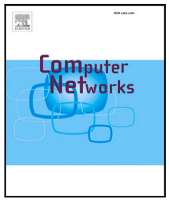
Computer Networks 251 (2024) 110603

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



FAMTDS: A novel MFO-based fully automated malicious traffic detection system for multi-environment networks

Furqan Rustam [a,*], Wajdi Aljedaani [b], Mahmoud Said Elsayed [a], Anca Delia Jurcut [a,**]

[a] School of Computer Science, University College Dublin, Belfield, D04 V1W8, Ireland
[b] University of North Texas, Denton, TX 76205, USA

ARTICLE INFO

Dataset link: https://github.com/furqanrustam/FAMTD-Malicious-Traffic-Detection

Keywords:
Multi-environment traffic
Moth flame optimizer
Malicious traffic detection
Machine learning
Zero-day attacks

ABSTRACT

Multi-environment networks, such as those in smart homes, handle both IoT and traditional IP-based traffic. Weak security protocols in IoT devices and the diverse traffic flow make these networks vulnerable to security breaches. This study delves into this pressing challenge and presents a pioneering solution—FAMTDS (Fully Automated Malicious Traffic Detection System)—designed explicitly for multi-environment networks. FAMTDS addresses the critical need for robust security measures by intelligently analyzing the amalgamation of IoT and IP-based traffic. FAMTDS comprises three pivotal stages: innovative multi-environment dataset creation, optimization of machine learning model hyperparameters, and holistic system optimization. For the multi-environment dataset, we amalgamate two prominent open-source datasets, UNSW-NB-15 and IoTID-20. Initially, crucial features are extracted from both datasets using an extra trees classifier. Subsequently, an equal number of features is generated by employing a neural network to harmonize these datasets. The resulting multi-environment dataset encompasses 19 distinct attack types, a comprehensive inclusion unprecedented in prior research on malicious traffic detection. This dataset exhibits diversity owing to its varied traffic samples, addressing a crucial gap in existing studies. To accommodate this diversity, machine learning models are deployed with fine-tuned hyperparameters. The Mouth Flame Optimizer streamlines feature extraction, feature generation, and hyperparameter tuning, automating the optimization process. FAMTDS demonstrates exceptional performance, achieving an accuracy score of 0.85 for the multi-environment dataset. We also integrated the CICDDOS2019 dataset with IoTID-20 in our multi-environment dataset, achieving a notable accuracy of 0.82 against recent attacks, thus enhancing our approach's validation. To further validate the generalizability of our proposed approach, we applied it to zero-day attack prediction. Our method demonstrated an accuracy of 0.84 for zero-day attacks, indicating its effectiveness in detecting newly emerging threats in multi-environment networks.

1. Introduction

Our society is advancing towards the creation of a smarter and more automated world. However, this technological progress also fuels the rise of cybercrimes. While technology greatly simplifies human life, it concurrently provides hackers with opportunities to easily access people's data without physical involvement [1,2]. According to [3], a total of 4100 data breaches occurred in 2022, exposing approximately 22 billion records. Another report [4] forecasts that cybercrime will cost worldwide companies approximately 10.5 trillion annually by 2025, a significant increase from 3 trillion in 2015, indicating a growth rate of 15% per year. These cyber-attacks not only affect companies but also target smart homes and other connected systems. Smart homes are particularly vulnerable to attacks, as noted in [5], with around 40.8% of homes equipped with smart systems being susceptible to cyber-attacks. These statistics indicate the high-risk exposure faced by both traditional IP-based networks and IoT networks in the foreseeable future. To combat this issue, numerous researchers are developing approaches that utilize artificial intelligence to protect these systems.

Analyzing the vast daily traffic manually to halt malicious activities is highly challenging and inefficient for humans. The use of artificial intelligence can help analyze the daily traffic automatically [6,7]. It can also assist in protecting networks from novel types of attacks based on their characteristics [8]. Since its launch 18 years ago, Gmail has been utilizing machine learning techniques to filter emails [9]. Machine learning, particularly deep learning, is now employed across almost all of its services. Deep learning enables algorithms to adjust

* Corresponding author.
** Corresponding author.
E-mail addresses: furqan.rustam@ucdconnect.ie (F. Rustam), anca.jurcut@ucd.ie (A.D. Jurcut).

https://doi.org/10.1016/j.comnet.2024.110603
Received 23 November 2023; Received in revised form 3 May 2024; Accepted 11 June 2024
Available online 28 June 2024

and self-regulate during training and evolution as they learn [10]. This integration of artificial intelligence helps create cost-effective and accurate tools for companies to safeguard their networks. However, there remains a challenge as cybercriminals continuously discover new attack methods, necessitating constant improvement to these systems.

This study contributes to malicious traffic detection by proposing an approach that can help secure multi-environment systems based on IoT and traditional IP network traffic flow. We design a fully automated malicious traffic detection system (FAMTDS) approach using several methods such as feature engineering, machine learning, deep learning, and optimization. We generate a novel multi-environment traffic dataset using three publicly available datasets, IoTID-20, UNSW-NB15, and CICDDOS2019 [11–13]. We deploy several techniques to generate datasets, such as the feature selection technique, which determines which features should be taken from the original datasets to generate the new dataset. After feature selection, we generate new features to ensure equal feature set sizes for both datasets. We deploy several machine learning models with various hyperparameters. There are several concerning questions regarding the implementation of an effective approach, which are outlined below:

- What criteria should be used to select important features from the original dataset? Any threshold value for the importance score?
- How many features will be enough to make an accurate system for the multi-environment networks?
- What should be the hyper-parameters setting?

Solution: Previously, researchers have attempted to address these questions through their experience, knowledge, or trial-and-error methods, which often resulted in weaknesses in the approach. It is crucial to align these settings with the nature of the dataset and the model being used. Therefore, the question that arises for researchers is how to determine these settings effectively. To assist researchers, we propose an approach that utilizes a fully automated system where each step is automated using an optimizer. We employ the Moth Flame Optimizer (MFO) to optimize each step, thereby creating the optimal environment for the proposed FAMTDS. Our research offers several significant contributions, which are as follows:

