sample Classification" to the bot name retrieved from the Aposemat-Bot IoT-23 website [93].
      - For "Benign" label:
      (a) Set "Binary Classification" to "Benign".
      (b) Set "Sample Classification" to blank ,"-",.
    - If the matched Zeek flows does not have a unique label (e.g., both 'Malicious' and 'Benign')
    (a) Set "Binary Classification" to "Suspicious".
    (b) Set "Sample Classification" to blank "-".
  - If no matching flow is found in the Zeek data:
    - Set "Binary Classification" to "Suspicious".
    - Set "Sample Classification" to blank "-".

(c) Suspicious Flow Removal:

- After labeling all flows in the NTLFlowLyzer CSV, remove any flows with the label "Suspicious" from the final output.

(d) Benign flows Removal:

- After labeling all files generated by NTLFLowlyzer, the rows labeled as "Benign" from all CSV files are dropped and saved into a separate CSV file.
- Merge all the CSV files whose rows are labeled as Benign.

The labeling process ensures all generated files have a unique classification label of either "Malicious" or "Benign" (refer to Algorithm 1 for details). Table 4.3 details the dataset composition, showing a significant imbalance between the number of "Malicious" and "Benign" labeled rows.

Algorithm 1 NTLFlowLyzer Labelling with Classification Details

```
Convert generated log files into CSV format.
Filter flows with TCP protocol in Zeek-generated CSV file.
for Each flow extracted by NTLFlowLyzer do
    Check match of source port, destination port, source IP, destination IP with Zeek flows.
    if match found then
        if Zeek flows have a unique label (e.g., Malicious or Benign) then
            Assign the same label to NTLFlowLyzer flow.
            if label is ”Malicious” then
                Set ”Binary Classification” to ”Malicious”.
                Set ”Sample Classification” to the Bot sample’s name from Aposemat-Bot IoT-23.
            else if label is ’Benign’ then
                Set ”Binary Classification” to ”Benign”.
                Set ”Sample Classification” to Blank ”-”.
            end if
        else if Zeek flows have different labels then
            Set ”Binary Classification” to ”Suspicious”.
            Set ”Sample Classification” to Blank ”-”.
        end if
    else
        Set ”Binary Classification” to ”Suspicious”.
        Set ”Sample Classification” to Blank ”-”.
    end if
end for
Remove flows labeled as Suspicious from the generated CSV by NTLFlowLyzer.
Drop the rows marked as ”Benign” from all CSV files and save them into a separate CSV file.
Merge all the CSV files whose rows have Label Benign.
```

Index	File name	Binary	Family	Sample	Number of Records per Scenario	Total per Sample	Total
1	Generated Benign File	Benign	-	-	24,528,833	24,528,833	24,528,833
2	CTU-IoT-Malware-Capture-3-1	Malware	Bots	Mushtik	84,814	84,814	235,895,020
3	CTU-IoT-Malware-Capture-8-1			Hakai	2056	2056	
4	CTU-IoT-Malware-Capture-36-1			Okiru	13,634,438	13,634,438	
5	CTU-IoT-Malware-Capture-39-1			IRC Bot	73,561,853	73,561,853	
6	CTU-IoT-Malware-Capture-17-1			Kenjiro	2,753,487	55,795,486	
7	CTU-IoT-Malware-Capture-33-1				53,041,997		
8	CTU-IoT-Malware-Capture-20-1			Torii	7	14	
9	CTU-IoT-Malware-Capture-21-1				7		
10	CTU-IoT-Malware-Capture-7-1			Linux, Mirai	11,337,694	11,337,694	
11	CTU-IoT-Malware-Capture-60-1			Gagfyt	3,578,485	3,578,485	
12	CTU-IoT-Malware-Capture-34-1			Mirai	4370	77,900,180	
13	CTU-IoT-Malware-Capture-35-1				7,002,457		
14	CTU-IoT-Malware-Capture-43-1				46,673,545		
15	CTU-IoT-Malware-Capture-44-1				13		
16	CTU-IoT-Malware-Capture-48-1				1,690,835		
17	CTU-IoT-Malware-Capture-49-1				5,201,628		
18	CTU-IoT-Malware-Capture-52-1				17,327,332		

Table 4.3: BCCC-Aposemat-Bot IoT-24 dataset Malware distribution

4.2 Sampling Technique

Machine learning and deep learning's growing role in automating processes based on data comes with challenges. Training these models often assumes a balanced distribution of data samples [117, 118]. However, when data is unevenly distributed, model accuracy and reliability can suffer. Imbalanced datasets, where one class has significantly fewer samples, are common in classification problems [119]. This necessitates strategies to address this issue. In real-world scenarios, imbalanced data can lead to the minority class being ignored as noise. Consequently, machine learning models become biased towards the majority class, increasing FP and decreasing TP [120]. The literature proposes several solutions to mitigate this problem [121, 122, 123]. Approaches to address imbalanced data challenges can be categorized into two main areas [118, 123]:

- Data-Level Preprocessing: This involves applying algorithms to manipulate the data itself. Techniques include US, OS,or a combination of both.
- Algorithmic-Level Processing: This focuses on using algorithms specifically designed for imbalanced data, such as cost-sensitive systems and ensemble machine-learning systems.

Traditional machine learning and deep learning approaches struggle with datasets where one class (the majority class) has significantly more examples than another class (the minority class). This imbalance makes it difficult for systems to learn effectively from the minority class data [118].
There are three main categories of resampling methods: US, OS, and HS [123, 119]. The US category reduces the majority class dataset, while the OS category enlarges the dataset of the minority class [124]. The HS method utilizes both OS and US approaches [125]. There are different methods within each category. Fig. 24 shows a taxonomy of resampling methods. Definitions for each method are outlined below:

- **Random US:** This method randomly removes instances from the majority class until the dataset is balanced [126, 119]. However, this method has the downside of losing important information needed for learning when deleting instances from the majority class data [127].
- **CBUS**: This method groups majority class instances into clusters and then removes a specific number of clusters to match the minority class size [128]. It offers two selection methods: using the cluster center or the nearest neighbor of the center as the representative for removal [119].
- **OSS**: This method focuses on removing redundant or borderline instances from the majority class, aiming for a more informative reduction [129].
- **Random OS:** This method randomly duplicates minority class instances to increase their representation [119].
- **ROSE:** This method utilizes smoothed bootstrap methods to create new minority class instances [130].
- **SMOTE:** This method generates synthetic minority class instances by interpolating between existing minority class instances and their KNN [131]. Several variations of SMOTE exist, such as SMOTEBoost [132], MSMOTE [133], and MWMOTE [134].

- **Random HS:** This method combines random OS and US to achieve balance [125].
- **CBOUS:** This method extends CBUS by creating clusters for both majority and minority classes. It then removes majority class clusters and generates synthetic minority class instances using methods like SMOTE to achieve balance [135].

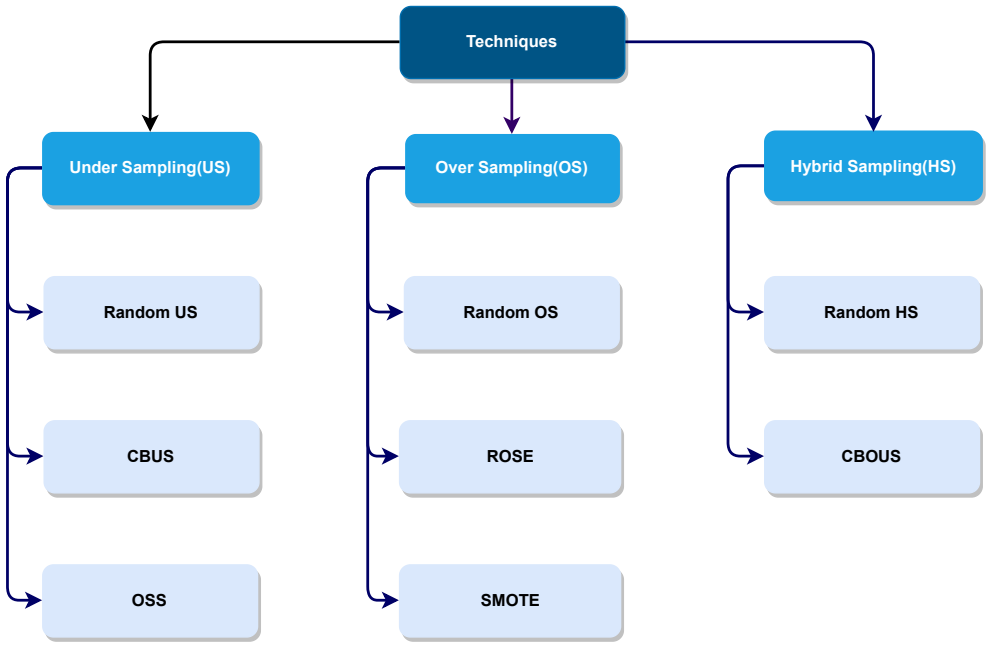


Figure 24: Taxonomy of the Available Resampling Techniques [20]

Table 4.3 shows a significant class imbalance in the BCCC-Aposemat-Bot-IoT-24 dataset. For example, Mirai has 77,900,180 records, while Gagfyt has only 3,578,485. This imbalance makes upsampling methods like SMOTE unsuitable, as SMOTE creates excessive synthetic data, potentially leading to poor generalization. The total number of malicious bot records 235,895,020 also poses computational challenges for training and testing, particularly with resource-intensive LSTMs. To address this, we used CBUS to preserve the data's structure while reducing its size.

To prepare the data for clustering, we performed Min-Max normalization on chunks of 20 million rows each. This method addressed the computational limitations due to the large number of rows in some bot samples (e.g., Mirai) and the overall size of the dataset. Equation 7 shows how Min-Max normalization scales each value X between 0 and 1, where X_{min} and X_{max} represent the minimum and maximum values in the data, respectively.

$$X_{\text{normalized}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} \tag{7}$$

Following normalization, we employed the K-means clustering algorithm. Due to computational constraints, we used 10 clusters for every 1 million rows. Subsequently, we undersampled each bot sample class by

selecting 2 million rows. For bot samples like Mirai, which consist of multiple CSV files, we calculated the proportion of rows in each file relative to the total number of rows. This ensured proportional representation when selecting the final 2 million rows from each bot sample. Finally, the selected rows from all samples were merged into a single CSV file, maintaining a balanced proportion of 2 million entries for each bot sample class.

4.3 Model implementation

For model evaluation, we used AWS cloud platforms, including Amazon SageMaker [136] and S3 storage [137]. SageMaker simplifies building, training, and deploying systems, offering a user interface and IDE integration [136]. We implemented the model using Keras [56] with a TensorFlow backend in Python. Data storage was managed by S3, ensuring scalability, security, and performance.
We utilized the "ml.g4dn.16xlarge" instance type [138], an EC2 instance [139] with 4 NVIDIA Tesla T4 GPUs, 256 GiB memory, 64 vCPUs, fast networking, and NVMe SSD storage. This setup supports efficient deep learning training, low-latency inference, large-scale data processing, and demanding simulations [140].

Table 4.4: NTLFlowLyzer features

Behavior	Feature	Feature Name
Packet based	F1	packets_IAT_mean
	F2	packet_IAT_std
	F3	packet_IAT_max
	F4	packet_IAT_min
	F5	packet_IAT_total
	F6	packets_IAT_median
	F7	packets_IAT_skewness
	F8	packets_IAT_cov
	F9	packets_IAT_mode
	F10	packets_IAT_variance
	F11	fwd_packets_IAT_mean
	F12	fwd_packets_IAT_std
	F13	fwd_packets_IAT_max
	F14	fwd_packets_IAT_min
	F15	fwd_packets_IAT_total
	F16	fwd_packets_IAT_median
	F17	fwd_packets_IAT_skewness
	F18	fwd_packets_IAT_cov
	F19	fwd_packets_IAT_mode
	F20	fwd_packets_IAT_variance
	F21	bwd_packets_IAT_mean
	F22	bwd_packets_IAT_std
	F23	bwd_packets_IAT_max
	F24	bwd_packets_IAT_min
	F25	bwd_packets_IAT_total
	F26	bwd_packets_IAT_median
	F27	bwd_packets_IAT_skewness
	F28	bwd_packets_IAT_cov
	F29	bwd_packets_IAT_mode
	F30	bwd_packets_IAT_variance
	F31	avg_fwd_packets_per_bulk
	F32	avg_bwd_packets_bulk_rate
	F33	fwd_bulk_per_packet
	F34	bwd_bulk_per_packet
	F35	min_bwd_packets_delta_time
	F36	max_bwd_packets_delta_time
	F37	mean_packets_delta_time

Table 4.4: NTLFlowLyzer extracted features (continued)

Behavior	Feature	Feature Name
Packet based	F38	mode_packets_delta_time
	F39	variance_packets_delta_time
	F40	std_packets_delta_time
	F41	median_packets_delta_time
	F42	skewness_packets_delta_time
	F43	cov_packets_delta_time
	F44	mean_bwd_packets_delta_time
	F45	mode_bwd_packets_delta_time
	F46	variance_bwd_packets_delta_time
	F47	std_bwd_packets_delta_time
	F48	median_bwd_packets_delta_time
	F49	skewness_bwd_packets_delta_time
	F50	cov_bwd_packets_delta_time
	F51	min_fwd_packets_delta_time
	F52	max_fwd_packets_delta_time
	F53	mean_fwd_packets_delta_time
	F54	mode_fwd_packets_delta_time
	F55	variance_fwd_packets_delta_time
	F56	std_fwd_packets_delta_time
	F57	median_fwd_packets_delta_time
	F58	skewness_fwd_packets_delta_time
	F59	cov_fwd_packets_delta_time
	F60	min_packets_delta_len
	F61	max_packets_delta_len
	F62	mean_packets_delta_len
	F63	mode_packets_delta_len
	F64	variance_packets_delta_len
	F65	std_packets_delta_len
	F66	median_packets_delta_len
	F67	skewness_packets_delta_len
	F68	cov_packets_delta_len
	F69	min_bwd_packets_delta_len
	F70	max_bwd_packets_delta_len
	F71	mean_bwd_packets_delta_len
	F72	mode_bwd_packets_delta_len
	F73	variance_bwd_packets_delta_len
	F74	std_bwd_packets_delta_len

Table 4.4: NTLFlowLyzer extracted features (continued)

Behavior	Feature	Feature Name
Packet based	F75	median_bwd_packets_delta_len
	F76	skewness_bwd_packets_delta_len
	F77	cov_bwd_packets_delta_len
	F78	min_fwd_packets_delta_len
	F79	max_fwd_packets_delta_len
	F80	mean_fwd_packets_delta_len
	F81	mode_fwd_packets_delta_len
	F82	variance_fwd_packets_delta_len
	F83	std_fwd_packets_delta_len
	F84	median_fwd_packets_delta_len
	F85	skewness_fwd_packets_delta_len
	F86	cov_fwd_packets_delta_len
	F87	packets_count
	F88	fwd_packets_count
	F89	bwd_packets_count
	F90	packets_rate
	F91	bwd_packets_rate
	F92	fwd_packets_rate
Status based	F93	duration
	F94	active_min
	F95	active_max
	F96	active_mean
	F97	active_std
	F98	active_median
	F99	active_skewness
	F100	active_cov
	F101	active_mode
	F102	active_variance
	F103	idle_min
	F104	idle_max
	F105	idle_mean
	F106	idle_std
	F107	idle_median
	F108	idle_skewness
	F109	idle_cov
	F110	idle_mode
	F111	idle_variance

Table 4.4: NTLFlowLyzer extracted features (continued)

Behavior	Feature	Feature Name
Status based	F112	handshake_duration
	F113	handshake_state
	F114	bwd_bulk_duration
	F115	fwd_bulk_duration
Header based	F116	total_header_bytes
	F117	max_header_bytes
	F118	min_header_bytes
	F119	mean_header_bytes
	F120	std_header_bytes
	F121	median_header_bytes
	F122	skewness_header_bytes
	F123	cov_header_bytes
	F124	mode_header_bytes
	F125	variance_header_bytes
	F126	fwd_total_header_bytes
	F127	fwd_max_header_bytes
	F128	fwd_min_header_bytes
	F129	fwd_mean_header_bytes
	F130	fwd_std_header_bytes
	F131	fwd_median_header_bytes
	F132	fwd_skewness_header_bytes
	F133	fwd_cov_header_bytes
	F134	fwd_mode_header_bytes
	F135	fwd_variance_header_bytes
	F136	bwd_total_header_bytes
	F137	bwd_max_header_bytes
	F138	bwd_min_header_bytes
	F139	bwd_mean_header_bytes
	F140	bwd_std_header_bytes
	F141	bwd_median_header_bytes
	F142	bwd_skewness_header_bytes
	F143	bwd_cov_header_bytes
	F144	bwd_mode_header_bytes
	F145	bwd_variance_header_bytes
	F146	fwd_avg_segment_size
	F147	bwd_avg_segment_size
	F148	avg_segment_size

Table 4.4: NTLFlowLyzer extracted features (continued)

Behavior	Feature	Feature Name
Header based	F149	min_header_bytes_delta_len
	F150	max_header_bytes_delta_len
	F151	mean_header_bytes_delta_len
	F152	mode_header_bytes_delta_len
	F153	variance_header_bytes_delta_len
	F154	std_header_bytes_delta_len
	F155	median_header_bytes_delta_len
	F156	skewness_header_bytes_delta_len
	F157	cov_header_bytes_delta_len
	F158	min_bwd_header_bytes_delta_len
	F159	max_bwd_header_bytes_delta_len
	F160	mean_bwd_header_bytes_delta_len
	F161	mode_bwd_header_bytes_delta_len
	F162	variance_bwd_header_bytes_delta_len
	F163	std_bwd_header_bytes_delta_len
	F164	median_bwd_header_bytes_delta_len
	F165	skewness_bwd_header_bytes_delta_len
	F166	cov_bwd_header_bytes_delta_len
	F167	min_fwd_header_bytes_delta_len
	F168	max_fwd_header_bytes_delta_len
	F169	mean_fwd_header_bytes_delta_len
	F170	mode_fwd_header_bytes_delta_len
	F171	variance_fwd_header_bytes_delta_len
	F172	std_fwd_header_bytes_delta_len
	F173	median_fwd_header_bytes_delta_len
	F174	skewness_fwd_header_bytes_delta_len
	F175	cov_fwd_header_bytes_delta_len
Payload based	F176	total_payload_bytes
	F177	fwd_total_payload_bytes
	F178	bwd_total_payload_bytes
	F179	payload_bytes_max
	F180	payload_bytes_min
	F181	payload_bytes_mean
	F182	payload_bytes_std
	F183	payload_bytes_variance
	F184	payload_bytes_median
	F185	payload_bytes_skewness

Table 4.4: NTLFlowLyzer extracted features (continued)

Behavior	Feature	Feature Name
Payload based	F186	payload_bytes_cov
	F187	payload_bytes_mode
	F188	fwd_payload_bytes_max
	F189	fwd_payload_bytes_min
	F190	fwd_payload_bytes_mean
	F191	fwd_payload_bytes_std
	F192	fwd_payload_bytes_variance
	F193	fwd_payload_bytes_median
	F194	fwd_payload_bytes_skewness
	F195	fwd_payload_bytes_cov
	F196	fwd_payload_bytes_mode
	F197	bwd_payload_bytes_max
	F198	bwd_payload_bytes_min
	F199	bwd_payload_bytes_mean
	F200	bwd_payload_bytes_std
	F201	bwd_payload_bytes_variance
	F202	bwd_payload_bytes_median
	F203	bwd_payload_bytes_skewness
	F204	bwd_payload_bytes_cov
	F205	bwd_payload_bytes_mode
	F206	min_payload_bytes_delta_len
	F207	max_payload_bytes_delta_len
	F208	mean_payload_bytes_delta_len
	F209	mode_payload_bytes_delta_len
	F210	variance_payload_bytes_delta_len
	F211	std_payload_bytes_delta_len
	F212	median_payload_bytes_delta_len
	F213	skewness_payload_bytes_delta_len
	F214	cov_payload_bytes_delta_len
	F215	min_bwd_payload_bytes_delta_len
	F216	max_bwd_payload_bytes_delta_len
	F217	mean_bwd_payload_bytes_delta_len
	F218	mode_bwd_payload_bytes_delta_len
	F219	variance_bwd_payload_bytes_delta_len
	F220	std_bwd_payload_bytes_delta_len
	F221	median_bwd_payload_bytes_delta_len
	F222	skewness_bwd_payload_bytes_delta_len

Table 4.4: NTLFlowLyzer extracted features (continued)

Behavior	Feature	Feature Name
Payload based	F223	cov_bwd_payload_bytes_delta_len
	F224	min_fwd_payload_bytes_delta_len
	F225	max_fwd_payload_bytes_delta_len
	F226	mean_fwd_payload_bytes_delta_len
	F227	mode_fwd_payload_bytes_delta_len
	F228	variance_fwd_payload_bytes_delta_len
	F229	std_fwd_payload_bytes_delta_len
	F230	median_fwd_payload_bytes_delta_len
	F231	skewness_fwd_payload_bytes_delta_len
	F232	cov_fwd_payload_bytes_delta_len
Flag based	F233	fin_flag_counts
	F234	psh_flag_counts
	F235	urg_flag_counts
	F236	ece_flag_counts
	F237	syn_flag_counts
	F238	ack_flag_counts
	F239	cwr_flag_counts
	F240	rst_flag_counts
	F241	fwd_fin_flag_counts
	F242	fwd_psh_flag_counts
	F243	fwd_urg_flag_counts
	F244	fwd_ece_flag_counts
	F245	fwd_syn_flag_counts
	F246	fwd_ack_flag_counts
	F247	fwd_cwr_flag_counts
	F248	fwd_rst_flag_counts
	F249	bwd_fin_flag_counts
	F250	bwd_psh_flag_counts
	F251	bwd_urg_flag_counts
	F252	bwd_ece_flag_counts
	F253	bwd_syn_flag_counts
	F254	bwd_ack_flag_counts
	F255	bwd_cwr_flag_counts
	F256	bwd_rst_flag_counts
	F257	fin_flag_percentage_in_total
	F258	psh_flag_percentage_in_total
	F259	urg_flag_percentage_in_total

Table 4.4: NTLFlowLyzer extracted features (continued)

Behavior	Feature	Feature Name
Flag based	F260	ece_flag_percentage_in_total
	F261	syn_flag_percentage_in_total
	F262	ack_flag_percentage_in_total
	F263	cwr_flag_percentage_in_total
	F264	rst_flag_percentage_in_total
	F265	fwd_fin_flag_percentage_in_total
	F266	fwd_psh_flag_percentage_in_total
	F267	fwd_urg_flag_percentage_in_total
	F268	fwd_ece_flag_percentage_in_total
	F269	fwd_syn_flag_percentage_in_total
	F270	fwd_ack_flag_percentage_in_total
	F271	fwd_cwr_flag_percentage_in_total
	F272	fwd_rst_flag_percentage_in_total
	F273	bwd_fin_flag_percentage_in_total
	F274	bwd_psh_flag_percentage_in_total
	F275	bwd_urg_flag_percentage_in_total
	F276	bwd_ece_flag_percentage_in_total
	F277	bwd_syn_flag_percentage_in_total
	F278	bwd_ack_flag_percentage_in_total
	F279	bwd_cwr_flag_percentage_in_total
	F280	bwd_rst_flag_percentage_in_total
	F281	fwd_fin_flag_percentage_in_fwd_packets
	F282	fwd_psh_flag_percentage_in_fwd_packets
	F283	fwd_urg_flag_percentage_in_fwd_packets
	F284	fwd_ece_flag_percentage_in_fwd_packets
	F285	fwd_syn_flag_percentage_in_fwd_packets
	F286	fwd_ack_flag_percentage_in_fwd_packets
	F287	fwd_cwr_flag_percentage_in_fwd_packets
	F288	fwd_rst_flag_percentage_in_fwd_packets
	F289	bwd_fin_flag_percentage_in_bwd_packets
	F290	bwd_psh_flag_percentage_in_bwd_packets
	F291	bwd_urg_flag_percentage_in_bwd_packets
	F292	bwd_ece_flag_percentage_in_bwd_packets
	F293	bwd_syn_flag_percentage_in_bwd_packets
	F294	bwd_ack_flag_percentage_in_bwd_packets
	F295	bwd_cwr_flag_percentage_in_bwd_packets
	F296	bwd_rst_flag_percentage_in_bwd_packets

Table 4.4: NTLFlowLyzer extracted features (continued)

Behavior	Feature	Feature Name
Side based	F297	fwd_init_win_bytes
	F298	bwd_init_win_bytes
	F299	avg_fwd_bytes_per_bulk
	F300	avg_fwd_bulk_rate
	F301	avg_bwd_bytes_per_bulk
	F302	avg_bwd_bulk_rate
	F303	fwd_bulk_state_count
	F304	fwd_bulk_total_size
	F305	bwd_bulk_state_count
	F306	bwd_bulk_total_size
	F307	subflow_fwd_bytes
	F308	subflow_bwd_bytes
	F309	down_up_rate
	F310	bytes_rate
	F311	fwd_bytes_rate
	F312	bwd_bytes_rate
	F313	delta_start
	F314	subflow_fwd_packets
	F315	subflow_bwd_packets

Index	Feature	Importance
1	mode_payload_bytes_delta_len	0.35611185
2	mode_fwd_packets_delta_len	0.2490578
3	mean_bwd_payload_bytes_delta_len	0.14168905
4	cov_bwd_header_bytes_delta_len	0.06315309
5	mode_packets_delta_len	0.041265965
6	cov_bwd_packets_delta_len	0.0316184
7	active_skewness	0.02379151
8	median_bwd_packets_delta_len	0.019767964
9	bwd_payload_bytes_skewness	0.01561531
10	mean_bwd_packets_delta_len	0.014975527
11	mode_bwd_payload_bytes_delta_len	0.012427035
12	fwd_total_header_bytes	0.012176397
13	median_fwd_payload_bytes_delta_len	0.006382107
14	skewness_bwd_header_bytes_delta_len	0.004504317
15	active_min	0.003328606
16	mode_fwd_payload_bytes_delta_len	0.00313341
17	fin_flag_counts	0.000952884
18	mean_fwd_packets_delta_len	4.36E-05
19	payload_bytes_std	1.63E-06
20	bwd_total_payload_bytes	1.17E-06
21	max_header_bytes_delta_len	2.18E-07
22	median_fwd_header_bytes_delta_len	2.05E-07
23	fwd_init_win_bytes	2.02E-07
24	min_fwd_payload_bytes_delta_len	1.76E-07
25	bwd_payload_bytes_median	1.63E-07
26	payload_bytes_mean	1.22E-07
27	min_fwd_packets_delta_len	1.00E-07
28	bwd_init_win_bytes	8.91E-08
29	bwd_avg_segment_size	8.21E-08
30	bwd_packets_IAT_mean	6.56E-08
31	variance_fwd_payload_bytes_delta_len	6.15E-08
32	total_payload_bytes	5.57E-08
33	fwd_ack_flag_counts	4.82E-08
34	fwd_max_header_bytes	4.53E-08
35	bwd_packets_count	4.29E-08
36	median_fwd_packets_delta_len	4.04E-08
37	bwd_payload_bytes_variance	3.65E-08
38	std_header_bytes	3.57E-08
39	cov_header_bytes	3.43E-08
40	bwd_min_header_bytes	3.34E-08
41	psh_flag_counts	3.25E-08
42	bwd_packets_IAT_mode	3.09E-08
43	packets_rate	2.97E-08
44	min_bwd_packets_delta_time	2.97E-08
45	fwd_std_header_bytes	2.86E-08
46	fwd_packets_IAT_cov	2.49E-08
47	bwd_packets_IAT_min	2.47E-08
48	min_packets_delta_len	2.34E-08
49	mean_fwd_payload_bytes_delta_len	2.01E-08
50	fwd_min_header_bytes	2.01E-08
51	bwd_psh_flag_counts	1.90E-08
52	fwd_payload_bytes_variance	1.74E-08
53	fwd_mode_header_bytes	1.63E-08
54	mean_fwd_header_bytes_delta_len	1.58E-08
55	mean_header_bytes	1.44E-08
56	bytes_rate	1.20E-08
57	min_bwd_packets_delta_len	1.14E-08
58	fwd_packets_IAT_mean	9.39E-09
59	skewness_bwd_packets_delta_time	8.95E-09
60	packets_IAT_mode	8.94E-09
61	bwd_bytes_rate	8.59E-09
62	fwd_median_header_bytes	7.05E-09
63	max_fwd_packets_delta_time	6.50E-09
64	fwd_bytes_rate	6.17E-09
65	syn_flag_percentage_in_total	5.38E-09
66	total_header_bytes	5.34E-09
67	packet_IAT_min	5.31E-09
68	packet_IAT_max	4.79E-09
69	max_bwd_header_bytes_delta_len	4.68E-09
70	variance_packets_delta_len	3.48E-09
71	packets_IAT_mean	2.60E-09
72	mode_bwd_packets_delta_len	2.38E-09
73	bwd_mean_header_bytes	2.03E-09
74	skewness_bwd_payload_bytes_delta_len	4.52E-10
75	ack_flag_counts	3.48E-10
76	max_header_bytes	8.70E-11
77	mean_header_bytes_delta_len	3.74E-11

Table 4.5: Feature Importance

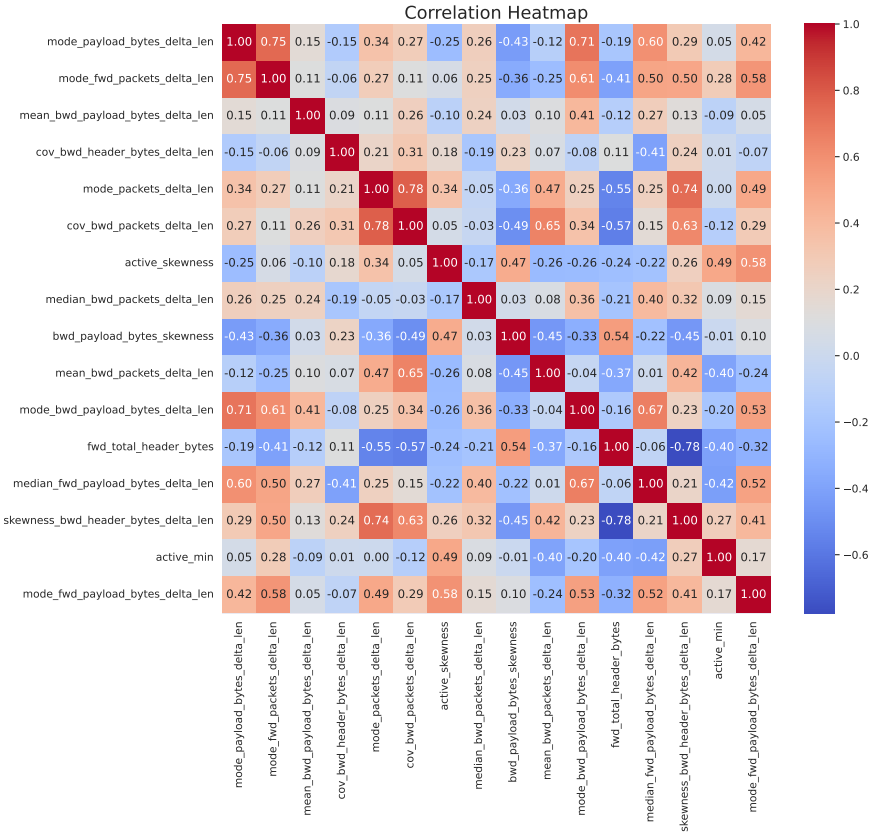


Figure 25: Correlation Coefficients between features

5 Experiments and results

A comprehensive testbed is required to train, evaluate, and compare the proposed model's performance and profile the malicious behavior of both samples. We utilized the generated augmented dataset to evaluate and profile the bot samples' malicious behavior. This section delves into the proposed model's performance analysis compared to other models and explains the utilized profiling approach.

First, we detail the selection process of the most suitable features in subsection 5.1. Next, we describe the training and evaluation process of our proposed model, including model optimization and hyperparameter tuning in subsection 5.2. Additionally, we explain the proposed profiling approach in subsection 5.2. Finally, we compare the performance of the proposed model with other models in subsection 5.4.

As Table 4.3 reveals, a significantly higher proportion of samples belong to bot types "Mirai," "Gagfyt," "Linux Mirai," "Kenjiro," "IRCBot," and "Okiru" compared to others like "Muhstik," "Hakai," and "Torii". Therefore, bot samples with limited data points ("Muhstik," "Hakai," and "Torii" with 84, 814, and 14 rows, respectively) were excluded from the evaluation and comparison of the proposed model. Additionally, since our research focuses on profiling the malicious behavior of bot samples, we used only bot samples for training and evaluation, excluding benign data.

5.1 Feature selection

Our feature selection technique has two steps. First, we used XGBoost to prioritize features based on their gain, resulting in 77 features with non-zero importance scores. As Table 4.5 shows, there exists a significant drop between the importance of 16th and 17th features; this led us to focus on the top 16. Next, we addressed collinearity by calculating correlation coefficients between the remaining features. For pairs with a correlation coefficient above 0.9, we removed the feature with the lower XGBoost importance score. This ensured our final feature set contained the most important and independent features, minimizing redundancy. As shown in Fig. 25, all final feature correlation coefficients are below 0.9, confirming their relative independence and suitability for building an effective detection and identification model.