- Propose an advanced, fully automated MTD approach that utilizes machine learning, deep learning, and optimization techniques for enhanced threat mitigation in multi-environment networks.
- Develop a unique, comprehensive dataset that integrates IoT and traditional IP network traffic, showcasing complex and diverse traffic samples that challenge current methods. Our goal is to create an approach that effectively manages multi-environment network complexities, enhancing research and enabling the development of robust, accurate detection techniques for varied traffic patterns.
- Implement highly optimized machine learning models according to the characteristics of the multi-environment dataset. These optimized models effectively capture and learn the complex patterns presented within the diverse multi-environment dataset. By leveraging advanced optimization techniques, these models exhibit superior performance and achieve significantly higher accuracy compared to existing state-of-the-art approaches.
- Propose a comprehensive and fully optimized system based on the MFO algorithm, designed to detect malicious traffic in both IoT networks and traditional IP-based networks. Leveraging the power of the MFO optimizer, the proposed system effectively addresses various challenges in the multi-environment setting. The MFO optimizer plays a crucial role in feature selection and feature augmentation processes, enabling the generation of a diverse and representative multi-environment dataset. Furthermore, it optimizes the machine learning models employed in the system, enabling them to effectively capture the intricate complexities embedded within the generated multi-environment dataset.
- Examine a collection of 19 attacks that, to our knowledge, have not been collectively studied or investigated in previous research. However, it is important to note that previous studies have addressed these attacks separately, focusing on either the UNSW-NB-15 dataset or the IoTID-20 dataset. In other words, they have not considered both datasets together when developing their approaches. By combining these two datasets, we aim to address the attacks in a unified manner within a multi-environment setting. Our study goes beyond the scope of previous works by considering the attacks from both datasets collectively. This allows us to capture the complexities and diversity that arise when multiple types of attacks are present in a real-world network environment.
- Employ a zero-day attack approach to validate the generalizability and significance of our proposed method. We conducted zero-day attack predictions in our multi-environment setup and introduced the latest attacks by incorporating the CIC-DDOS-2019 dataset into our multi-environment dataset.

This paper is organized into the following sections: Section 2 presents existing work on MTD and the motivation behind this study. Section 3 describes the flow of the proposed methodology, while Section 4 provides information about our newly generated multi-environment dataset. Section 5, contains machine learning algorithms description. Section 6 details the architecture of FAMTDS, and in Section 7, we discuss the results of our study and compare them with existing methods. Finally, Section 8 concludes the study and offers future directions for this work.

2. Literature review

In this technology era, network security has become very crucial because people share their data through networks, and conduct business transactions, and governments use it to share national secrets [14]. Everyone wants to be in a secure system where their privacy and confidentiality are prioritized. Therefore, strengthening the network security and investing a significant amount of money in buying protection protocols and systems is a very important aspect. However, security attackers continue to discover new ways every day to breach security protocols and take control of networks. Specifically, they attempt to find an easy way to enter the networks, such as through less secure devices. On the other hand, researchers are proposing new approaches to counter these new methods of breaching security systems, making them more secure and impregnable [15]. They are developing new security protocols to ensure unbreakable security in the smart world. In this section, we will discuss previous work in the network security domain and highlight the gap that our study aims to fill.

2.1. IoT-based datasets

The IoT devices are weaker in terms of security protocols, making it easy for hackers to breach their security systems and gain access to networks. To address this issue, the study [16] focused on developing an IoT intrusion detection system. They worked on feature selection and proposed a novel method for selecting high-performance features. They utilized the information gain and gain ratio techniques to extract meaningful features. The experiments were conducted using the IoTID-20 dataset, and several machine learning models, including kNN, C4.5, Bagging, and Ensemble, were deployed. Their proposed approach achieved an accuracy score of 0.9998 for the IoTID-20 dataset. In another study [17], an approach named SEHIDS (Self-Evolving Host-Based Intrusion Detection System) was proposed. This study focused on IoT network intrusion detection and employed deep learning algorithms for experiments using the BoT-IoT, TON-IoT, and IoTID-20 datasets. Recurrent neural networks (RNN) achieved significant accuracy scores of 1.00, 0.99, and 1.00 for all three used datasets.

The study [18] focused on developing an intrusion detection and prevention system using machine learning techniques. They conducted experiments using the Smart Home Dataset. Several machine learning models, including DT, KNN, RF, Adaboost, Bagging, and the voting model, were deployed. They utilized traffic statistics and header features to achieve favorable results. Adaboost performed well for both datasets, achieving accuracy scores of 0.838 for the Smart Home Dataset. Another study [19] conducted experiments on IoT intrusion detection using deep learning. A lightweight neural network (LNN) with Principal Component Analysis (PCA) was deployed for feature dimensionality reduction, resulting in significant results. The experiments were conducted on the BoT-IoT dataset. The LNN achieved accuracy scores of 0.9999 for the BoT-IoT dataset. Similarly, the study [20] proposed an approach for spam detection in IoT devices to enhance their security systems. Additionally, the study [21] proposed an approach for IoT intrusion detection using regularization techniques and a CNN-based approach. The experiments were performed using the Bot-IoT dataset. Information gain was employed to select important features, followed by the deployment of the CNN model to detect attacks. The approach was tested in three scenarios: DDoS attack, OS Fingerprint attack, and Service Scan attack. The results achieved for each scenario were 0.9998, 0.9849, and 0.9075, respectively. Furthermore, the study [22] proposed an approach for anomaly detection in the Internet of Railways (IoR). They utilized the IoR dataset for experiments. Attack detection was performed using Extended Neural Networks (ENN), CNN, LSTM, and DNN. The proposed approach achieved accuracy scores of 0.993 for the IoR dataset.

2.2. Traditional IP-based networks datasets

In this section, we will discuss several studies that have conducted work on general network malicious traffic detection. The study [23] focused on intrusion detection using bagging methods. They performed experiments on the NSL-KDD, UNSW-NB15, and HIKARI-202 datasets. Several bagging-based models, including GBM, LightGBM, CatBoost, XGBoost, and Bagging-GBM, were deployed for attack detection. Bagging-GBM outperformed all the datasets, achieving accuracy scores of 0.9466, 0.9157, and 0.8235 for UNSW-NB15, NSL-KDD, and HIKARI, respectively. Another study [24] conducted experiments on the UNSW-NB15 dataset, which is a traditional IP traffic dataset, and deployed the Bi-LSTM model. The proposed approach achieved an accuracy score of 0.9905 for the dataset used. Similarly, the study [25] performed experiments using the UNSW-NB15 dataset, which is a traditional IP-based network traffic dataset. They proposed a Multi-task Learning Model with Hybrid Deep Features based on CNN. The proposed approach achieved an accuracy score of 0.9811 for the dataset used. The proposed approach was also tested on other datasets, including UNSW-NB15.

The study [26] conducted experiments on intrusion detection using weighted features. They deployed the proposed approach on the UNSW-NB15 and TON_IoT datasets. The proposed approach utilized the Gini Impurity-based Weighted Random Forest technique for feature selection. DT with the proposed feature selection technique outperformed other models, achieving accuracy scores of 93.01 and 99.90 for the UNSW-NB15 and TON_IoT datasets, respectively. Similarly, another study [27] focused on denial of service attack detection. They performed experiments on the KDD-CUP 1999 and UNSW-NB15 datasets. Various feature selection techniques, such as information gain (IF) and Chi2, Merge IF+Chi2, XGB, DT, RF, and auto feature selection, were employed. Machine learning models, including SVM, LR, RF, NB, KNN, XGBoost, ANN, and CNN, were also used. RF achieved the highest accuracy score of 0.9988 for the UNSW-NB15 dataset.