5.2 Model Optimization

To optimize the proposed model, we utilized an optimization approach focusing on hyperparameter tuning. The optimization process involved adjusting hyperparameters such as learning rate, dropout rate, optimizer function, input activation function, number of epochs, units per layer, and batch size. We used softmax for the classification output activation and a single hidden layer for simplicity. For classification, we employed the "categorical cross-entropy" loss function, which compares predicted and actual probability distributions, ideal for multi-class tasks [141]. Hyperparameter tuning was conducted using Amazon SageMaker.
Amazon SageMaker simplifies hyperparameter tuning for machine learning and deep learning models on AWS by automating the exploration of different hyperparameter combinations [142]. Users define the search space, choose a search strategy (random, grid, or Bayesian), and launch tuning jobs to identify the best configurations based on metrics like accuracy [142]. This approach saves time and resources, leading to more optimized systems. Well-defined hyperparameter ranges and early stopping techniques can further improve tuning efficiency [60]. The hyperparameter strategies include:

- **Random Search**: Samples hyperparameter combinations randomly. While simple, it may miss promising areas of the search space, especially for complex systems [60].
- **Grid Search**: Exhaustively searches all possible combinations in a predefined grid. It can be inefficient for systems with many hyperparameters [143].
- **Bayesian Optimization**: Uses past evaluations to prioritize promising configurations, building a statistical model to select the next hyperparameter set, leading to faster convergence [144].

We chose Bayesian optimization due to its balance between efficiency and intelligent exploration for our large numerical dataset [145]. We set SageMaker Experiments' `max_jobs` parameter to 20 to control concurrent training jobs [142]. Detailed hyperparameter ranges are provided in Table 5.1.
Following tuning, the selected values for each job are presented in Table 5.1. Tables 5.2, 5.3, and 5.4 show system performance, with most jobs achieving 0.9999 accuracy, except the 6th and 12th, which reached 0.9301 and 0.7790, respectively.
The 10th training job was chosen as the best overall system due to its efficient parameters, including fewer units, a lower learning rate, fewer epochs, and a smaller batch size, making it suitable for edge devices

with limited resources [146]. It uses the Adam optimizer for faster convergence and adaptive learning rates [147] and the ReLU activation function for faster training and mitigating the vanishing gradient problem, making it more effective in deep networks [148]. This optimized model ensures high performance without compromising resource efficiency, making it potentially ideal for deployment on IoT edge devices.

Parameters	Range Type	Range	Selected value
Optimizer	Categorical	['RMSprop', 'Adam', 'Adagrad', 'Adadelta', 'Adamax', 'Nadam']	Adam
Activation	Categorical	['tanh', 'relu', 'sigmoid']	relu
Learning rate	Continuous	(0.0001, 0.01)	0.0001
Dropout	Continuous	(0.0, 0.5)	0.4460
Units	Integer	(50, 200)	58
Batch size	Integer	(64, 512)	192
Epochs	Integer	(5, 50)	9

Table 5.1: LSTM parameters and selected values utilizing Amazon SageMaker experiments for multiclass classification

Index	Overall Accuracy	Overall Precision	Overall Recall	Overall F1 Score	Units	Learning Rate	Dropout	Epochs	Batch Size	Optimizer	Activation
1	0.9999	0.9999	0.9999	0.9999	101	0.0004	0.4194	33	367	Nadam	tanh
2	0.9999	0.9999	0.9999	0.9999	192	0.0018	0.4157	47	364	RMSprop	relu
3	0.9999	0.9999	0.9999	0.9999	164	0.0027	0.0632	42	277	Adamax	relu
4	0.9999	0.9999	0.9999	0.9999	200	0.0045	0.0063	6	331	Adam	tanh
5	0.9999	0.9999	0.9999	0.9999	62	0.0002	0.1526	34	330	Adam	sigmoid
6	0.9301	0.9507	0.9300	0.9268	139	0.0001	0.3033	22	281	Adagrad	sigmoid
7	0.9999	0.9999	0.9999	0.9999	190	0.0004	0.3741	44	382	Adadelta	tanh
8	0.9999	0.9999	0.9999	0.9999	60	0.0015	0.0575	14	388	Adadelta	sigmoid
9	0.9999	0.9999	0.9999	0.9999	81	0.0002	0.3598	48	394	Adagrad	relu
10	0.9999	0.9999	0.9999	0.9999	58	0.0001	0.4460	9	192	Adam	relu
11	0.9999	0.9999	0.9999	0.9999	196	0.0092	0.4453	6	349	RMSprop	relu
12	0.7790	0.8987	0.7791	0.7647	60	0.0001	0.3911	5	211	Adadelta	relu
13	0.9999	0.9999	0.9999	0.9999	100	0.0002	0.3438	50	274	Adam	relu
14	0.9999	0.9999	0.9999	0.9999	66	0.0002	0.2102	13	194	Adagrad	tanh
15	0.9999	0.9999	0.9999	0.9999	164	0.0037	0.1795	40	448	Nadam	sigmoid
16	0.9999	0.9999	0.9999	0.9999	83	0.0098	0.0919	47	387	Adam	relu
17	0.9999	0.9999	0.9999	0.9999	195	0.0055	0.4331	49	511	Nadam	sigmoid
18	0.9999	0.9999	0.9999	0.9999	164	0.0065	0.4814	47	474	Adam	sigmoid
19	0.9999	0.9999	0.9999	0.9999	190	0.0082	0.3548	12	459	Adam	sigmoid
20	0.9999	0.9999	0.9999	0.9999	175	0.0006	0.1683	50	322	Adadelta	relu

Table 5.2: Model performance after applying hyperparameter tuning using amazon sagemaker experiments

Index	Label	Precision	Recall	F1 Score	Units	Learning Rate	Dropout	Epochs	Batch Size	Optimizer	Activation
1	Gagfyt	0.9999	1	0.9999	101	0.0004	0.4194	33	367	Nadam	tanh
	Mirai	1	0.9999	0.9999	101	0.0004	0.4194	33	367	Nadam	tanh
	Linux,Mirai	0.9999	1	0.9999	101	0.0004	0.4194	33	367	Nadam	tanh
	Kenjiro	0.9999	0.9999	1	101	0.0004	0.4194	33	367	Nadam	tanh
	Okiru	0.9999	0.9999	0.9999	101	0.0004	0.4194	33	367	Nadam	tanh
	IRCBot	1	1	0.9999	101	0.0004	0.4194	33	367	Nadam	tanh
2	Gagfyt	1	1	1	192	0.0018	0.4157	47	364	RMSprop	relu
	Mirai	0.9999	0.9999	0.9999	192	0.0018	0.4157	47	364	RMSprop	relu
	Linux,Mirai	0.9999	1	0.9999	192	0.0018	0.4157	47	364	RMSprop	relu
	Kenjiro	1	1	1	192	0.0018	0.4157	47	364	RMSprop	relu
	Okiru	0.9999	0.9999	0.9999	192	0.0018	0.4157	47	364	RMSprop	relu
	IRCBot	1	1	1	192	0.0018	0.4157	47	364	RMSprop	relu
3	Gagfyt	1	1	1	164	0.0027	0.0631	42	277	Adamax	relu
	Mirai	0.9999	0.9999	0.9999	164	0.0027	0.0631	42	277	Adamax	relu
	Linux,Mirai	1	1	1	164	0.0027	0.0631	42	277	Adamax	relu
	Kenjiro	1	1	1	164	0.0027	0.0631	42	277	Adamax	relu
	Okiru	0.9999	0.9999	0.9999	164	0.0027	0.0631	42	277	Adamax	relu
	IRCBot	1	1	1	164	0.0027	0.0631	42	277	Adamax	relu
4	Gagfyt	1	1	1	200	0.0045	0.0063	6	331	Adam	tanh
	Mirai	0.9999	0.9999	0.9999	200	0.0045	0.0063	6	331	Adam	tanh
	Linux,Mirai	1	1	1	200	0.0045	0.0063	6	331	Adam	tanh
	Kenjiro	1	1	1	200	0.0045	0.0063	6	331	Adam	tanh
	Okiru	0.9999	0.9999	0.9999	200	0.0045	0.0063	6	331	Adam	tanh
	IRCBot	1	1	1	200	0.0045	0.0063	6	331	Adam	tanh
5	Gagfyt	0.9999	0.9999	0.9997	62	0.0002	0.1526	34	330	Adam	sigmoid
	Mirai	0.9999	0.9999	0.9999	62	0.0002	0.1526	34	330	Adam	sigmoid
	Linux,Mirai	0.9999	0.9999	0.9999	62	0.0002	0.1526	34	330	Adam	sigmoid
	Kenjiro	1	1	1	62	0.0002	0.1526	34	330	Adam	sigmoid
	Okiru	0.9999	0.9999	0.9999	62	0.0002	0.1526	34	330	Adam	sigmoid
	IRCBot	1	1	1	62	0.0002	0.1526	34	330	Adam	sigmoid
6	Gagfyt	0.9998	0.9999	0.9999	139	0.0001	0.3033	22	281	Adagrad	sigmoid
	Mirai	0.7043	0.9999	0.8265	139	0.0001	0.3033	22	281	Adagrad	sigmoid
	Linux,Mirai	0.9999	1	0.9999	139	0.0001	0.3033	22	281	Adagrad	sigmoid
	Kenjiro	1	1	1	139	0.0001	0.3033	22	281	Adagrad	sigmoid
	Okiru	0.9999	0.5802	0.7344	139	0.0001	0.3033	22	281	Adagrad	sigmoid
	IRCBot	1	1	1	139	0.0001	0.3033	22	281	Adagrad	sigmoid
7	Gagfyt	1	1	1	190	0.0004	0.3741	44	382	Adadelta	tanh
	Mirai	1	0.9999	0.9999	190	0.0004	0.3741	44	382	Adadelta	tanh
	Linux,Mirai	0.9999	1	0.9999	190	0.0004	0.3741	44	382	Adadelta	tanh
	Kenjiro	1	1	1	190	0.0004	0.3741	44	382	Adadelta	tanh
	Okiru	0.9999	0.9999	0.9999	190	0.0004	0.3741	44	382	Adadelta	tanh
	IRCBot	1	1	1	190	0.0004	0.3741	44	382	Adadelta	tanh
8	Gagfyt	0.9998	0.9999	0.9999	60	0.0015	0.0575	14	388	Adadelta	sigmoid
	Mirai	0.9999	0.9999	0.9999	60	0.0015	0.0575	14	388	Adadelta	sigmoid
	Linux,Mirai	0.9999	1	0.9999	60	0.0015	0.0575	14	388	Adadelta	sigmoid
	Kenjiro	0.9998	0.9999	0.9999	60	0.0015	0.0575	14	388	Adadelta	sigmoid
	Okiru	0.9999	0.9999	0.9999	60	0.0015	0.0575	14	388	Adadelta	sigmoid
	IRCBot	1	1	1	60	0.0015	0.0575	14	388	Adadelta	sigmoid
9	Gagfyt	1	1	1	81	0.0002	0.3598	48	394	Adagrad	relu
	Mirai	1	0.9999	0.9999	81	0.0002	0.3598	48	394	Adagrad	relu
	Linux,Mirai	0.9999	1	0.9999	81	0.0002	0.3598	48	394	Adagrad	relu
	Kenjiro	0.9999	1	1	81	0.0002	0.3598	48	394	Adagrad	relu
	Okiru	0.9999	0.9999	0.9999	81	0.0002	0.3598	48	394	Adagrad	relu
	IRCBot	0.9998	0.9999	0.9998	81	0.0002	0.3598	48	394	Adagrad	relu
10	Gagfyt	0.9999	0.9999	0.9998	58	0.0001	0.4460	9	192	Adam	relu
	Mirai	0.9999	0.9999	0.9999	58	0.0001	0.4460	9	192	Adam	relu
	Linux,Mirai	0.9999	0.9999	0.9999	58	0.0001	0.4460	9	192	Adam	relu
	Kenjiro	0.9997	0.9999	0.9999	58	0.0001	0.4460	9	192	Adam	relu
	Okiru	0.9997	0.9998	0.9999	58	0.0001	0.4460	9	192	Adam	relu
	IRCBot	0.9999	0.9998	0.9999	58	0.0001	0.4460	9	192	Adam	relu
11	Gagfyt	0.9999	0.9998	0.9999	196	0.0092	0.4453	6	349	RMSprop	relu
	Mirai	0.9999	0.9999	0.9999	196	0.0092	0.4453	6	349	RMSprop	relu
	Linux,Mirai	0.9999	0.9998	0.9999	196	0.0092	0.4453	6	349	RMSprop	relu
	Kenjiro	0.9999	0.9999	0.9998	196	0.0092	0.4453	6	349	RMSprop	relu
	Okiru	0.9999	0.9999	0.9999	196	0.0092	0.4453	6	349	RMSprop	relu
	IRCBot	0.9997	0.9999	0.9999	196	0.0092	0.4453	6	349	RMSprop	relu
12	Gagfyt	0.9529	0.9999	0.9759	60	0.0001	0.3911	5	211	Adadelta	relu
	Mirai	0.4391	0.9999	0.6103	60	0.0001	0.3911	5	211	Adadelta	relu
	Linux,Mirai	0.9998	0.2737	0.4297	60	0.0001	0.3911	5	211	Adadelta	relu
	Kenjiro	0.9998	0.9996	0.9997	60	0.0001	0.3911	5	211	Adadelta	relu
	Okiru	0.9999	0.4011	0.5725	60	0.0001	0.3911	5	211	Adadelta	relu
	IRCBot	0.9998	0.9997	0.9998	60	0.0001	0.3911	5	211	Adadelta	relu

Table 5.3: Model performance on each bot sample after applying hyperparameter tuning using amazon sagemaker experiments

Index	Label	Precision	Recall	F1 Score	Units	Learning Rate	Dropout	Epochs	Batch Size	Optimizer	Activation
13	Gagfyt	1	1	1	100	0.0002	0.3438	50	274	Adam	relu
	Mirai	0.9999	0.9999	0.9999	100	0.0002	0.3438	50	274	Adam	relu
	Linux,Mirai	0.9999	1	0.9999	100	0.0002	0.3438	50	274	Adam	relu
	Kenjiro	1	1	1	100	0.0002	0.3438	50	274	Adam	relu
	Okiru	0.9999	0.9999	0.9999	100	0.0002	0.3438	50	274	Adam	relu
	IRCBot	1	1	1	100	0.0002	0.3438	50	274	Adam	relu
14	Gagfyt	1	1	1	66	0.0002	0.2102	13	194	Adagrad	tanh
	Mirai	1	0.9999	0.9999	66	0.0002	0.2102	13	194	Adagrad	tanh
	Linux,Mirai	0.9999	1	0.9999	66	0.0002	0.2102	13	194	Adagrad	tanh
	Kenjiro	1	1	1	66	0.0002	0.2102	13	194	Adagrad	tanh
	Okiru	0.9999	0.9999	0.9999	66	0.0002	0.2102	13	194	Adagrad	tanh
	IRCBot	1	1	1	66	0.0002	0.2102	13	194	Adagrad	tanh
15	Gagfyt	1	1	1	164	0.0037	0.1795	40	448	Nadam	sigmoid
	Mirai	0.9999	0.9999	0.9999	164	0.0037	0.1795	40	448	Nadam	sigmoid
	Linux,Mirai	1	1	1	164	0.0037	0.1795	40	448	Nadam	sigmoid
	Kenjiro	1	1	1	164	0.0037	0.1795	40	448	Nadam	sigmoid
	Okiru	0.9999	0.9999	0.9999	164	0.0037	0.1795	40	448	Nadam	sigmoid
	IRCBot	1	1	1	164	0.0037	0.1795	40	448	Nadam	sigmoid
16	Gagfyt	1	1	1	83	0.0098	0.0919	47	387	Adam	relu
	Mirai	0.9999	0.9999	0.9999	83	0.0098	0.0919	47	387	Adam	relu
	Linux,Mirai	0.9999	1	0.9999	83	0.0098	0.0919	47	387	Adam	relu
	Kenjiro	1	1	1	83	0.0098	0.0919	47	387	Adam	relu
	Okiru	0.9999	0.9999	0.9999	83	0.0098	0.0919	47	387	Adam	relu
	IRCBot	1	1	1	83	0.0098	0.0919	47	387	Adam	relu
17	Gagfyt	1	1	1	195	0.0055	0.4331	49	511	Nadam	sigmoid
	Mirai	0.9999	0.9999	0.9999	195	0.0055	0.4331	49	511	Nadam	sigmoid
	Linux,Mirai	0.9999	1	0.9999	195	0.0055	0.4331	49	511	Nadam	sigmoid
	Kenjiro	1	1	1	195	0.0055	0.4331	49	511	Nadam	sigmoid
	Okiru	0.9999	0.9999	0.9999	195	0.0055	0.4331	49	511	Nadam	sigmoid
	IRCBot	1	1	1	195	0.0055	0.4331	49	511	Nadam	sigmoid
18	Gagfyt	1	1	1	164	0.0065	0.4814	47	474	Adam	sigmoid
	Mirai	0.9999	0.9999	0.9999	164	0.0065	0.4814	47	474	Adam	sigmoid
	Linux,Mirai	1	1	1	164	0.0065	0.4814	47	474	Adam	sigmoid
	Kenjiro	1	1	1	164	0.0065	0.4814	47	474	Adam	sigmoid
	Okiru	0.9999	0.9999	0.9999	164	0.0065	0.4814	47	474	Adam	sigmoid
	IRCBot	1	1	1	164	0.0065	0.4814	47	474	Adam	sigmoid
19	Gagfyt	1	1	1	190	0.0082	0.3548	12	459	Adam	sigmoid
	Mirai	0.9999	0.9999	0.9999	190	0.0082	0.3548	12	459	Adam	sigmoid
	Linux,Mirai	1	1	1	190	0.0082	0.3548	12	459	Adam	sigmoid
	Kenjiro	1	1	1	190	0.0082	0.3548	12	459	Adam	sigmoid
	Okiru	0.9999	0.9999	0.9999	190	0.0082	0.3548	12	459	Adam	sigmoid
	IRCBot	1	1	1	190	0.0082	0.3548	12	459	Adam	sigmoid
20	Gagfyt	1	1	1	175	0.0006	0.1683	50	322	Adadelta	relu
	Mirai	1	0.9999	0.9999	175	0.0006	0.1683	50	322	Adadelta	relu
	Linux,Mirai	0.9999	1	0.9999	175	0.0006	0.1683	50	322	Adadelta	relu
	Kenjiro	1	1	1	175	0.0006	0.1683	50	322	Adadelta	relu
	Okiru	0.9999	0.9999	0.9999	175	0.0006	0.1683	50	322	Adadelta	relu
	IRCBot	1	1	1	175	0.0006	0.1683	50	322	Adadelta	relu