2.3. Gap and motivation

The existing literature reviewed for this work shows that significant research efforts have been dedicated to network security in traditional IP-based and particularly IoT networks. Various publicly available datasets have been utilized to address security challenges in these networks. However, there is a noticeable gap in the literature that remains unexplored. This gap pertains to the lack of exploration and understanding of multi-environment traffic, which refers to networks that encompass both traditional IP-based networks and IoT networks, as shown in Fig. 2. The inclusion of IoT devices in a network introduces vulnerabilities as many of these devices have inadequate security measures. Common IoT devices, such as light bulbs, refrigerators, and thermostats, essentially function as small Linux computers. Unfortunately, most of these devices cannot be easily patched, and manual patching is often not feasible. Consequently, maintaining a flat network for all connected devices poses significant security risks [28]. There have been instances where an unnamed casino fell victim to a cyber attack due to a compromised thermostat, allowing hackers to gain control of the entire database [28,29].

To address the security challenges associated with multi-environment networks, which encompass both IoT and traditional IP-based networks, extensive research is required. In this study, we focus on Multi-Environment networks that encompass both traditional IP-based and IoT traffic flows. The inclusion of both types of traffic introduces a significant diversity in network traffic patterns, making it challenging for a state-of-the-art malicious traffic detection system to effectively handle such diversity and protect networks. Previous studies have mostly utilized machine learning techniques that are designed for individual network types, such as IoT or traditional IP-based networks. However, these systems fail to capture the diversity present in multi-environment networks. Machine learning models that perform well on one type of network may not generalize well to other datasets [32]. Few studies have focused on multi-environment networks, primarily addressing binary classification problems, such as the study [33] that introduced a malicious traffic detection system in multi-environment networks. They utilized advanced machine learning models, proposed stacked ensemble models, and applied PCA to combine IoT and traditional IP-based traffic data. Another study [34] tackled the issue of imbalanced datasets in multi-environment contexts, proposing a balanced dataset approach to minimize model overfitting, focusing on binary classification. Similarly, another research [34] introduced a Particle Swarm Optimizer-based method for addressing malicious traffic in multi-environment networks, employing ensemble prediction models to achieve significant outcomes. However, these studies mainly concentrated on binary classification and did not explore individual attack types. They conducted simple approach experiments without testing the generalizability of the proposed approach, such as testing for zero-day attacks. Unlike these, our approach with the CICDDOS2019 dataset addresses the latest IP-based network attacks, offering a more generalized and up-to-date solution. The summary of the discussed literature is presented in Table 1, providing an overview of the gaps, problems, and motivations identified within the existing research landscape.

3. Methodology

This study explores the implementation of a supervised machine learning approach for multi-environment Malicious Traffic Detection (MTD). Our proposed approach combines various techniques, including multi-environment dataset generation, feature selection, feature generation, and model optimization.

Fig. 1, illustrates the steps of our proposed approach. To begin, we obtained two datasets, namely IoTID-20 and UNSW-NB15, from open-source platforms. The IoTID-20 dataset represents IoT traffic, while the UNSW-NB15 dataset represents traditional IP traffic. By applying

Table 1
Summary of literature review.

Ref.	Year	Study Purpose	ML/DL	Dataset	Dataset size	Target	Classes	Feature Engineering	Algorithms
[16]	2022	IoT Intrusion Detection System	ML	IoTID20	IoTID20 = 625783	Binary & Multi	IoTID20 = Mirai, DoS, Scan, MAS, Normal	Information gain & Gain ratio	KNN, C4.5, Bagging and Ensemble
[17]	2022	Intrusion detection in internet of things	DL	BoT-IoT, TON-IoT, and IoTID20	Bot-IoT = 733,704,43 & TON_IoT = 22,339,021 & IoTID20 = 625783	Multi	–	–	RNN
[18]	2022	IoT intrusion detection and prevention system	ML	Smart Home Dataset	Smart Home Dataset	Multi	Smart Home Dataset = namely scanning, DoS, IoT-toolkit, and MITM	Packet header+traffic features	DT, KNN, RF, Adaboost, Bagging, and the voting
[19]	2022	IoT intrusion detection system	DL	BoT-IoT	Bot-IoT = 733,704,43	Binary	Normal & Attack	PCA	LNN
[21]	2022	IoT Intrusion Detection System	DL	Bot-IoT	Bot-IoT = 733,704,43	Binary	Normal & Attack	Information Gain	CNN
[22]	2022	Anomaly Detection in Internet of Railway	DL	IOR dataset	IOR dataset =-	Binary	Normal & Attack	Feature Rank	Extended NN, CNN, LSTM, and DNN
[23]	2023	Bagging approach for intrusion detection system	ML	NSL-KDD, UNSW-NB15, and HIKARI-2021	NSL-KDD = 148517 & UNSW-NB15 = 257673 & HIKARI = 444,223	Binary	Normal & Attack	–	GBM, LightGBM, CatBoost, XGBoost, Bagging-GBM
[24]	2023	Deep learning for intrusion detection system	DL	UNSW-NB15	UNSW-NB15 = 257673	Binary	Normal & Attack	–	Bi-LSTM
[25]	2022	Hybrid features for intrusion detection system	DL	Bot-IoT, UNSW-NB15, CICIDS2017, ISCX2012	Bot-IoT = 733,704,43, UNSW-NB15 = 2,540,044, CICIDS2017 = 2,830,696, ISCX2012 = 2,448,242	Multi	–	Statistical Features	Multi-task lEarning Model with hyBrid dEep featuRes (MEMBER)
[26]	2022	Weighted feature for intrusion detection system	ML & DL	UNSW-NB 15 & TON_IoT	UNSW-NB15 = 2540044, TON_IoT = 22,339,021	Binary	Normal & Attack	Gini Impurity-based Weighted Random Forest	DT, GBT, Adaboost, MLP, LSTM, GRU
[27]	2022	Feature selection and optimization for distributed denial of service attacks detection	ML & DL	KDD-CUP 1999 & UNSW-NB15	KDD CUP 1999 = 311029, UNSW-NB15 = 2540044	Binary	Normal & Attack	Information gain (IF) & Chi2, Merge IF+Chi2, XGB, DT, RF, Auto Feature Selection	SVM, LR, RF, NB, KNN, XGBoost, ANN, CNN
[30]	2022	To solve the non-deterministic feature selection problem	ML & DL	NSL-KDD & UNSW-NB15	NSL-KDD = 1,764,604 & UNSW-NB15 = 278,998	Multi	–	Fitness function, and Search space	RF, XGBoost, LightGBM, CatBoost, CNN, NN
[31]	2022	To categorize the network traffic into normal and attack classes	DL	NSL-KDD & UNSW-NB15	NSL-KDD = 148,517 & UNSW-NB15 = 257,673	binary	–	Integer encoding scheme	NN(Neural networks)
Our Study	**2023**	**MTD in multi-environment networks**	**ML & DL**	**UNSW-NB15 & IoTID-20**	**UNSW-NB15 = 1700 & IoTID-20 = 1530**	**Multi**	**Analysis, Backdoor, DoS, DoS-Synflooding, Exploits, Exploits, Generic, MITM ARP Spoofing, Mirai-Ackflooding, Mirai-HTTP Flooding, Mirai-Hostbruteforceg, Mirai-UDP Flooding, Normal-U, Normal-I, Reconnaissance, Scan Hostport, Scan Port OS, Shellcode, Worms**	**Feature Selection using ETC, Feature Generation using NN**	**RF, ETC, LR, SVM, LR, KNN, CNN, LSTM, RNN, NN**

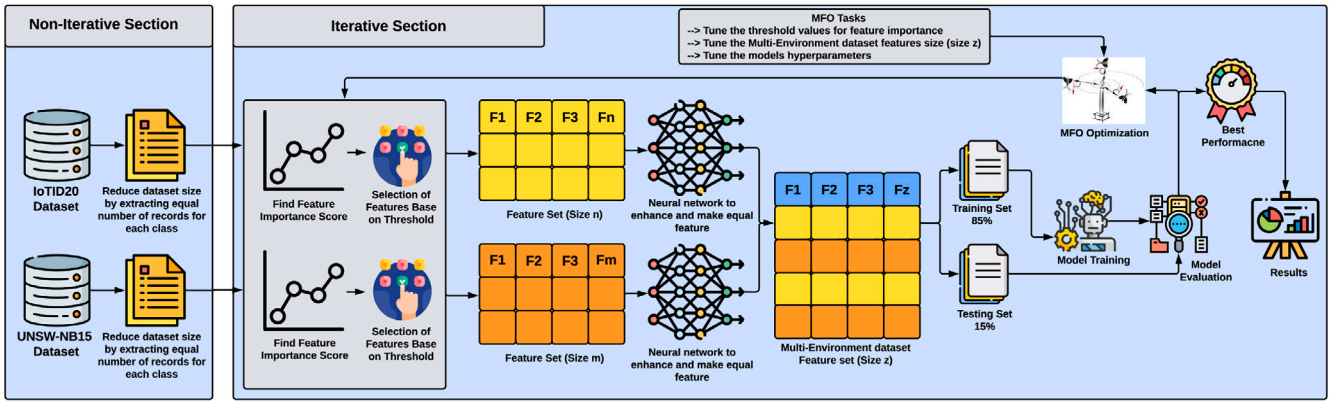


Fig. 1. FAMTDS architecture.

feature selection and feature generation techniques, we combined these datasets to create a multi-environment dataset. In the feature selection process, we utilized the extra trees classifier to determine the importance scores for each feature [35]. We then set a threshold value based on ETC extracted scores, where features above the threshold value were retained for further experimentation. For feature generation, we employed neural networks that take the features from both datasets and generate optimal feature sets of equal size.

During our experiments, we specifically targeted subcategories of attacks, comprising a total of 19 target classes. Notably, these subcategories of attacks had not been previously combined and considered collectively in the context of MTD studies. After generating the dataset, we split it into a training set (85%) and a testing set (15%). Multiple learning models, including deep learning and machine learning families, were utilized for MTD prediction. The models were trained using hyperparameters which were optimized based on the characteristics of the dataset.

To optimize all the aforementioned steps, we employed an MFO (Moth Flame optimizer) optimizer. The MFO facilitated the optimization of the number of selected features, the number of features to generate, and the hyperparameters of the models, all based on the nature of the dataset. By iterating the MFO optimizer, we achieved the best configuration for each step, resulting in a fully automated malicious traffic detection system (FAMTDS). The performance of the

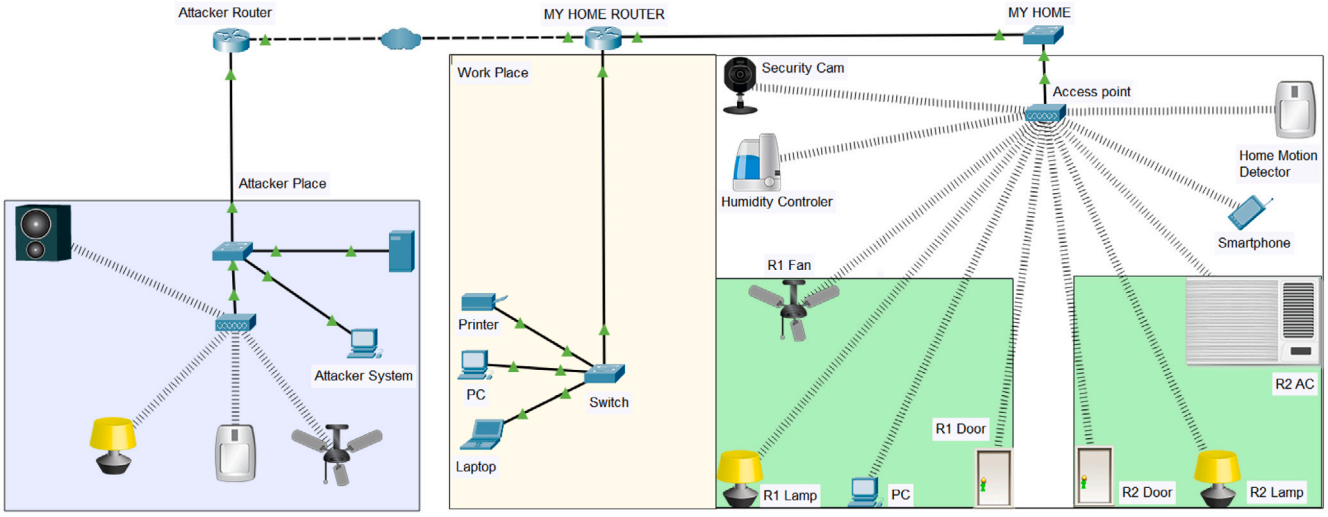


Fig. 2. Multi-environment networks implementation in CISCO.