Table 5.4: Model performance on each bot sample after applying hyperparameter tuning using Amazon SageMaker experiments

5.3 Profiling

The final phase involves profiling the malicious behavior of bot samples based on attribute, relation, and entity extraction. To achieve this, we first explain the behavior of each malicious bot samples in subsection 5.3.1. Next, we analyze the selected features in the feature selection process in subsection 5.3.2. Finally, we profile the malicious behavior of bot samples both individually and collectively in subsection 5.3.3.

5.3.1 Bots behavior Analysis

A botnet is a set of networked devices that have been compromised by a malicious actor (a botmaster) to conduct various activities, such as sending spam or initiating Denial of Service (DoS) attacks. Such compromises often occur due to vulnerabilities in the software running on the devices, leaving said devices amenable to outside control. A device that is assumed into a botnet is known as a bot or a zombie [149].
A bot is an automated software application that performs repetitive tasks over a network, imitating human behavior but faster and more accurately, often operating without human intervention [150]. Coordinated by a Command and Control(C&C) structure [149], individual bots are not highly threatening alone, but a botmaster controlling many bots poses a significant threat, commonly using botnets for DDoS attacks, spamming, phishing, malware distribution, click fraud, and crypto-jacking [149]. Bots can interact with websites, chat with visitors, or scan content. While many bots are beneficial, some are designed with malicious intent. Organizations secure their systems from harmful bots and use helpful ones for operational efficiency [151, 150]. Some define bots as semi-automatic agents controlled by a botmaster or human [150]. This section explains the behavior and purpose of bot samples in the BCCC-Aposema-BoT IoT-24 dataset utilized in the training and evaluation of the proposed model 4.1.

- **Gagfyt**: Also known as Bashlite, this botnet targets IoT devices to launch DoS attacks like TCP, junk, and User Datagram Protocol (UDP) flooding, causing target servers to crash. It spreads by exploiting system vulnerabilities, has a modular architecture, and evades detection through code obfuscation and mimicking legitimate traffic [152, 149, 153].

- **Mirai**: This botnet exploits IoT device vulnerabilities, using default credentials and open ports. It infects devices and launches packet-based attacks, causing server crashes. Mirai's decentralized structure allows bots to communicate directly with each other, making it harder to dismantle [154, 149, 155].

- **Linux Mirai**: Similar to Mirai, this botnet targets Linux-based IoT devices using default credentials and open ports. It infects devices via common IoT protocols and launches packet-based attacks. Newer variants adopt decentralized communication to evade detection [156, 154].

- **IRCBot**: This bot performs tasks like moderating chats or managing file sharing in IRC channels. In contrast, IRCBotnets involve compromised computers controlled by an attacker to launch attacks or steal data, communicating via IRC channels disguised as regular chat [157, 158, 159].

- **Kenjiro**: It is a variant of the Hakai botnet. Kenjiro targets IoT devices with weak Telnet and SSH credentials. It has a modular structure for easy adaptability, launches DDoS attacks, and uses techniques like code obfuscation to evade detection. Infections can lead to slow device performance and data exposure [154, 160].

- **Okiru**: A variant of Mirai, Okiru targets IoT devices with Argonaut RISC Core (ARC) processors. It uses TCP and UDP for communication, spreading through open ports and weak credentials. It hides

its code to evade antivirus detection and can infect multiple CPU architectures, making it versatile for large-scale DDoS attacks [161, 162, 154].

5.3.2 Selected Features Analysis

This section carefully analyzes feature selection and selected and non-selected feature categories in detail. Based on Tables 4.4 and, 4.5, the selected features can be categorized into Payload-based, Packet-based, Header-based, and status-based. Table 5.5 shows the category and importance of the selected features. The provided index in Table 5.5 is based on the index of Table 4.5. Below is the category of selected and non-selected features in detail.

Category	Index	Feature Name	Feature Importance
Payload based	1	mode_payload_bytes_delta_len	0.3561
	3	mean_bwd_payload_bytes_delta_len	0.1416
	9	bwd_payload_bytes_skewness	0.0156
	11	mode_bwd_payload_bytes_delta_len	0.0124
	13	median_fwd_payload_bytes_delta_len	0.0063
	16	mode_fwd_payload_bytes_delta_len	0.0031
Packet based	2	mode_fwd_packets_delta_len	0.2490
	5	mode_packets_delta_len	0.0412
	6	cov_bwd_packets_delta_len	0.0316
	8	median_bwd_packets_delta_len	0.0197
	10	mean_bwd_packets_delta_len	0.0149
Header based	4	cov_bwd_header_bytes_delta_len	0.0631
	12	fwd_total_header_bytes	0.0121
	14	skewness_bwd_header_bytes_delta_len	0.0045
Status based	7	active_skewness	0.0237
	15	active_min	0.0033

Table 5.5: Selected Features Category
Index column is based on the index in Table 4.5

- **Packet-Related Features**: Botnets are networks of compromised devices (bots) controlled by a malicious actor. These bots often communicate with each other or a central C&C server to carry out attacks like DDoS, spam campaigns, or data theft. This communication involves network traffic, which is made up of packets. Packet-based features are measurable characteristics extracted from these packets. Botnet traffic often has distinct patterns in how packets are structured and transmitted, which differ from normal, legitimate traffic. By analyzing these features, we can potentially identify signs of botnet activity. For instance, "Mode Fwd Packets Delta Len" looks at the most common difference in length between packets sent in the forward direction (from the bot to the C&C or between bots). Botnets might use specific packet sizes for their communication, and this feature could reveal unusual patterns. Each botnet family (Gagfyt, Mirai, etc.) might have unique communication patterns. For example, Mirai is Known for using specific packet lengths and timing intervals in its DDoS attacks. Analyzing features like the selected ones can help identify these patterns [154, 163].

- **Payload-Related features**: In the context of network traffic, a payload refers to the actual data being transmitted, excluding the headers or metadata. Payload bytes are simply the size of this data in bytes. Botnets, networks of compromised devices controlled by a malicious actor, often utilize specific patterns in their payload bytes to carry out their malicious activities [164]. For instance, the mode of payload length differences (the most frequent change in payload size between consecutive packets) can be an indicator of botnet behavior. Botnets often communicate with a C&C server, receiving commands and sending back reports or exfiltrated data. These communications can lead to varying payload sizes, potentially resulting in a high mode of payload length differences [164].

  Different botnet families tend to exhibit unique patterns in their payload-byte usage due to variations in their architecture, communication protocols, and intended tasks. By analyzing these features, security researchers and network administrators can develop detection models tailored to identify specific botnets. For example, Gafgyt/Mirai botnets, known for their involvement in DDoS attacks, may show a high mode of payload length differences due to the varying sizes of attack commands and the data sent during the attacks[165]. As Table 5.5 shows, 5 out of 16 top selected features belong to payload-based features. This shows the importance of payload-related features in botnet detection.

- **Header-Related Features**: Network traffic consists of packets, each with a header containing essential information (e.g., source/destination IP, protocol, length). Botnets, being networks of compromised devices, often exhibit distinct patterns in their packet headers compared to legitimate traffic. Benign TCP connections typically follow standardized procedures, especially during the initial handshake, leading to predictable patterns in TCP header values [115].

  Botnets, however, may deviate from these norms. For example, they might use spoofed source IP addresses, employ unusual protocols, or send malformed packets with incorrect header values. These anomalies, especially those appearing at the beginning of a connection, can be strong indicators of botnet activity. While attacks like Gagfyt overwhelm resources, their detection often relies on analyzing traffic volume and content rather than solely header anomalies [154].

- **Status based**: Status-based features are network traffic characteristics derived from the state and timing of connections. These features, such as packet inter-arrival times, connection duration, and packet counts, can reveal anomalies and distinguish normal traffic from potentially malicious botnet activity. Botnets often exhibit distinct communication patterns due to their coordinated actions and C&C structures. Status-based features can capture these patterns, making them valuable for botnet detection.

  For example, ”active skewness” measures the asymmetry of packet transmission times within a flow. Botnets engaged in synchronized activities, like DDoS attacks, may show higher skewness compared to normal traffic. Similarly, ”active min,” the minimum time interval between packets, can be consistently small in botnets due to rapid C&C communication. Different botnets may have unique status-based feature profiles due to variations in their architecture, purpose, and communication protocols. While botnets like Mirai and its variants are known for DDoS attacks and their associated traffic patterns, other botnets may exhibit different behaviors. For instance, spam botnets might show different patterns in the frequency and timing of connections [154, 163].

- **All together**: The presence of anomalies across diverse feature categories significantly improves attack detection accuracy. Recognizing patterns in these features strengthens the detection mechanism while examining correlations between them unveils more comprehensive attack signatures. For instance, a combination of high ”active skewness” and unusual ”mode payload bytes delta len” might signal a complex DDoS attack. Understanding these relationships offers deeper insights into evolving attack strategies, leading to improved detection. Additionally, attackers often modify their tactics during an attack. Continuous monitoring of features enables the identification of shifting attack patterns. A nuanced understanding of this dynamic nature is crucial for creating adaptive detection mechanisms that can respond to emerging threats in real time [115].

  Incorporating diverse and informative feature categories provides a comprehensive view of network traffic. This approach empowers the development of robust and adaptable detection models capable of identifying a wide range of DDoS attack tactics. By thoroughly examining correlations and evolving patterns, these models effectively stay ahead of attackers and respond dynamically to the complex landscape of cyber threats [115].

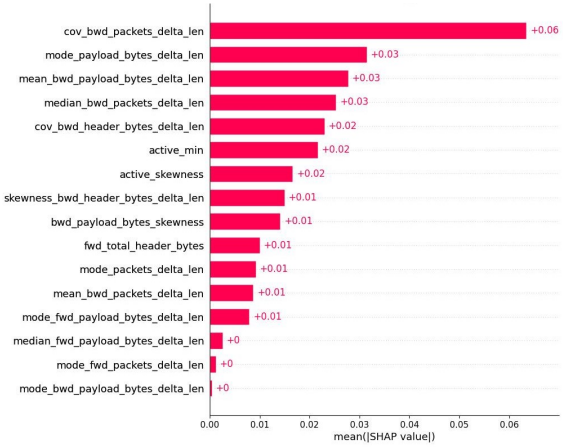
- **Not-Selected Features Analysis**: In network traffic analysis, the absence of certain features, such as flag-based, rate-based, and side-based features, can offer insights into the nature of the data being examined. Understanding why these features might be absent or considered less informative is crucial. The absence of flag-based features (e.g., TCP flags) could be because, while essential for connection establishment and termination, modern bots often mimic legitimate traffic patterns, making these features less distinctive for botnet identification. Similarly, excluding rate-based features (e.g., packet rates, byte rates) is due to sophisticated bots operating at lower rates to avoid detection. While these features are helpful in detecting volumetric attacks, they are less effective against stealthier botnets. Side-based features can be divided into two subcategories: Flow-Based and Session/Connection-Based. These features might not be highly informative due to:

  - **Mimicking Legitimate Traffic**: Bots blend in with normal traffic using standard parameters.
  - **Diversity of Bot Behaviors**: Varied botnet communication strategies make single side-based features less reliable.
  - **Network Variability**: Latency, and bandwidth changes affect transfer rates, complicating the establishment of a normal baseline.
  - **Feature Correlation**: Some side-based features correlate with other features, leading to the selection of more unique features.

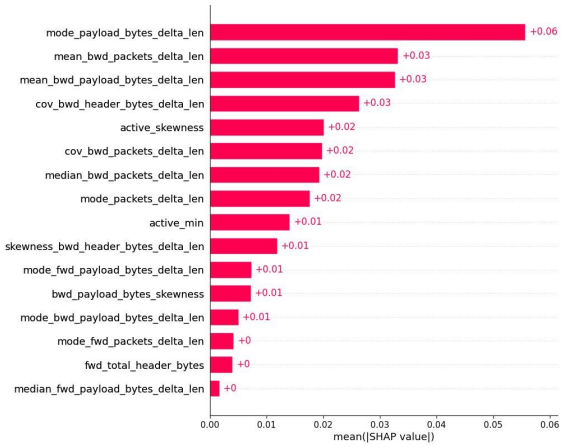
  In summary, traditional features like flags and rates are less effective due to modern bots' evasion tactics, and side-based features are limited by bot diversity, network variability, and feature correlation.

5.3.3 Bot Samples Behavior Profiling

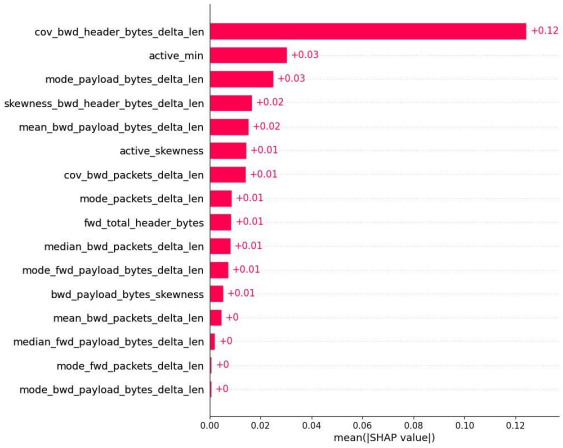
To profile the behavior of botnet samples both individually and collectively, we utilized SHAP, specifically Gradient SHAP. Gradient SHAP estimates each feature's contribution to the model's prediction by computing



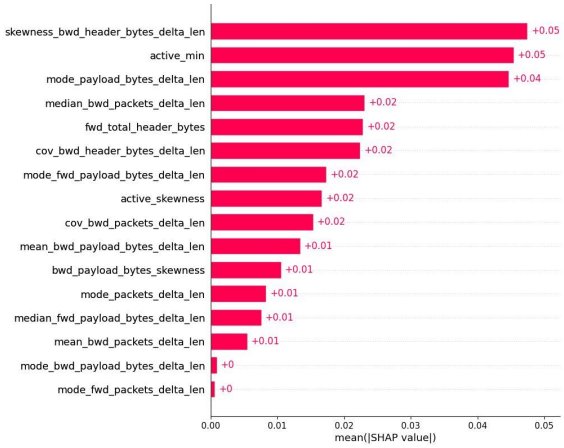
((a)) Gagfyt



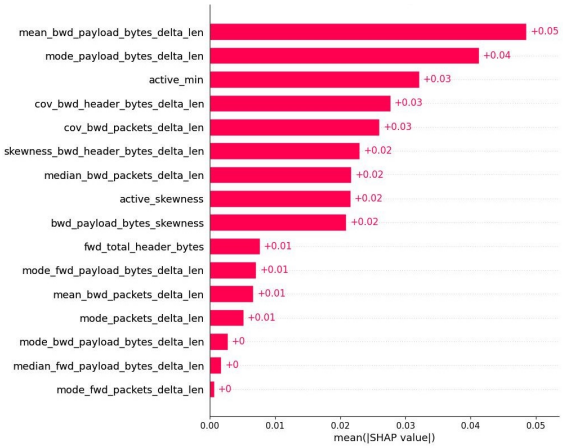
((b)) IRCBot



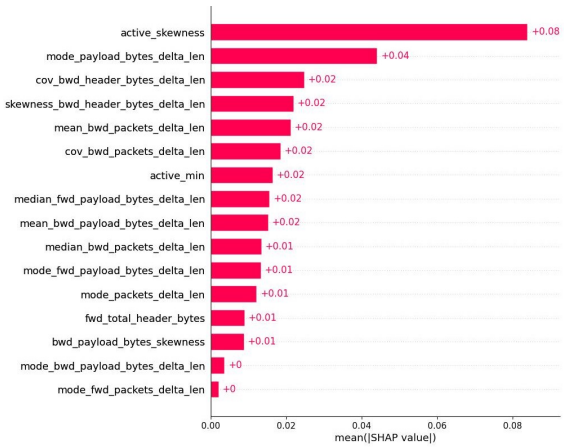
((c)) Kenjiro



((d)) Mirai



((e)) Linux Mirai



((f)) Okiru

Figure 26: Bar plot of different bot samples

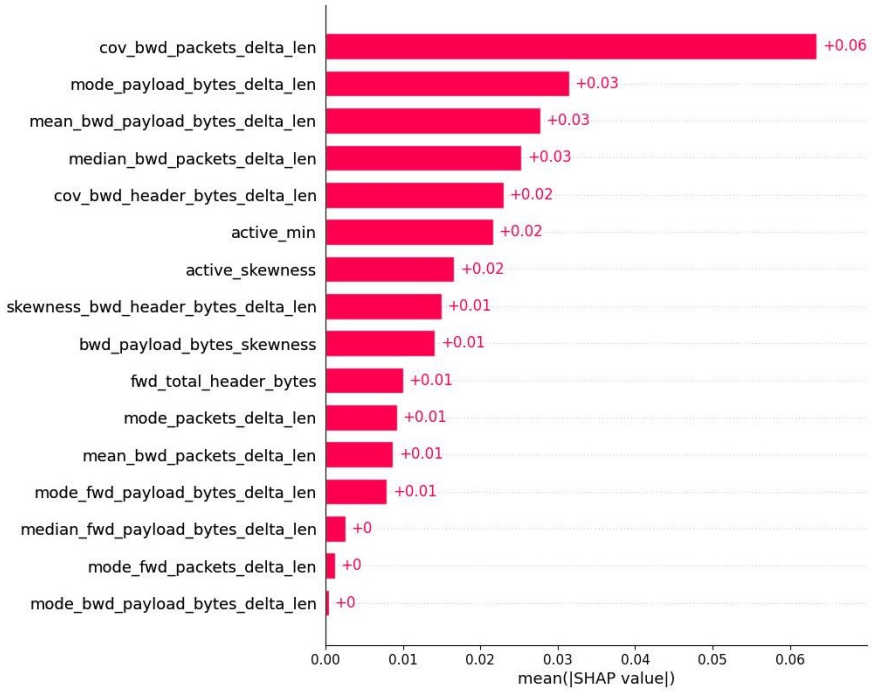


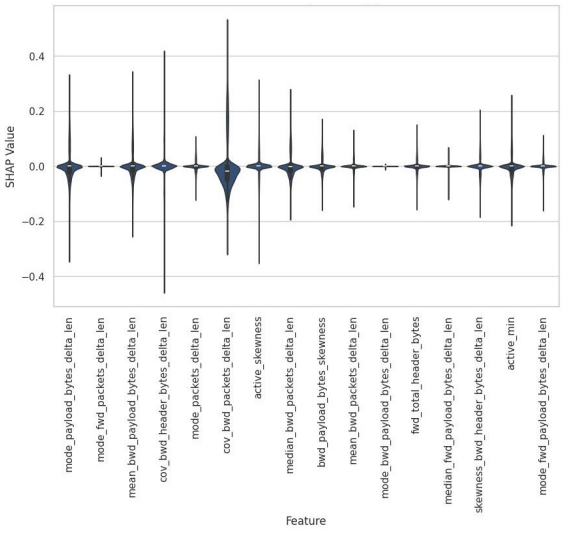
Figure 27: Bar plot of all bot samples

gradients along a path from a baseline input to the actual input, capturing both local and global feature importance. We analyzed 100,000 random samples, with approximately 16,666 samples per bot. SHAP bar and violin plots are used to visualize the bot profiles. Fig. 26 and 28 show the profiles for individual bots. Fig. 27 and Fig.29 show the collective profile of bot samples. The individual and collective profiles of the bots are discussed below.

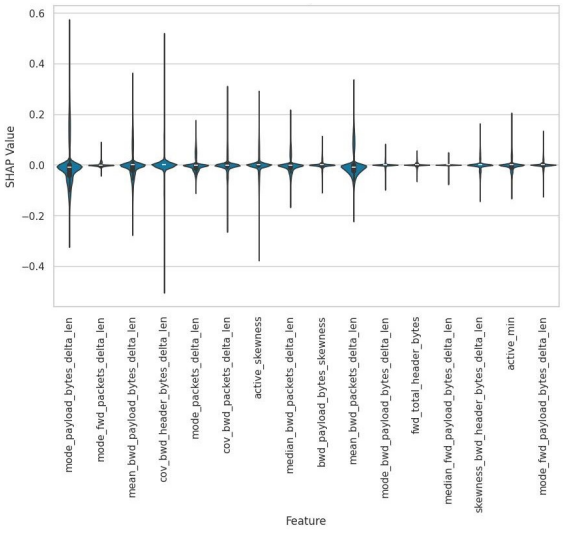
- **Gafgyt:** The analysis reveals that Gafgyt's network traffic is primarily characterized by distinctive patterns in packet payload size and length distributions, particularly in the backward direction (responses or exfiltration/traffic from the bot to the server). This is evident in both the SHAP violin and bar plots, where features related to payload size changes (e.g., *mean_bwd_payload_bytes_delta_len*) and backward packet lengths (e.g., *cov_bwd_packets_delta_len*) are consistently ranked as the most influential.

  Gafgyt's communication exhibits significant variability in payload sizes, as indicated by the wide distribution of *mode_payload_bytes_delta_len*. This means Gafgyt botnet communicates in a non-uniform way, with packets and payloads of varying sizes. This could be a deliberate attempt to evade detection by network security systems.

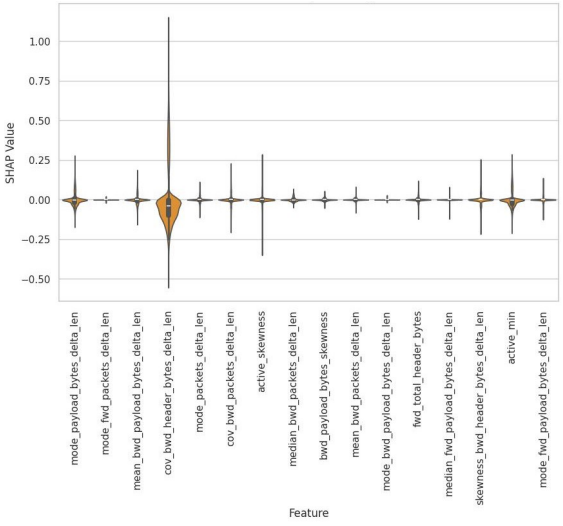
  The dominance of mode-based features (e.g.,*mode_payload_bytes_delta_len*) highlights the importance of analyzing the most frequent values within packet size and length distributions. This suggests that Gafgyt has preferred payload sizes and packet lengths that it uses more frequently, potentially revealing specific communication patterns or operational characteristics. While backward (response) traffic features are dominant in this model, some forward traffic features, such as *fwd_total_header_bytes*, also contribute to identifying Gafgyt traffic, indicating that analyzing both incoming and outgoing traffic



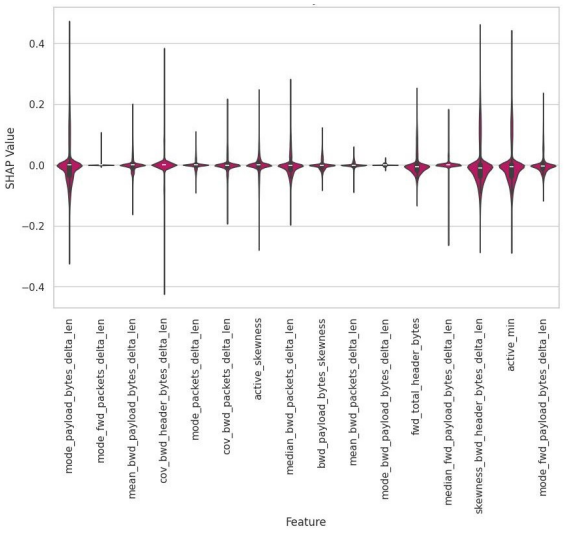
((a)) Gagfyt



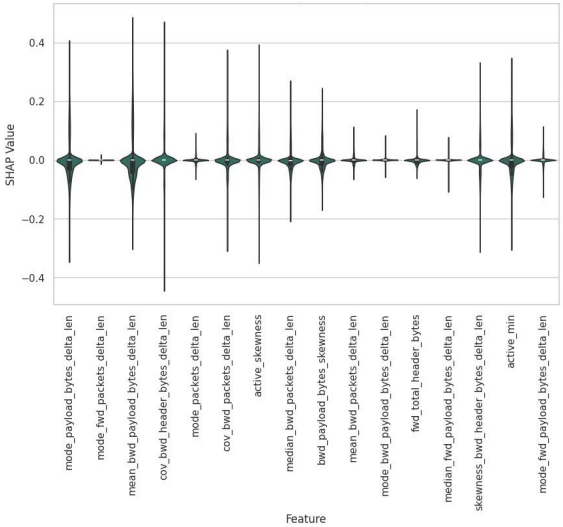
((b)) IRCBot



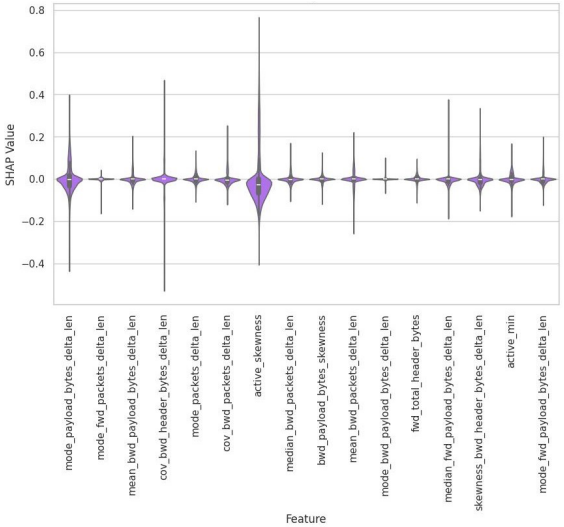
((c)) Kenjiro



((d)) Mirai



((e)) Linux Mirai



((f)) Okiru

Figure 28: Violin Summary plot of different bot samples

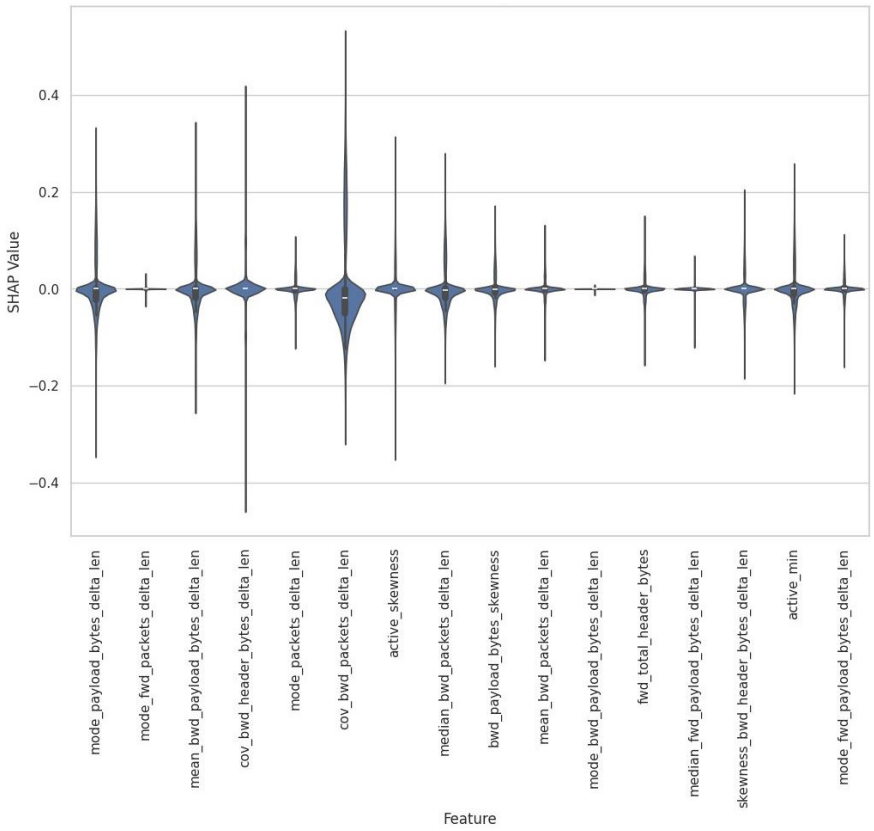


Figure 29: Violin plot of all bot samples

patterns is important for comprehensive detection.

Activity-based features like *active_min* and *active_skewness*, while not as dominant as payload and packet dynamics, contribute to the model's understanding of Gafgyt's behavior. The skewness and range of these features provide insights into the botnet's operational tempo, highlighting periods of minimal activity punctuated by bursts of communication. Additionally, the skewness of activity suggests that while Gafgyt maintains minimal activity levels, it occasionally exhibits bursts of higher activity, which could be indicative of C&C interactions or coordinated attacks.

- **IRCBot:** The IRC bot's behavior is primarily influenced by features related to backward traffic. Features like *mean_bwd_packets_delta_len* and *mean_bwd_payload_bytes_delta_len* consistently appear as top contributors. The analysis reveals that the size and variability of backward packets and payloads significantly influence the bot's behavior. Features like *mode_payload_bytes_delta_len* and *mean_bwd_payload_bytes_delta_len* emphasize the importance of payload characteristics. Additionally, *mean_bwd_packets_delta_len* and *cov_bwd_header_bytes_delta_len* highlight the relevance of overall packet structure and variability in header sizes.

  The prominence of *mode_payload_bytes_delta_len* and *mode_fwd_packets_delta_len* emphasizes that the bot frequently uses specific packet sizes and payloads, potentially for C&C or establishing covert channels. The violin plots show that the impact of these features can vary, suggesting the bot's flexibility in adapting its communication based on the context. While backward traffic is dominant, the

moderate influence of *mode_fwd_packets_delta_len* indicates that the bot actively sends requests or data, confirming a bidirectional communication pattern.

The feature *active_min* plays a less significant role in bot identification. However, *active_skewness* is of higher importance. Specifically, the skewness of activity suggests that while these bots maintain minimal activity levels, they occasionally exhibit bursts of higher activity, potentially indicative of C&C interactions or coordinated attacks to avoid detection or a need for periodic communication with a C&C server.

While features related to forward traffic are present, their lower SHAP values and narrower distributions in the violin plots indicate they are less influential than backward traffic features. This further reinforces the notion that the bot's responses are its most distinguishing characteristic.

- **Kenjiro:** Kenjiro's behavior is significantly influenced by features related to backward traffic, as highlighted by both bar and violin plots. The most influential features, such as *cov_bwd_header_bytes_delta_len* and *mean_bwd_payload_bytes_delta_len*, emphasize the dominance of backward traffic patterns. The plots also underscore the importance of changes in packet lengths (*cov_bwd_packets_delta_len*) and payload lengths (*mean_bwd_payload_bytes_delta_len*) in identifying Kenjiro's behavior, likely capturing the bot's communication patterns, data size, and how these change over time.

  The violin plots for features like *cov_bwd_header_bytes_delta_len* and *mode_payload_bytes_delta_len* show wide distributions, indicating significant variability. This suggests Kenjiro's behavior is adaptive, potentially to evade detection or optimize tasks. The analysis also reveals a notable emphasis on skewness in various aspects of Kenjiro's behavior. Features such as *active_skewness*, *bwd_payload_bytes_skewness*, and *skewness_bwd_header_bytes_delta_len* indicate that the bot is sensitive to asymmetrical distributions in its activity levels, payload sizes, and changes in header lengths. This focus on skewness reflects an adaptive strategy where the bot tailors its behavior to varying network conditions and traffic patterns, mimicking legitimate traffic to avoid detection while optimizing operations.