Table 2
UNSW-NB15 dataset sample.

id	dur	proto	service	state	spkts	...	ct_srv_dst	is_sm_ips_ports	attack_cat
1	0.000011	udp	–	INT	2	...	2	0	Normal
323	1.155184	tcp	–	FIN	10	...	1	0	Fuzzers
58116	0.000004	udp	dns	INT	2	...	4	0	Generic
49000	0.000008	eigrp	–	INT	2	...	12	0	DoS

learning models was evaluated using accuracy, precision, recall, F1 score, and the confusion matrix. These metrics provided a comprehensive assessment of the effectiveness of our approach in detecting malicious traffic across multiple environments.

4. Dataset description

We use two datasets in this study to propose MTD systems for multi-environment systems. We acquired both datasets: UNSW-NB15 and IoTID-20 from open-source platforms. The UNSW-NB15 dataset is a traditional IP traffic dataset while the IoTID-20 dataset contains IoT traffic. Due to their extensive use in recent research, we chose these two datasets for our experiments.

4.1. UNSW-NB15 dataset

The UNSW-NB15 is collected by the Cyber Range Lab of the Australian Centre for Cyber Security [12]. The dataset contains normal traffic and synthetic contemporary attack traffic. The dataset contains 257,673 samples and consists of 45 features collected through traditional IP traffic. The dataset consists of 10 categories and 9 of them represent the attacks. A sample of the USNW-NB15 dataset is shown in Table 2.

4.2. IoTID-20

This dataset was collected in a typical smart home environment [11]. Several smart home devices such as EZVIZ Wi-Fi camera, SKT NGU, Tablets, Laptops, and Smartphones are used to generate the dataset. The IoTID-20 dataset consists of 625783 samples and 86 features. The dataset consists of nine categories and eight are attacks. Table 3 shows the samples from the IoTID-20 dataset.

4.3. Multi-environment dataset generation

In this study, we aim to contribute to network security in a scenario where both traditional IP-based and IoT traffic coexist within a single network. An ideal multi-environment traffic scenario is a smart home, where IoT devices like smart cameras, smart fans, smart doors, etc., coexist with traditional IP-based network devices such as printers and computers. Both IoT and traditional IP-based networks are interconnected within the same home network, as depicted in Fig. 2. To validate the feasibility of our proposed scenario, we implemented it in Cisco Packet Tracer. This implementation allows us to ensure that our envisioned setup can be practically realized.

To implement the multi-environment MTD approach, we encountered a challenge as there is no publicly available dataset specifically designed for this purpose. As a solution, we decided to generate our dataset. To create this dataset, we utilized two distinct traffic datasets: IoTID-20 and UNSW-NB15. By merging these datasets, we were able to construct a multi-environment dataset that includes samples of both traditional IP-based and IoT traffic. Merging datasets is a complex task since both datasets have different numbers of features.

Both datasets are different in size and contain different numbers of classes. In this study, we focus on 19 target attack samples within the multi-environment dataset. The specific target attack class "Worms" has only 174 samples, which is relatively low compared to other classes in the dataset. To ensure equal representation and balanced training for all attack classes, we extract 170 samples from each of the other target classes as shown in Table 4. This equal sampling approach aims to address the diversity within the dataset and to allow the proposed approach to focus on each target class effectively.

Before combining the datasets, preprocessing is necessary due to the presence of different features in each dataset. To handle this, we employ the extra trees classifier (ETC) method to select important features from each dataset. However, to effectively combine the datasets, it is crucial to have an equal number of features. To address this issue, we generate an equal number of features by utilizing only the ETC-selected important features. These features are passed through a neural network (NN) for further processing. Once an equal number of features is generated, we proceed to combine the datasets. The multi-environment dataset exhibits a significant degree of diversity due to the inclusion of traffic from two different types of networks. To capture the

Table 3
Sample from IoTID-20 dataset.

Flow_ID	Src_IP	Src_Port	Dst_IP	Idle_Max	Idle_Min	...	Sub_Cat
192.168.0.13-192.168.0.16-10000-10101-17	192.168.0.13	10 000	192.168.0.16	75	75	...	Mirai-Ackflooding
192.168.0.24-58.225.75.83-41467-443-6	192.168.0.24	41 467	58.225.75.83	443	6	...	Scan Port OS
192.168.0.13-192.168.0.16-9020-49784-6	192.168.0.13	9020	192.168.0.16	49 784	6	...	Normal
192.168.0.13-192.168.0.16-9020-53190-6	192.168.0.16	53 190	192.168.0.13	9020	6	...	MITM ARP Spoofing

Table 4
Dataset count before and after under-sampling.

UNSW-NB15			IoTID-20		
Attack	Original Dataset	Under-Sampling	Attack	Original Dataset	Under-Sampling
'Normal-U'	93 000	170	'Mirai-Ackflooding'	55 124	170
'Backdoor'	2329	170	'DoS-Synflooding'	59 391	170
'Analysis'	2677	170	'Scan Port OS'	53 073	170
'Fuzzers'	24 246	170	'Mirai-Hostbruteforceg'	121 181	170
'Shellcode'	1511	170	'Mirai-UDP Flooding'	183 554	170
'Reconnaissance'	13 987	170	'Mirai-HTTP Flooding'	55 818	170
'Exploits'	44 525	170	'Normal-I'	40 073	170
'DoS'	16 353	170	'Scan Hostport'	22 192	170
'Worms'	174	170	'MITM ARP Spoofing'	35 377	170
'Generic'	58 871	170	–		

complex patterns present in the dataset, we deploy machine learning models with optimized settings.

4.4. ETC for feature selection

The network datasets often contain a large number of features, including both relevant and irrelevant ones to the target class. ETC's ability to handle noisy data and effectively filter out irrelevant features makes it a suitable choice for feature selection [36]. Additionally, ETC's capability to capture non-linear patterns and interactions enables it to identify important features that significantly contribute to the overall network traffic behavior. By utilizing ETC for feature selection, we ensure that only the most informative features are retained which helps to reduce noise and enhance the quality of the feature set used in subsequent analysis.

ETC is a tree-based ensemble model that combines multiple decision trees under the bagging method. ETC used majority voting criteria to make the prediction. Each tree gets fitted with random k features and selects the best features to split the node based on mathematics such as Information Gain or Gini Index. We used information gain, which can be calculated; first entropy is calculated using Eq. (1) and the gain using Eq. (2).

$$Entropy(S) = \sum_{i=1}^{c} -p_i log_2(p_i) \tag{1}$$

$$Gain(S, A) = Entropy(S) - \sum_{v \ \epsilon \ values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \tag{2}$$

Here, p_i is the sample of the dataset with label i, c is the number of unique target classes, and A is a feature/attribute [37].

$$etc_{model} = mode\{t_1, t_2, \ldots, t_{300}\} \tag{3}$$

$$etc_{model} = mode\{\sum_{i=1}^{3} 00t_i\} \tag{4}$$

Eqs. (3) and (4) show the ETC model mathematics. Here, $t_1, t_2, \ldots, t_{300}$ are decision trees and we use 300 trees under ETC. We pass all features to the ETC with the target and find the feature's importance.

$$f_{imp}\_Score = etc_{model}(features, target) \tag{5}$$

First, we pass the original datasets to the ETC model and find the importance score for each feature. We set threshold values for importance scores to select the important features to use further in experiments. This threshold value varies from data to data and we find this threshold using an optimizer.