  Moreover, mode-based features focusing on packet and payload length changes suggest that Kenjiro aligns with the most common network traffic patterns, blending with legitimate traffic or optimizing operations based on frequent network conditions. Features like *active_min* and *active_skewness* highlight the importance of Kenjiro's activity level and its variability, implying that the bot adjusts its strategies based on the overall pace and fluctuations in network traffic.

  Although forward traffic features like *mode_fwd_packets_delta_len* are present, their lower SHAP values compared to backward traffic features indicate a lesser influence. This aligns with the observation of backward traffic dominance but suggests that analyzing forward traffic could still provide insights into the bot's behavior.

- **Mirai:**

  The SHAP plots reveal that Mirai botnet traffic is primarily characterized by specific patterns in activity duration and the distribution of backward header bytes length changes. These features, namely *active_min* and *skewness_bwd_header_bytes_delta_len*, stand out as the most influential in distinguishing

Mirai botnet traffic. The violin plot of *active_min* reveals a wide distribution of values for Mirai traffic, suggesting variations in the minimum duration of active periods across different botnet instances or campaigns. Furthermore, Mirai traffic tends to have short *active_min* values to evade detection by blending in with background noise.

Both the bar and violin plots highlight the significance of packet and payload size variability. Features like *mode_payload_bytes_delta_len* and *median_bwd_packets_delta_len* indicate that Mirai utilizes a variety of packet sizes, potentially to evade detection and blend with normal traffic. The high impact of features like *skewness_bwd_header_bytes_delta_len* and *bwd_payload_bytes_skewness* emphasizes the asymmetric distribution of packet sizes. This skewness could be a deliberate tactic to disguise C&C communications by making them appear less uniform. Moreover, the high importance of *active_skewness* shows that Mirai botnets could manipulate the skewness of their active time periods to blend in with normal traffic or to make their activity appear less suspicious.

The presence of features like *mode_payload_bytes_delta_len* in the top contributors suggests that the most frequent packet size changes play a crucial role in Mirai's communication patterns. This information could be leveraged to develop detection mechanisms that focus on identifying these characteristic packet size distributions. The feature *active_min* indicates that Mirai botnets often establish short-lived connections. This behavior aligns with the need for quick command transmission and updates while minimizing the risk of detection due to prolonged activity. While backward traffic features dominate, some forward traffic features like *fwd_total_header_bytes* also contribute significantly. This suggests that analyzing both incoming and outgoing traffic is essential for a comprehensive understanding of Mirai's behavior. The total size of headers in forward packets, for instance, might reveal specific C&C protocols or payloads.

- **Linux Mirai:**

  The analysis of Linux Mirai's behavior emphasizes the significance of backward traffic direction, as features related to backward payload size and packet dynamics consistently rank high in importance.

  The importance of packet and payload dynamics is also evident. Variability in features like *mode_payload_bytes_delta_len*, *mean_bwd_payload_bytes_delta_len*, and packet length indicates that Linux Mirai employs diverse strategies, varying payload sizes and packet lengths to evade detection and optimize its communication. The significant impact of features like *skewness_bwd_header_bytes_delta_len* highlights the asymmetric distribution of packet sizes. This skewness could be a deliberate tactic to disguise C&C communications by making them appear less uniform. Similarly, the high importance of *active_skewness* suggests that Linux Mirai botnets might manipulate the skewness of their active time periods to blend in with normal traffic or make their activity appear less suspicious.

  The prominence of mode-based features, such as *mode_payload_bytes_delta_len*, indicates that Linux Mirai's behavior is often characterized by the most frequent values (modes) of payload sizes and packet lengths. This suggests that Linux Mirai bots may exhibit distinct, repetitive patterns in their communication. While less dominant than backward traffic, forward traffic features like *fwd_total_header_bytes*

still play a role in characterizing the bot's behavior. This suggests that the initial outward communication, while potentially less informative than the return traffic, still carries some distinguishing features.

Additionally, the feature *active_min* being among the top features suggests that Linux Mirai bots often exhibit short-lived connection durations. This behavior aligns with the botnet's need for efficient C&C communication and rapid updates while minimizing the risk of detection associated with prolonged network activity.

- **Okiru:** The analysis of Okiru reveals a significant emphasis on backward traffic features.This is evident from the prominence of features like *cov_bwd_header_bytes_delta_len* in both bar and violin plots. The bot's communication patterns show considerable variability in packet and payload lengths, as highlighted by features like *mode_payload_bytes_delta_len*, which means that the Okiru botnet communicates in a non-uniform way to evade detection by network security systems.

  Additionally, the significant role of active skewness implies that Okiru botnets might manipulate the asymmetry of their activity to blend in with legitimate traffic or obscure malicious intent. This could involve introducing irregular communication intervals to evade detection systems relying on predictable patterns or normalizing the skewness to mimic typical network behavior. The high value of *skewness_bwd_header_bytes_delta_len* indicates an asymmetric distribution of packet sizes.

  The consistent presence of mode-based features underscores their importance in the bot's decision-making process, suggesting the bot is designed to identify and react to the most common patterns in network traffic. Features such as *active_min* and *active_skewness* highlight the significance of the bot's activity level and the variability of this activity. This sensitivity to activity levels implies that Okiru's behavior may be influenced by the overall pace and fluctuations in network traffic, potentially adapting its strategies based on the intensity of network communication.

  While backward traffic is dominant, forward traffic features like *mode_fwd_packets_delta_len* and *fwd_total_header_bytes* also contribute to characterizing the bot's behavior. This indicates that initial outgoing communication, though less informative than return traffic, still carries distinguishing features. The strong emphasis on backward traffic features suggests the bot primarily focuses on incoming data and its deviation from expected patterns.

Based on the analysis and Figs. 26 and 28, it can be inferred that almost all botnets primarily focused on backward traffic. This suggests that the responses generated by bots, rather than their initial requests, carry stronger signatures of their malicious nature. Also they share a common reliance on packet and payload-based features. The variations lie in specific metrics (e.g., length, bytes, skewness) and their relative importance. Features related to time also play a significant role across all botnets, highlighting the importance of temporal patterns in network traffic for botnet detection. Each botnet exhibits a few unique features that significantly contribute to its detection, which could be exploited for developing targeted detection mechanisms.

Figs. 27 and 29 show a collective profile of all bot samples. They indicate a clear dominance of backward traffic features in determining bot behavior, especially *cov_bwd_packets_delta_len*. The higher magnitude and

wider distribution of SHAP values for backward traffic features emphasize their importance in distinguishing bot traffic from legitimate traffic.

Features capturing the variability of packet and payload lengths, such as the covariance of delta lengths, consistently exhibit high importance. This implies that bots actively vary their packet and payload sizes, potentially as an evasion tactic to avoid detection by signature-based systems. Additionally, skewness features, particularly for backward payload bytes, indicate that the distribution of payload sizes in bot responses is often asymmetric and skewed. This could reflect the nature of the data being exfiltrated or commands being issued by the botmaster.

Mode-based features, such as *mode_payload_bytes_delta_len*, represent the most frequent change in payload size observed in network traffic. The consistent significance of these features suggests that bots may exhibit repetitive patterns in their payload size adjustments, possibly due to the nature of their communication. Features like *active_min*, which represent the minimum active time of a connection, reveal the importance of connection dynamics in bot identification. The wide distribution of SHAP values for this feature suggests that bots exhibit varying levels of activity, potentially reflecting different stages of their attack cycle or communication with different C&C servers.

The contrast between forward and backward traffic highlights the asymmetric nature of bot communication. While forward traffic may be more consistent with legitimate traffic patterns, backward traffic exhibits unique statistical properties and patterns that are highly indicative of bot activity. This asymmetry can be exploited for targeted detection strategies.

In summary, the SHAP analysis underscores the significance of analyzing both packet and payload features, particularly in backward traffic, for effective bot detection. The dominance of backward traffic, the importance of variability and skewness, the role of mode-based features, and the varying activity levels collectively paint a picture of bot behavior characterized by evasive tactics, asymmetric communication, and dynamic patterns. This understanding can inform the development of more sophisticated and robust bot detection mechanisms.

5.4 Evaluation comparison

Tables 5.6 and 5.7 compare the performance of the most used models whose performance has been evaluated on Aposemat-Bot-IoT-23 datasets in academic papers [166, 167, 168, 167, 169, 170]. As the tables indicate, the most used models include DT, RF, SVM, and XGBoost. In this order, we also evaluated the performance of these models on our generated augmented dataset. As Table 5.6 and 5.7 show, the overall and detailed performance of the models are almost the same. All the models, including the proposed model, reached an overall Accuracy, Precision, Recall, and F1 Score of 0.9999.

However, the proposed model has distinct benefits compared to other models. While tree-based systems like DT and RF can be effective, they are more prone to overfitting, especially with high-dimensional data or complex relationships. This tendency can limit their generalizability to new, unseen data [171, 172]. Moreover, these models often struggle with imbalanced datasets, common in real-world bot detection scenarios where malicious traffic is significantly outnumbered by legitimate traffic [173].

Conversely, the proposed LSTM-based model, with its ability to capture temporal dependencies and handle

Table 5.6: Models' performance comparison

system's name	Overall Accuracy	Overall Precision	Overall Recall	Overall F1 Score
Decision Tree [166, 167, 168]	0.9853	0.9853	0.9843	0.9788
Random Forest [166, 168]	0.9999	0.9999	0.9999	0.9999
SVM [166, 167]	0.9779	0.9878	0.9810	0.9811
XGBoost [169, 170]	0.9999	0.9999	0.9999	0.9999
Proposed system	0.9999	0.9999	0.9999	0.9999

sequential and time-series data, is well-suited for bot detection tasks. LSTMs can effectively model the patterns and behaviors exhibited in network traffic over time, making them robust against overfitting and capable of learning from imbalanced datasets. This inherent advantage positions the LSTM model as a more suitable choice for real-world bot detection systems, where the ability to generalize and detect subtle patterns in unbalanced data is crucial [174, 175].

The selected hyperparameters for the proposed LSTM-based model, derived through Amazon SageMaker experiments (Table 5.1), further demonstrate its suitability for edge devices. The choice of the 'Adam' optimizer, known for its efficiency and low memory requirements, aligns with the need for resource conservation on edge devices. Similarly, the 'relu' activation function, being computationally inexpensive, complements this approach. Moreover, the relatively small number of 'Units' (58) and a moderate 'Batch size' (192) indicate a system architecture that balances complexity with computational efficiency. This design choice, coupled with a limited number of 'Epochs' (9), signifies a system that can be trained rapidly and deployed with minimal overhead.

Collectively, these hyperparameters suggest an efficient LSTM-based model optimized for performance on IoT edge devices. This characteristic is crucial in IoT bot detection and identification scenarios, where real-time analysis and resource constraints necessitate accurate and computationally efficient models. The proposed LSTM-based model, with its tailored architecture and hyperparameter choices, addresses these requirements, enabling effective bot detection and identification even in resource-constrained environments.

In conclusion, the proposed IoT Bot detection and identification profiling model offers a promising solution for bot detection in IoT networks. Its ability to handle sequential and time-series data, manage imbalanced datasets, and its efficient architecture optimized for edge devices make it a valuable asset in enhancing the security and integrity of IoT ecosystems.

system's name	Label	Precision	Recall	F1 Score
Decision Tree	Gagfyt	0.9789	0.9898	0.9889
	Mirai	0.9899	0.9799	0.9699
	LinuxMirai	0.9798	0.9879	0.9787
	Kenjiro	0.9889	0.9799	0.9698
	Okiru	0.9869	0.9789	0.9856
	IRCBot	0.9878	0.9898	0.9799
Random Forest	Gagfyt	0.9999	1	0.9999
	Mirai	1	0.9999	0.9999
	LinuxMirai	1	0.9999	1
	Kenjiro	0.9999	1	1
	Okiru	0.9999	1	0.9999
	IRCBot	1	0.9999	1
SVM	Gagfyt	0.9799	0.9798	0.9897
	Mirai	0.9898	0.9799	0.9789
	LinuxMirai	0.9897	0.9889	0.9799
	Kenjiro	0.9799	0.9899	0.9698
	Okiru	0.9888	0.9679	0.9889
	IRCBot	0.9989	0.9796	0.9799
XGBoost	Gagfyt	1	0.9999	1
	Mirai	0.9999	0.9999	0.9999
	LinuxMirai	0.9999	0.9999	9.9999
	Kenjiro	1	1	0.9999
	Okiru	0.9999	0.9999	0.9999
	IRCBot	1	0.9999	1
Proposed system	Gagfyt	0.9999	0.9999	0.9998
	Mirai	0.9999	0.9999	0.9999
	LinuxMirai	0.9997	0.9999	0.9999
	Kenjiro	0.9999	0.9999	0.9999
	Okiru	0.9997	0.9998	0.9999
	IRCBot	0.9999	0.9998	0.9999

Table 5.7: System's performance on each bot sample after applying hyperparameter tuning with specific values

6 Conclusion

The proliferation of IoT devices has enhanced connectivity but exposed networks to new cyber threats, particularly from botnets. Detecting and identifying malicious data is critical for early threat detection, understanding attack patterns, and deploying countermeasures. This research proposes an optimized IoT Bot detection and identification profiling model using XAI. The proposed model utilizes a novel feature selection method with the XGBoost algorithm and a correlation-based feature selection technique to enhance efficiency. An optimized LSTM neural network enables accurate bot detection and identification, with hyperparameters selected using the Bayesian Optimization algorithm. SHAP analysis provides insightful individual and collective bot characteristic profiles.

The model, evaluated using the augmented BCCC-Aposemat-Bot-IoT-24 dataset, achieved outstanding performance, with accuracy, precision, recall, and F1-score all reaching 0.9999. The results were compared against established models primarily evaluated on the Aposemat-Bot-IoT-23 dataset in previous research (DT, RF, and SVM, and XGBoost). Compared to other models, it excels in handling sequential data, managing imbalanced datasets, and providing explainable insights into botnet behavior. Its computational efficiency makes it potentially suitable for deployment in resource-constrained environments.

Future work will focus on further optimizing the model to ensure it can be deployed on highly resource-constrained IoT devices. This includes exploring more efficient neural network architectures and reducing the computational overhead of the model. Additionally, expanding the dataset with more diverse and real-world IoT traffic scenarios will help improve the model's robustness and generalizability. Continued research will also investigate integrating the model with existing IoT security frameworks to provide a comprehensive security solution.

Candidate's full name: Sepideh Niktabe

University attended (with dates and degrees obtained):
Amirkabir University of Technology-Bachelor of Science in Computer Science 2015-2020

Publications:
Niktabe, S., Lashkari, A.H. Roudsari, A.H. Unveiling DoH tunnel: Toward generating a balanced DoH encrypted traffic dataset and profiling malicious behavior using inherently interpretable machine learning. Peer-to-Peer Netw. Appl. 17, 507–531 (2024). https://doi.org/10.1007/s12083-023-01597-4

Niktabe, S., Lashkari, A.H. Sharma, D.P. Detection, characterization, and profiling DoH Malicious traffic using statistical pattern recognition. Int. J. Inf. Secur. 23, 1293–1316 (2024). https://doi.org/10.1007/s10207-023-00790-z

Sepideh Niktabe, Arash Habibi Lashkari, "Unveiling IoT network Intruders behaviors: Profiling Malicious Bot activities on IoT Network Based on Explainable Artificial Intelligence (XAI) Using LSTM and SHAP", submitted to the International Journal of Information Security, June 2024.

Bibliography

[1] M. Mohy-eddine, A. Guezzaz, S. Benkirane, and M. Azrour, "An efficient network intrusion detection model for iot security using k-nn classifier and feature selection," *Multimedia Tools and Applications*, pp. 1–19, 2023.

[2] M. Douiba, S. Benkirane, A. Guezzaz, and M. Azrour, "An improved anomaly detection model for iot security using decision tree and gradient boosting," *The Journal of Supercomputing*, vol. 79, no. 3, pp. 3392–3411, 2023.

[3] A. Abbas, M. A. Khan, S. Latif, M. Ajaz, A. A. Shah, and J. Ahmad, "A new ensemble-based intrusion detection system for internet of things," *Arabian Journal for Science and Engineering*, pp. 1–15, 2021.

[4] S. Kumar and A. Kumar, "Image-based malware detection based on convolution neural network with autoencoder in industrial internet of things using software defined networking honeypot," *Engineering Applications of Artificial Intelligence*, vol. 133, p. 108374, 2024.

[5] O. B. J. Rabie, S. Selvarajan, T. Hasanin, A. M. Alshareef, C. Yogesh, and M. Uddin, "A novel iot intrusion detection framework using decisive red fox optimization and descriptive back propagated radial basis function models," *Scientific Reports*, vol. 14, no. 1, p. 386, 2024.