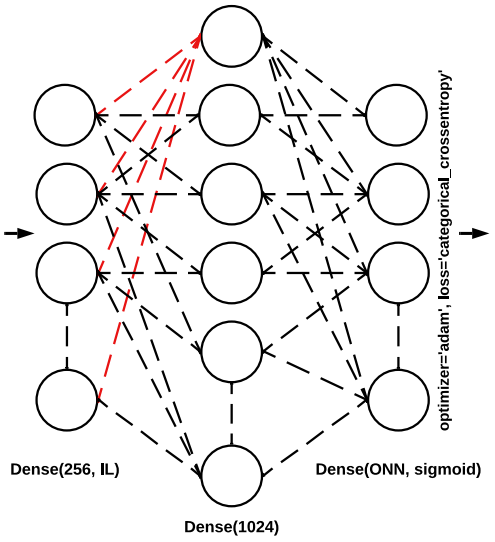


Fig. 3. Neural networks architecture to generate the features.

4.5. NN for feature generation

Recently lots of researchers used convolutional or other neural networks to generate the features [38,39]. We use a similar kind of idea in our approach after selecting important features. We make both datasets equal in terms of feature set size and for that, we use NN to enhance and generate an equal number of features. However, determining the specific number of features to be generated is a question that needs to be addressed. To tackle this, we introduce an optimizer that dynamically determines the feature set size based on the nature of the model. The architecture of the neural network used for feature generation is illustrated in Fig. 3.

The initial network layer comprises 256 neurons, while 'IL' denotes the number of features or input length. The specific number of input features may differ across datasets, and our approach involves the optimizer determining the value of 'IL' by identifying the significant features in the early phase of feature selection. The second layer comprises 1024 neurons, while the last layer consists of optimized numbers of neurons (ONN) and the sigmoid function [40]. By leveraging this flexible approach, the optimizer effectively determines the appropriate feature set size, enhancing the adaptability and performance of our

approach. After extracting the optimized number of features (ONF) from both datasets we combine them vertically and make a multi-environment dataset that contains both IoT and traditional IP-based network traffic samples. Finally, We shuffle the dataset to mix all the samples.

Algorithm 1 shows the approach to generate the multi-environment data. Here, I represents the IoTID-20 data and U represents UNSW-NB15 data. FI_{score}, FG_{score} are feature importance score extracted using ETC. SF_I and SF_U are the selected feature using score threshold for both IoTID-20 and UNSW-NB15 dataset respectively. Then, GF_I, GF_U are generated feature sets using NN for both IoTID-20 and UNSW-NB15 datasets respectively. In the end, we combine both GF_I, GF_U to generate the multi-environment dataset.

Algorithm 1 Algorithm for Multi-environment Data Generation

Input
IoT_t = IoTID-20 traffic samples
U_t = UNSW-NB15 traffic samples
etc_{model} = ETC models for feature importance.
NN_{model} = NN model for feature generation.
Output: Multi-environment Dataset

1: **procedure:**
2: **def** Multi-Environment(I,U):
3: $FI_{score} = etc_{model}(I)$
4: $FG_{score} = etc_{model}(U)$
5: $SF_I, SF_U = \{FI_{score} \& FG_{score}\} > t$, where $t \epsilon \{FI_{score}, FG_{score}\}$
6: $GF_I, GF_U = NN_{model}(ONF)\{SF_I, SF_U\}$
7: multi-environment data= merge(GF_I, GF_U, axis=0)
8: **return (multi-environment data)**
9: **Call:**Multi-Environment(IoT_t, U_t)

Fig. 4 shows the architecture utilized for generating a multi-environment dataset. In this instance, we present a specific scenario with actual values for the number of features after feature selection and generation, which are determined by the optimizer. Let us consider two datasets, one with 83 features and the other with 43 features. After applying the feature selection process, we obtained 30 and 23 features, respectively. Attempting to combine the datasets at this stage would result in an inaccurate dataset and is not an optimal approach, primarily due to the disparate number of features present in each dataset, as illustrated in Fig. 4. Consequently, we refrain from combining the datasets immediately after feature selection and instead pass the features through an NN to generate an equal number of features. The NN, in this scenario, generates 2000 features for each dataset, facilitating an accurate and precise combination of the datasets.

5. Machine learning models

This study uses various machine learning models for MTD, including two tree-based ensemble models RF [41], ETC [42], two linear models LR [43], SVM [44], and a simple distance based model KNN [45]. These models contain numerous parameters with wide value ranges, making it challenging to determine the optimal hyperparameters for the given dataset. It is crucial to set the models' parameters based on the specific nature of the dataset. Researchers often face the question of determining the appropriate parameter values that suit the dataset's characteristics. Conducting hyperparameter tuning through trial and error methods can be time-consuming and may not yield the optimal combination.

To address this issue, we employ the Moth-Flam Optimizer (MFO) to assist in finding the optimal solution based on the dataset's nature. We pass a range of parameters and their corresponding values to the optimizer, allowing it to fine-tune these parameters considering the number of features and the characteristics of the feature set. The range for the hyperparameters of the used machine learning models is presented in Table 5.

Table 5
Machine learning hyper-parameters settings.

Model	Hyper-parameters Range
RF	n_estimators = {2 to 300}, max_depth = {2 to 50}, random_state = {2 to 50}
ETC	n_estimators = {2 to 300}, max_depth = {2 to 50}, random_state = {2 to 50}
LR	solver = ['saga','sag','newton-cg', 'lbfgs'], random_state = {2 to 1000}, multi_class = 'multinomial', C = {2 to 10 }
SVM	kernel = ['linear', 'poly', 'rbf', 'sigmoid'], random_state = {2 to 1000}, multi_class = 'multinomial', C = {2 to 10 }, C = {2 to 10 }
KNN	Algorithm = ['auto', 'ball_tree', 'kd_tree', 'brute'], n_neighbors = {2 to 10}

RF, and ETC are both tree-based ensemble models that employ decision trees as weak learners and combine them using a majority voting criterion. In the majority voting criterion, each decision tree contributes its predicted class, and the class with the highest number of predictions becomes the final prediction. The only distinction between RF, and ETC lies in their training approach. RF trains each decision tree on a sub-sample from the original dataset of the same size, while ETC trains all decision trees on the complete dataset. Both models share common hyperparameter settings, such as n_estimators, which determines the number of decision trees for voting, max_depth, which controls the maximum depth of each decision tree to prevent overfitting, and random_state, which introduces randomness into the dataset.