[6] H. C. Altunay and Z. Albayrak, "A hybrid cnn+ lstm based intrusion detection system for industrial iot networks," *Engineering Science and Technology, an International Journal*, vol. 38, p. 101322, 2023.

[7] S. Khan and A. B. Mailewa, "Discover botnets in iot sensor networks: A lightweight deep learning framework with hybrid self-organizing maps," *Microprocessors and Microsystems*, vol. 97, p. 104753, 2023.

[8] G. Zhao, C. Ren, J. Wang, Y. Huang, and H. Chen, "Iot intrusion detection model based on gated recurrent unit and residual network," *Peer-to-Peer Networking and Applications*, pp. 1–13, 2023.

[9] A. Dushimimana, T. Tao, R. Kindong, and A. Nishyirimbere, "Bi-directional recurrent neural network for intrusion detection system (ids) in the internet of things (iot)," *International Journal of Advanced Engineering Research and Science*, vol. 7, no. 3, 2020.

[10] I. Ullah and Q. H. Mahmoud, "A framework for anomaly detection in iot networks using conditional generative adversarial networks," *IEEE Access*, vol. 9, pp. 165907–165931, 2021.

[11] S. Moghanian, F. B. Saravi, G. Javidi, and E. O. Sheybani, "Goamlp: Network intrusion detection with multilayer perceptron and grasshopper optimization algorithm," *IEEE Access*, vol. 8, pp. 215202–215213, 2020.

[12] S. Ullah, J. Ahmad, M. A. Khan, E. H. Alkhammash, M. Hadjouni, Y. Y. Ghadi, F. Saeed, and N. Pitropakis, "A new intrusion detection system for the internet of things via deep convolutional neural network and feature engineering," *Sensors*, vol. 22, no. 10, p. 3607, 2022.

[13] M. Almiani, A. AbuGhazleh, A. Al-Rahayfeh, S. Atiewi, and A. Razaque, "Deep recurrent neural network for iot intrusion detection system," *Simulation Modelling Practice and Theory*, vol. 101, p. 102031, 2020.

[14] A. Elsaeidy, K. S. Munasinghe, D. Sharma, and A. Jamalipour, "Intrusion detection in smart cities using restricted boltzmann machines," *Journal of Network and Computer Applications*, vol. 135, pp. 76–83, 2019.

[15] A. Telikani and A. H. Gandomi, "Cost-sensitive stacked auto-encoders for intrusion detection in the internet of things," *Internet of Things*, vol. 14, p. 100122, 2021.

[16] N. Balakrishnan, A. Rajendran, D. Pelusi, and V. Ponnusamy, "Deep belief network enhanced intrusion detection system to prevent security breach in the internet of things," *Internet of things*, vol. 14, p. 100112, 2021.

[17] M. A. Ferrag, L. Maglaras, A. Ahmim, M. Derdour, and H. Janicke, "Rdtids: Rules and decision tree-based intrusion detection system for internet-of-things networks," *Future internet*, vol. 12, no. 3, p. 44, 2020.

[18] J. B. Awotunde, F. E. Ayo, R. Panigrahi, A. Garg, A. K. Bhoi, and P. Barsocchi, "A multi-level random forest model-based intrusion detection using fuzzy inference system for internet of things networks," *International Journal of Computational Intelligence Systems*, vol. 16, no. 1, p. 31, 2023.

[19] A. Khraisat, I. Gondal, P. Vamplew, J. Kamruzzaman, and A. Alazab, "A novel ensemble of hybrid intrusion detection system for detecting internet of things attacks," *Electronics*, vol. 8, no. 11, p. 1210, 2019.

[20] S. Niktabe, A. H. Lashkari, and A. H. Roudsari, "Unveiling doh tunnel: Toward generating a balanced doh encrypted traffic dataset and profiling malicious behavior using inherently interpretable machine learning," *Peer-to-Peer Networking and Applications*, vol. 17, no. 1, pp. 507–531, 2024.

[21] F. Sadikin, T. Van Deursen, and S. Kumar, "A zigbee intrusion detection system for iot using secure and efficient data collection," *Internet of Things*, vol. 12, p. 100306, 2020.

[22] M. F. Elrawy, A. I. Awad, and H. F. Hamed, "Intrusion detection systems for iot-based smart environments: a survey," *Journal of Cloud Computing*, vol. 7, no. 1, pp. 1–20, 2018.

[23] S. Tsimenidis, T. Lagkas, and K. Rantos, "Deep learning in iot intrusion detection," *Journal of network and systems management*, vol. 30, pp. 1–40, 2022.

[24] M. Abomhara and G. M. Koien, "Cyber security and the internet of things: vulnerabilities, threats, intruders and attacks," *Journal of Cyber Security and Mobility*, pp. 65–88, 2015.

[25] N. Vlajic and D. Zhou, "Iot as a land of opportunity for ddos hackers," *Computer*, vol. 51, no. 7, pp. 26–34, 2018.

[26] Q.-D. Ngo, H.-T. Nguyen, V.-H. Le, and D.-H. Nguyen, "A survey of iot malware and detection methods based on static features," *ICT Express*, vol. 6, no. 4, pp. 280–286, 2020.

[27] J. King and A. I. Awad, "A distributed security mechanism for resource-constrained iot devices," *Informatica*, vol. 40, no. 1, 2016.

[28] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in internet of things: The road ahead," *Computer networks*, vol. 76, pp. 146–164, 2015.

[29] M. Keshk, N. Koroniotis, N. Pham, N. Moustafa, B. Turnbull, and A. Y. Zomaya, "An explainable deep learning-enabled intrusion detection framework in iot networks," *Information Sciences*, vol. 639, p. 119000, 2023.

[30] A. Khraisat and A. Alazab, "A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges," *Cybersecurity*, vol. 4, pp. 1–27, 2021.

[31] A. Thakkar and R. Lohiya, "A review on machine learning and deep learning perspectives of ids for iot: recent updates, security issues, and challenges," *Archives of Computational Methods in Engineering*, vol. 28, pp. 3211–3243, 2021.

[32] G. Kocher and G. Kumar, "Machine learning and deep learning methods for intrusion detection systems: recent developments and challenges," *Soft Computing*, vol. 25, no. 15, pp. 9731–9763, 2021.

[33] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: A survey," *applied sciences*, vol. 9, no. 20, p. 4396, 2019.

[34] D. J. Livingstone, *Artificial neural networks: methods and applications*. Springer, 2008.

[35] J. Gu and S. Lu, "An effective intrusion detection approach using svm with na¨ive bayes feature embedding," *Computers & Security*, vol. 103, p. 102158, 2021.

[36] S. Manimurugan, "Iot-fog-cloud model for anomaly detection using improved na¨ive bayes and principal component analysis," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–10, 2021.

[37] D. Stiawan, M. E. Suryani, Susanto, M. Y. Idris, M. N. Aldalaien, N. Alsharif, and R. Budiarto, "Ping flood attack pattern recognition using a k-means algorithm in an internet of things (iot) network," *IEEE Access*, vol. 9, pp. 116475–116484, 2021.

[38] A. Krenker, J. Bevster, and A. Kos, "Introduction to the artificial neural networks," *Artificial Neural Networks: Methodological Advances and Biomedical Applications. InTech*, pp. 1–18, 2011.

[39] J. Singh and R. Banerjee, "A study on single and multi-layer perceptron neural network," in *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 35–40, 2019.

[40] D. Javeed, M. S. Saeed, M. Adil, P. Kumar, and A. Jolfaei, "A federated learning-based zero trust intrusion detection system for internet of things," *Ad Hoc Networks*, p. 103540, 2024.

[41] D. Stiawan, M. E. Suryani, Susanto, M. Y. Idris, M. N. Aldalaien, N. Alsharif, and R. Budiarto, "Ping flood attack pattern recognition using a k-means algorithm in an internet of things (iot) network," *IEEE Access*, vol. 9, pp. 116475–116484, 2021.

[42] N. Hu, Z. Tian, H. Lu, X. Du, and M. Guizani, "A multiple-kernel clustering based intrusion detection scheme for 5g and iot networks," *International Journal of Machine Learning and Cybernetics*, pp. 1–16, 2021.

[43] Y. Wang, *Statistical techniques for network security: modern statistically-based intrusion detection and protection: modern statistically-based intrusion detection and protection*. Igi Global, 2008.

[44] Q. Liu, V. Hagenmeyer, and H. B. Keller, "A review of rule learning-based intrusion detection systems and their prospects in smart grids," *IEEE Access*, vol. 9, pp. 57542–57564, 2021.

[45] S. Hajiheidari, K. Wakil, M. Badri, and N. J. Navimipour, "Intrusion detection systems in the internet of things: A comprehensive investigation," *Computer Networks*, vol. 160, pp. 165–191, 2019.

[46] X. Yang, G. Peng, D. Zhang, Y. Lv, *et al.*, "An enhanced intrusion detection system for iot networks based on deep learning and knowledge graph," *Security and Communication Networks*, vol. 2022, 2022.

[47] C. Stylianopoulos, S. Kindström, M. Almgren, O. Landsiedel, and M. Papatriantafilou, "Co-evaluation of pattern matching algorithms on iot devices with embedded gpus," in *Proceedings of the 35th Annual Computer Security Applications Conference*, pp. 17–27, 2019.

[48] Y. Fu, Z. Yan, J. Cao, O. Koné, X. Cao, *et al.*, "An automata based intrusion detection metchod for internet of things," *Mobile Information Systems*, vol. 2017, 2017.

[49] M. J. Babu and A. R. Reddy, "Sh-ids: specification heuristics based intrusion detection system for iot networks," *Wireless Personal Communications*, vol. 112, no. 3, pp. 2023–2045, 2020.

[50] A. Verma and V. Ranga, "Machine learning based intrusion detection systems for iot applications," *Wireless Personal Communications*, vol. 111, pp. 2287–2310, 2020.

[51] O. Can and O. K. Sahingoz, "A survey of intrusion detection systems in wireless sensor networks," in *2015 6th international conference on modeling, simulation, and applied optimization (ICMSAO)*, pp. 1–6, IEEE, 2015.

[52] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: Lstm cells and network architectures," *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[53] J. Schmidhuber, S. Hochreiter, *et al.*, "Long short-term memory," *Neural Comput*, vol. 9, no. 8, pp. 1735–1780, 1997.

[54] M. D. Hossain, H. Inoue, H. Ochiai, D. Fall, and Y. Kadobayashi, "Lstm-based intrusion detection system for in-vehicle can bus communications," *Ieee Access*, vol. 8, pp. 185489–185502, 2020.

[55] K. Greff, R. K. Srivastava, J. Koutn´ik, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2016.

[56] M. Watson, C. Qian, J. Bischof, F. Chollet, *et al.*, "Kerasnlp." https://github.com/keras-team/keras-nlp, 2022.

[57] N. Reimers and I. Gurevych, "Optimal hyperparameters for deep lstm-networks for sequence labeling tasks," *arXiv preprint arXiv:1707.06799*, 2017.

[58] S. Merity, N. S. Keskar, and R. Socher, "Regularizing and optimizing lstm language models," *arXiv preprint arXiv:1708.02182*, 2017.

[59] B. Bischl, M. Binder, M. Lang, T. Pielok, J. Richter, S. Coors, J. Thomas, T. Ullmann, M. Becker, A.-L. Boulesteix, *et al.*, "Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 13, no. 2, p. e1484, 2023.

[60] T. Yu and H. Zhu, "Hyper-parameter optimization: A review of algorithms and applications," *arXiv preprint arXiv:2003.05689*, 2020.

[61] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark," *Neurocomputing*, vol. 503, pp. 92–108, 2022.

[62] J. Lederer, "Activation functions in artificial neural networks: A systematic overview," *arXiv preprint arXiv:2101.09957*, 2021.

[63] J. Frankle, D. J. Schwab, and A. S. Morcos, "The early phase of neural network training," *arXiv preprint arXiv:2002.10365*, 2020.

[64] B. Khemani, S. Patil, K. Kotecha, and S. Tanwar, "A review of graph neural networks: concepts, architectures, techniques, challenges, datasets, applications, and future directions," *Journal of Big Data*, vol. 11, no. 1, p. 18, 2024.

[65] F. M. Shiri, T. Perumal, N. Mustapha, and R. Mohamed, "A comprehensive overview and comparative analysis on deep learning models: Cnn, rnn, lstm, gru," *arXiv preprint arXiv:2305.17473*, 2023.

[66] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, pmlr, 2015.

[67] A. Devarakonda, M. Naumov, and M. Garland, "Adabatch: Adaptive batch sizes for training deep neural networks," *arXiv preprint arXiv:1712.02029*, 2017.

[68] M. S. Nakhodnov, M. S. Kodryan, E. M. Lobacheva, and D. S. Vetrov, "Loss function dynamics and landscape for deep neural networks trained with quadratic loss," in *Doklady Mathematics*, vol. 106, pp. S43–S62, Springer, 2022.

[69] A. Tushar, "Making sense of hidden layer information in deep networks by learning hierarchical targets," *arXiv preprint arXiv:1505.00384*, 2015.

[70] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, "Dive into deep learning," *arXiv preprint arXiv:2106.11342*, 2021.

[71] H. R. Sayegh, W. Dong, and A. M. Al-madani, "Enhanced intrusion detection with lstm-based model, feature selection, and smote for imbalanced data," *Applied Sciences*, vol. 14, no. 2, p. 479, 2024.

[72] F. Laghrissi, S. Douzi, K. Douzi, and B. Hssina, "Intrusion detection systems using long short-term memory (lstm)," *Journal of Big Data*, vol. 8, no. 1, p. 65, 2021.

[73] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[74] R. Younisse, A. Ahmad, and Q. Abu Al-Haija, "Explaining intrusion detection-based convolutional neural networks using shapley additive explanations (shap)," *Big Data and Cognitive Computing*, vol. 6, no. 4, p. 126, 2022.

[75] S. Hariharan, R. Rejimol Robinson, R. R. Prasad, C. Thomas, and N. Balakrishnan, "Xai for intrusion detection system: comparing explanations based on global and local scope," *Journal of Computer Virology and Hacking Techniques*, vol. 19, no. 2, pp. 217–239, 2023.

[76] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, "Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks.," in *IJCAI*, vol. 2, p. 8, 2019.

[77] M. Frye, J. Mohren, and R. H. Schmitt, "Benchmarking of data preprocessing methods for machine learning-applications in production," *Procedia CIRP*, vol. 104, pp. 50–55, 2021.

[78] D. Theng and K. K. Bhoyar, "Feature selection techniques for machine learning: a survey of more than two decades of research," *Knowledge and Information Systems*, vol. 66, no. 3, pp. 1575–1637, 2024.

[79] W.-B. Wu, S.-B. Chen, C. Ding, and B. Luo, "Non-linear feature selection based on convolution neural networks with sparse regularization," *Cognitive Computation*, vol. 16, no. 2, pp. 654–670, 2024.

[80] S. Ledesma, G. Cerda, G. Avina, D. Hernandez, and M. Torres, "Feature selection using artificial neural networks," in *MICAI 2008: Advances in Artificial Intelligence: 7th Mexican International Conference on Artificial Intelligence, Atizapán de Zaragoza, Mexico, October 27-31, 2008 Proceedings 7*, pp. 351–359, Springer, 2008.

[81] J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent," *Journal of statistical software*, vol. 33, no. 1, p. 1, 2010.

[82] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

[83] Y. Wang and X. S. Ni, "A xgboost risk model via feature selection and bayesian hyper-parameter optimization," *arXiv preprint arXiv:1901.08433*, 2019.

[84] A. H., "A guide to the appropriate use of correlation coefficient in medical research," *Turkish journal of emergency medicine*, vol. 18, pp. 91–93, 2018.

[85] H. Akoglu, "User's guide to correlation coefficients," *Turkish journal of emergency medicine*, vol. 18, no. 3, pp. 91–93, 2018.

[86] S. S. D. D. Gregorich, M. and G. Heinze, "Regression with highly correlated predictors: Variable omission is not the solution.," *International journal of environmental research and public health*, vol. 18, 04 2021.

[87] K. Jerabek, K. Hynek, O. Rysavy, and I. Burgetova, "Dns over https detection using standard flow telemetry," *IEEE Access*, vol. 11, pp. 50000–50012, 2023.

[88] H. Yu and A. D. Hutson, "A robust spearman correlation coefficient permutation test," *Communications in Statistics-Theory and Methods*, pp. 1–13, 2022.

[89] S. L. Schober P, Boer C, "Correlation coefficients: Appropriate use and interpretation," *ANESTHESIA ANALGESIA*, vol. 126, p. 1763–1768, 05 2018.