LR and SVM are linear models and both require a large feature set for good performance. LR uses a sigmoid function to categorize target classes, and SVM uses a hyperplane to classify data. SVM places multiple hyper-planes in feature space and selects the best one that has a good margin for the classification. LR is used with solver, which defines optimization algorithms, random_state, and C which regularizes the learning rate. SVM is used with the kernel, which helps to improve the accuracy of SVM and C, which regularizes the learning rate.

KNN is the simplest model, employing a distance matrix to identify the nearest neighboring point. It is often referred to as a "lazy learner" because it matches training samples with the testing sample and predicts the testing sample based on the best match among the target values of the training samples. In our study, we use the Algorithm parameter, which aids in computing the nearest neighbors, and n_neighbors, which determines the number of neighbors considered for distance computation.

Algorithm 2 shows the approach for the model evaluation. Here, $M - E_{traffic}$ is multi-environment traffic data, $model$ is one of the used models, $Trained_m$ is the trained model, and $Model_{pred}$ is the model prediction on test data.

6. FAMTDS

FAMTDS is a combination of several techniques, such as feature selection, feature generation, machine learning models, and optimization algorithms. We generate multi-environment traffic using feature selection and feature generation and use machine learning models for malicious traffic prediction and MFO optimizer to make the whole system fully automated. MFO addresses several key questions, including determining the appropriate threshold value for selecting important features, identifying the optimal number of features required to achieve a good accuracy score, selecting the suitable hyperparameter settings, and choosing the appropriate parameter values based on the characteristics of the feature set and dataset.

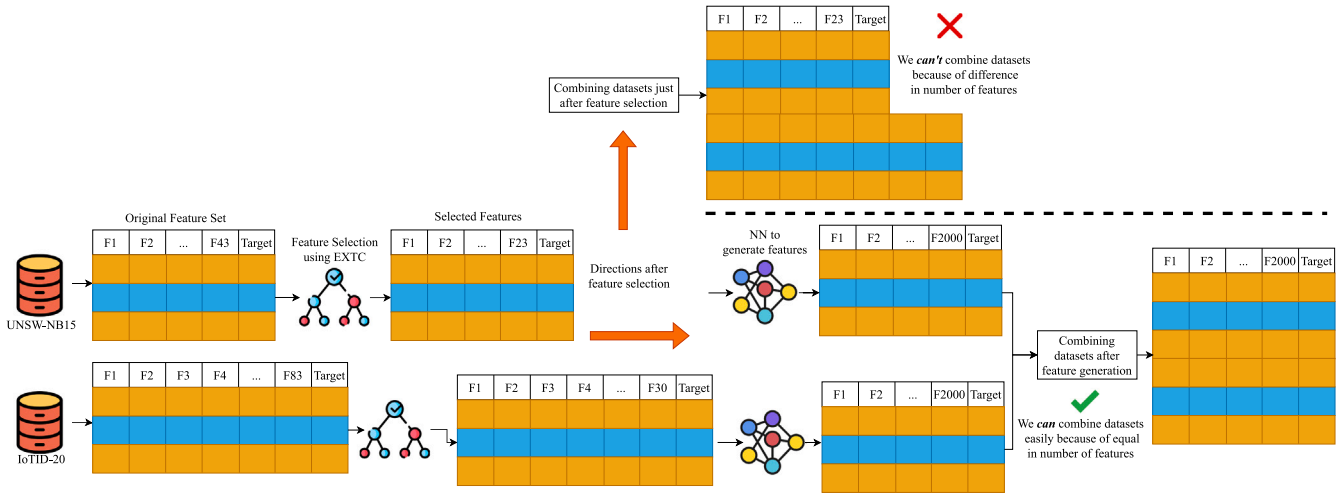


Fig. 4. Proposed architecture to generate the multi-environment dataset.

Algorithm 2 Algorithms of Machine Learning Model Optimization

Input
$M - E_{traffic}$ = Multi-Environment(IoT_t, U_t)
$model$ = {RF, ETC, SVM, LR, KNN}
Output: Attack Prediction Accuracy

procedure:
def Model-Evaluation(D,M):
    Training-set, Testing-set = split(D, test-size=0.15)
    $Trained_m$ = M.fit(Training-set)
    $Model_{pred} = Trained_m.(Testing - set)$
    score= evaluation($Model_{pred}$)
    return (score)
Call: Model-Evaluation($M - E_{traffic}$, $model$)

6.1. MFO

Moth Flam Optimizer (MFO) is one of the meta-heuristic algorithms that is successfully deployed in several problems such as energy and power, engineering, medical application, imaging, and economics [46]. It was developed by the Mirjalili inspired by the moths' movement in the night [47]. The population-based algorithm is combined with the local search strategy to improve the global and local search. MFO is a flexible, simple, and easy-to-implement metaheuristic, similar to other metaheuristic algorithms. The MFO algorithm consists of three fundamental steps: (a) establishing an initial population of moths, (b) updating the moths' positions, and (c) updating the number of flames.

Step 1: In the first step, MFO creates the initial population of moths, which are capable of flying in various dimensional spaces (1-D, 2-D, 3-D) [46,47]. The population of moths can be represented as follows:

$$Mo = \begin{bmatrix} mo_{1,1} & mo_{1,2} & \cdots & mo_{1,d} \\ mo_{1,1} & mo_{1,2} & \cdots & mo_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ mo_{n,1} & mo_{n,2} & \cdots & mo_{n,d} \end{bmatrix}_{n\ x\ d} \tag{6}$$

Here, n and d represent the number of moths and the number of dimensions, respectively [46,47]. The fitness values are also stored in an array, which can be expressed as follows:

$$FvM = \begin{bmatrix} fvm_1 \\ fvm_2 \\ \vdots \\ fvn_n \end{bmatrix} \tag{7}$$

The FvM represents the fitness values for all moths. The remaining elements of the MFO algorithm are the flames Fl in dimension d, which can be represented in the matrix shown in Eq. (8). Additionally, the fitness values FvF for the flames are represented in the matrix shown in Eq. (9).

$$Fl = \begin{bmatrix} fl_{1,1} & fl_{1,2} & \cdots & fl_{1,d} \\ fl_{1,1} & fl_{1,2} & \cdots & fl_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ fl_{n,1} & fl_{n,2} & \cdots & fl_{n,d} \end{bmatrix}_{n\ x\ d} \tag{8}$$

$$FvF = \begin{bmatrix} fvf_1 \\ fvf_2 \\ \vdots \\ fvf_n \end{bmatrix} \tag{9}$$

Moths and flames serve as solutions in the MFO algorithm, but with a distinction: moths are the searching agents, whereas flames represent the best solution or position achieved by the moths thus far. Both moths and flames are updated after each iteration.