[90] M. M. Mukaka, "A guide to appropriate use of correlation coefficient in medical research," *Malawi medical journal*, vol. 24, no. 3, pp. 69–71, 2012.

[91] B. Kaur, S. Dadkhah, F. Shoeleh, E. C. P. Neto, P. Xiong, S. Iqbal, P. Lamontagne, S. Ray, and A. A. Ghorbani, "Internet of things (iot) security dataset evolution: Challenges and future directions," *Internet of Things*, p. 100780, 2023.

[92] S. Garcia, A. Parmisano, and M. J. Erquiaga, "Iot-23: A labeled dataset with malicious and benign iot network traffic," *Stratosphere Lab., Praha, Czech Republic, Tech. Rep*, 2020.

[93] A. Parmisano, S. Garcia, and M. J. Erquiaga, "A labeled dataset with malicious and benign iot network traffic," *Stratosphere Laboratory: Praha, Czech Republic*, 2020.

[94] I. Ullah and Q. H. Mahmoud, "A scheme for generating a dataset for anomalous activity detection in iot networks," in *Canadian conference on artificial intelligence*, pp. 508–520, Springer, 2020.

[95] H. Kang, D. H. Ahn, G. M. Lee, J. D. Yoo, K. H. Park, and H. K. Kim, "Iot network intrusion dataset," 2019.

[96] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset," *Future Generation Computer Systems*, vol. 100, pp. 779–796, 2019.

[97] N. Moustafa, "A new distributed architecture for evaluating ai-based security systems at the edge: Network ton_iot datasets," *Sustainable Cities and Society*, vol. 72, p. 102994, 2021.

[98] M. Sarhan, S. Layeghy, N. Moustafa, and M. Portmann, "Netflow datasets for machine learning-based network intrusion detection systems," in *Big Data Technologies and Applications: 10th EAI International Conference, BDTA 2020, and 13th EAI International Conference on Wireless Internet, WiCON 2020, Virtual Event, December 11, 2020, Proceedings 10*, pp. 117–135, Springer, 2021.

[99] M. Sarhan, S. Layeghy, and M. Portmann, "Towards a standard feature set for network intrusion detection system datasets," *Mobile networks and applications*, pp. 1–14, 2022.

[100] M. Sarhan, S. Layeghy, and M. Portmann, "Evaluating standard feature sets towards increased generalisability and explainability of ml-based network intrusion detection," *Big Data Research*, vol. 30, p. 100359, 2022.

[101] M. Al-Hawawreh, E. Sitnikova, and N. Aboutorab, "X-iiotid: A connectivity-agnostic and device-agnostic intrusion data set for industrial internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3962–3977, 2021.

[102] M. Zolanvari, "Wustl-iiot-2021," 2021.

[103] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-iiotset: A new comprehensive realistic cyber security dataset of iot and iiot applications: Centralized and federated learning," 2022.

[104] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-baiot—network-based detection of iot botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.

[105] A. Guerra-Manzanares, J. Medina-Galindo, H. Bahsi, and S. Nõmm, "Medbiot: Generation of an iot botnet dataset in a medium-sized iot network.," in *ICISSP*, pp. 207–218, 2020.

[106] OCSLab, "OCSLab Datasets." https://ocslab.hksecurity.net/Datasets.

[107] H. Hindy, C. Tachtatzis, R. Atkinson, E. Bayne, and X. Bellekens, "Mqtt-iot-ids2020: Mqtt internet of things intrusion detection dataset," 2020.

[108] N. Moustafa, M. Keshky, E. Debiez, and H. Janicke, "Federated ton_iot windows datasets for evaluating ai-based security applications," in *2020 IEEE 19th international conference on trust, security and privacy in computing and communications (TrustCom)*, pp. 848–855, IEEE, 2020.

[109] S. Dadkhah, H. Mahdikhani, P. K. Danso, A. Zohourian, K. A. Truong, and A. A. Ghorbani, "Towards the development of a realistic multidimensional iot profiling dataset," in *2022 19th Annual International Conference on Privacy, Security & Trust (PST)*, pp. 1–11, IEEE, 2022.

[110] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, "Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment," *Sensors*, vol. 23, no. 13, p. 5941, 2023.

[111] dataset-of-legitimate-iot-data, "dataset-of-legitimate-iot-data." https://www.data.gouv.fr/en/datasets/dataset-of-legitimate-iot-data//information.

[112] D. Stiawan, "Tcp fin flood and zbassocflood dataset," 2021.

[113] F. Hussain, S. G. Abbas, G. A. Shah, I. M. Pires, U. U. Fayyaz, F. Shahzad, N. M. Garcia, and E. Zdravevski, "Iot healthcare security dataset," 2021.

[114] V. Graveto, P. Simões, and T. Cruz, "A dataset bundle for building automation and control systems security analysis," 2022.

[115] M. Shafi, A. H. Lashkari, V. Rodriguez, and R. Nevo, "Toward generating a new cloud-based distributed denial of service (ddos) dataset and cloud intrusion traffic characterization," *Information*, vol. 15, no. 4, p. 195, 2024.

[116] T. Z. Project, "Zeek: Network security monitoring."

[117] B. Krawczyk, "Learning from imbalanced data: Open challenges and future directions," *Progress in Artificial Intelligence*, vol. 5, 04 2016.

[118] V. Werner de Vargas, J. A. Schneider Aranda, R. dos Santos Costa, P. R. da Silva Pereira, and J. L. Victória Barbosa, "Imbalanced data preprocessing techniques for machine learning: a systematic mapping study," *Knowledge and Information Systems*, vol. 65, 01 2023.

[119] A. Jadhav, S. M. Mostafa, H. Elmannai, and F. K. Karim, "An empirical assessment of performance of data balancing techniques in classification task," *Applied Sciences*, vol. 12, no. 8, 2022.

[120] G. Y. Wong, F. H. Leung, and S.-H. Ling, "A novel evolutionary preprocessing method based on over-sampling and under-sampling for imbalanced datasets," in *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*, pp. 2354–2359, 2013.

[121] J. M. Moyano, E. L. Gibaja, K. J. Cios, and S. Ventura, "Review of ensembles of multi-label classifiers: models, experimental study and prospects," *Information Fusion*, vol. 44, pp. 33–45, 2018.

[122] G. Haixiang, Y. Li, J. Shang, G. Mingyun, H. Yuanyue, and B. Gong, "Learning from class-imbalanced data: Review of methods and applications," *Expert Systems with Applications*, vol. 73, 12 2016.

[123] N. Rout, D. Mishra, and M. K. Mallick, "Handling imbalanced data: a survey," in *International Proceedings on Advances in Soft Computing, Intelligent Systems and Applications: ASISA 2016*, pp. 431–443, Springer, 2018.

[124] R. Mohammed, J. Rawashdeh, and M. Abdullah, "Machine learning with oversampling and undersampling techniques: overview study and experimental results," in *2020 11th international conference on information and communication systems (ICICS)*, pp. 243–248, IEEE, 2020.

[125] A. Hanskunatai, "A new hybrid sampling approach for classification of imbalanced datasets," in *2018 3rd International Conference on Computer and Communication Systems (ICCCS)*, pp. 67–71, IEEE, 2018.

[126] S. C. Rathi, S. Misra, R. Colomo-Palacios, R. Adarsh, L. B. M. Neti, and L. Kumar, "Empirical evaluation of the performance of data sampling and feature selection techniques for software fault prediction," *Expert Systems with Applications*, vol. 223, p. 119806, 2023.

[127] M. Zheng, T. Li, X. Zheng, Q. Yu, C. Chen, D. Zhou, C. Lv, and W. Yang, "Uffdfr: Undersampling framework with denoising, fuzzy c-means clustering, and representative sample selection for imbalanced data classification," *Information Sciences*, vol. 576, pp. 658–680, 2021.

[128] W.-C. Lin, C.-F. Tsai, Y.-H. Hu, and J.-S. Jhang, "Clustering-based undersampling in class-imbalanced data," *Information Sciences*, vol. 409-410, pp. 17–26, 2017.

[129] M. Kubat, "Addressing the curse of imbalanced training sets: One-sided selection," *Fourteenth International Conference on Machine Learning*, 06 2000.

[130] N. Lunardon, G. Menardi, and N. Torelli, "Rose: a package for binary imbalanced learning," *R Journal*, vol. 6, pp. 79–89, 06 2014.

[131] K. W. Bowyer, N. V. Chawla, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority oversampling technique," *CoRR*, vol. abs/1106.1813, 2011.

[132] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "Smoteboost: Improving prediction of the minority class in boosting," in *Knowledge Discovery in Databases: PKDD 2003*, (Berlin, Heidelberg), pp. 107–119, Springer Berlin Heidelberg, 2003.

[133] S. Hu, Y. Liang, L. Ma, and Y. He, "Msmote: Improving classification performance when training data is imbalanced," in *2009 Second International Workshop on Computer Science and Engineering*, vol. 2, pp. 13–17, 2009.

[134] S. Barua, M. M. Islam, X. Yao, and K. Murase, "Mwmote–majority weighted minority oversampling technique for imbalanced data set learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 2, pp. 405–425, 2014.

[135] A. Jadhav, "Clustering based data preprocessing technique to deal with imbalanced dataset problem in classification task," in *2018 IEEE Punecon*, pp. 1–7, 2018.

[136] Amazon Web Services, "Amazon sagemaker." https://aws.amazon.com/sagemaker/.

[137] Amazon Web Services, "Amazon s3 user guide." https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.h

[138] Amazon Web Services, "Available amazon sagemaker notebook instance types."

[139] Amazon Web Services, "Amazon ec2 concepts."

[140] Amazon Web Services, "Amazon sagemaker pricing." https://aws.amazon.com/sagemaker/pricing/.

[141] A. Mao, M. Mohri, and Y. Zhong, "Cross-entropy loss functions: Theoretical analysis and applications," in *International Conference on Machine Learning*, pp. 23803–23828, PMLR, 2023.

[142] A. M. T. with SageMaker.

[143] B. Bischl, M. Binder, M. Lang, T. Pielok, J. Richter, S. Coors, J. Thomas, T. Ullmann, M. Becker, A.-L. Boulesteix, *et al.*, "Hyperparameter optimization: Foundations, algorithms, best practices and open challenges. arxiv," *arXiv preprint arXiv:2107.05847*, 2021.

[144] R. Turner, D. Eriksson, M. McCourt, J. Kiili, E. Laaksonen, Z. Xu, and I. Guyon, "Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020," in *NeurIPS 2020 Competition and Demonstration Track*, pp. 3–26, PMLR, 2021.

[145] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[146] R. Joshi, R. S. Somesula, and S. Katkoori, "Empowering resource-constrained iot edge devices: A hybrid approach for edge data analysis," in *IFIP International Internet of Things Conference*, pp. 168–181, Springer, 2023.

[147] E. M. Dogo, O. Afolabi, N. Nwulu, B. Twala, and C. Aigbavboa, "A comparative analysis of gradient descent-based optimization algorithms on convolutional neural networks," in *2018 international conference on computational techniques, electronics and mechanical systems (CTEMS)*, pp. 92–99, IEEE, 2018.

[148] B. Ding, H. Qian, and J. Zhou, "Activation functions and their characteristics in deep neural networks," in *2018 Chinese control and decision conference (CCDC)*, pp. 1836–1841, IEEE, 2018.

[149] A. Woodiss-Field, M. N. Johnstone, and P. Haskell-Dowland, "Examination of traditional botnet detection on iot-based bots," *Sensors*, vol. 24, no. 3, p. 1027, 2024.

[150] K. Hayawi, S. Saha, M. M. Masud, S. S. Mathew, and M. Kaosar, "Social media bot detection with deep learning methods: a systematic review," *Neural Computing and Applications*, vol. 35, no. 12, pp. 8903–8918, 2023.

[151] D. Calvaresi, S. Eggenschwiler, Y. Mualla, M. Schumacher, and J.-P. Calbimonte, "Exploring agent-based chatbots: a systematic literature review," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, no. 8, pp. 11207–11226, 2023.

[152] A. Marzano, D. Alexander, O. Fonseca, E. Fazzion, C. Hoepers, K. Steding-Jessen, M. H. Chaves, Í. Cunha, D. Guedes, and W. Meira, "The evolution of bashlite and mirai iot botnets," in *2018 IEEE Symposium on Computers and Communications (ISCC)*, pp. 00813–00818, IEEE, 2018.

[153] R. S. S. Moorthy and N. Nathiya, "Botnet detection using artificial intelligence," *Procedia Computer Science*, vol. 218, pp. 1405–1413, 2023.

[154] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, *et al.*, "Understanding the mirai botnet," in *26th USENIX security symposium (USENIX Security 17)*, pp. 1093–1110, 2017.

[155] H. Griffioen and C. Doerr, "Examining mirai's battle over the internet of things," in *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pp. 743–756, 2020.

[156] G. Kambourakis, C. Kolias, and A. Stavrou, "The mirai botnet and the iot zombie armies," in *MILCOM 2017-2017 IEEE military communications conference (MILCOM)*, pp. 267–272, IEEE, 2017.

[157] G.-J. Ahn, N. Paxton, and K. Pearson, "Understanding irc bot behaviors in network-centric attack detection and prevention framework," in *3rd International Conference on Information Warfare and Security, ICIW 2008*, pp. 1–9, Academic Conferences Limited, 2008.

[158] B. AsSadhan, A. Bashaiwth, J. Al-Muhtadi, and S. Alshebeili, "Analysis of p2p, irc and http traffic for botnets detection," *Peer-to-Peer Networking and Applications*, vol. 11, pp. 848–861, 2018.

[159] A. H. R. A. Awadi and B. Belaton, "Multi-phase irc botnet and botnet behavior detection model," *arXiv preprint arXiv:1501.03241*, 2015.

[160] C. Cimpanu, "New hakai iot botnet takes aim at d-link, huawei, and realtek routers," *ZDNet. url: https://www. zdnet. com/article/new-hakai-iot-botnettakes-aim-at-d-link-huawei-and-realtek-routers/(visited on 04/13/2021)*, 2018.

[161] Y. Xu, H. Koide, D. V. Vargas, and K. Sakurai, "Tracing mirai malware in networked system," in *2018 sixth international symposium on computing and networking workshops (CANDARW)*, pp. 534–538, IEEE, 2018.

[162] The Hacker News, "New mirai okiru botnet targets arc-based iot devices." https://thehackernews.com/2018/01/mirai-okiru-arc-botnet.html, 2018.

[163] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, and D. Garant, "Botnet detection based on traffic behavior analysis and flow intervals," *computers & security*, vol. 39, pp. 2–16, 2013.

[164] F. Haddadi, D. Le Cong, L. Porter, and A. N. Zincir-Heywood, "On the effectiveness of different botnet detection approaches," in *Information Security Practice and Experience: 11th International Conference, ISPEC 2015, Beijing, China, May 5-8, 2015, Proceedings*, pp. 121–135, Springer, 2015.

[165] K. Snow, F. Monrose, and M. Antonakakis, "The circle of life: A large-scale study of the iot malware lifecycle,"

[166] N.-A. Stoian, "Machine learning for anomaly detection in iot networks: Malware analysis on the iot-23 data set," B.S. thesis, University of Twente, 2020.

[167] F. Jeelani, D. S. Rai, A. Maithani, and S. Gupta, "The detection of iot botnet using machine learning on iot-23 dataset," in *2022 2nd International Conference on Innovative Practices in Technology and Management (ICIPTM)*, vol. 2, pp. 634–639, IEEE, 2022.

[168] A. Nascita, F. Cerasuolo, D. Di Monda, J. T. A. Garcia, A. Montieri, and A. Pescapè, "Machine and deep learning approaches for iot attack classification," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 1–6, IEEE, 2022.

[169] M. A. da Cruz, L. R. Abbade, P. Lorenz, S. B. Mafra, and J. J. Rodrigues, "Detecting compromised iot devices through xgboost," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[170] S. Garg, V. Kumar, and S. R. Payyavula, "Identification of internet of things (iot) attacks using gradient boosting: a cross dataset approach," *Telematique*, vol. 21, no. 1, pp. 6982–7012, 2022.

[171] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*, vol. 2. Springer, 2009.

[172] J. Ali, R. Khan, N. Ahmad, and I. Maqsood, "Random forests and decision trees," *International Journal of Computer Science Issues (IJCSI)*, vol. 9, no. 5, p. 272, 2012.

[173] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[174] A. Graves and A. Graves, "Long short-term memory," *Supervised sequence labelling with recurrent neural networks*, pp. 37–45, 2012.

[175] S. Parshutin, A. Kirshners, Y. Kornijenko, V. Zabiniako, M. Gasparovica-Asite, and A. Rozkalns, "Classification with lstm networks in user behaviour analytics with unbalanced environment," *Automatic Control and Computer Sciences*, vol. 55, pp. 85–91, 2021.