Step 2: The second step aims to converge towards the global optimum of the optimization problem by utilizing three functions. Function (L) involves determining the random location of the moths, function (M) involves the movement of the moths within the search space, and function (F) signifies the completion of the search process.

$$MFO = (L, M, F) \tag{10}$$

Eq. (11) shows the function L which is the random locations of the moth's, where ub is the upper bound and lb lower bound [46].

$$Mo(s, v) = ((ub(s) - lb(v)) * rand() - lb(s)) \tag{11}$$

Therefore, Eq. (12) defines the MFO algorithm spiral, where, Dm_s is the space between sth and vth moths.

$$S(Mo_s, Fl_v) = Dm_s.e^{bt}.cos(2\pi t) + Fl_v \tag{12}$$

Step 3: In the third step of the MFO algorithm, the number of flames is updated. When the moths' locations are updated across different positions, the search space expands. Reducing the number of flames can aid in finding the best solution. The criteria for determining the number of flames (NoF) is expressed in Eq. (13), where MnF represents the maximum number of flames, CnI represents the current number of iterations, and MnI represents the maximum number of iterations.

$$NoF = round(MnF - CnI * \frac{MnF - CnI}{MnI}) \tag{13}$$

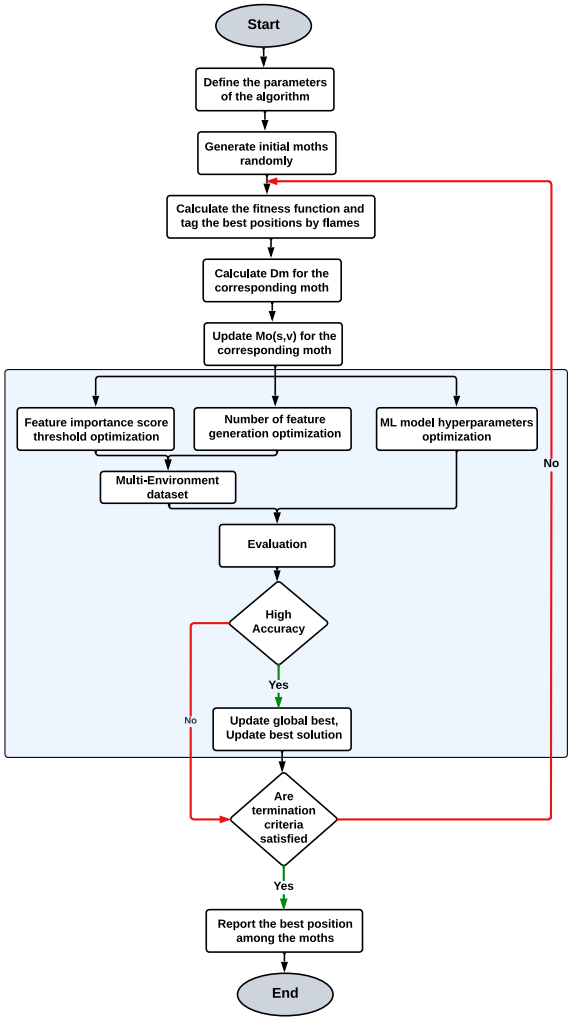


Fig. 5. FAMTDS Flowchart diagram.

6.2. Implementation of MFO for FAMTDS

We implement MFO for generating a multi-environment dataset (feature selection, feature generation) and model optimization according to the dataset's nature. The MFO combination with other used approaches helps to create a fully automated malicious detection system.

Fig. 5 shows the flow diagram for our proposed FAMTDS. We updated the MFO flow diagram by adding our approach. MFO will get the problem statement and start searching for the best solution for the feature importance score threshold value, the number of features generated, and the optimized hyper-parameters in correlation with the generated multi-environment dataset. We evaluate the score after deploying the learning models on a multi-environment dataset. If the score is highest as compared to the previous iteration we add it to the best solution, and if it is not good as compared to the previous iteration we will not update the best solution and start the next iteration. We used MFO with several parameters as shown in Table 6.

Table 7 shows the lower bound (LB) and upper bound (UB) values for each optimized step. We pass hyper-parameters of machine learning models to be selected according to importance score threshold value (Imp. Scr. Thrs) and the number of features (Num. of Feat.) to be generated before merging both IoTID-20 and UNSW-NB15 datasets to generate the multi-environment dataset. We set these values after analyzing the dataset's nature and literature knowledge. MFO first finds the feature importance score threshold value using ETC for both IoTID-20 and UNSW-NB15 datasets and selects only important features for processing. Then, it finds the best number to generate the feature set using NN and generates equal numbers of features for both IoTID-20 and UNSW-NB15 datasets and merges them horizontally to generate a multi-environment dataset. Finally, it finds the best values for learning models according to the generated dataset for each model. Algorithm 3 shows our proposed FAMTDS approach.

Table 6
MFO parameters and their values.

Parameter	Values
Problem	{"obj_func": fitness_function, "lb": [LB values], "ub": [UB values], "minmax": "max", "log_to": None, "save_population": False,}
Epoch	10
Pop_size	50
Pr	0.03

7. Results & discussion

The experiments in this work were conducted using 11th generation Core i7 with Windows operating system and NIDIA GPU. The system contains 16 GB RAM and 1 TB SSD. Furthermore, we implemented the approach in Python language on Jupyter Notebook. Several libraries such as Mealpy, Sci-kit Learn Tensorflow, and Keras were used to implement our approach.

7.1. Results on original dataset without under-sampling using machine learning models

This section assesses the performance of machine learning models using the original datasets without employing feature selection and data sampling. Notably, our FAMTDS was not deployed; instead, machine learning models were directly applied to the dataset. The RF model demonstrates commendable performance using the IoID-20 dataset, showcasing a strong accuracy score of 0.75 alongside an F1 score matching this accuracy level as shown in Table 8. However, LR exhibits indications of overfitting, as evidenced by its lower accuracy of 0.68 and the lowest F1 score of 0.53 among the models tested. Similarly, models on UNSW-NB15 also perform poorly with the original dataset. RF achieves 0.85 accuracy but a 0.60 F1 score, showing that the model is overly fitted to the majority sample, while linear models perform even worse. Overall, we can observe the impact of model overfitting due to imbalanced datasets. To address this issue, we implemented under-sampling, which effectively reduces the dataset size while aiding in mitigating model overfitting.

Table 9 presents the results of machine learning models for both the IoTID-20 and UNSW-NB15 datasets after under-sampling. The machine learning models were applied directly to the original dataset but underwent under-sampling for analysis. Table 9 displays the individual models' performance, showing RF performing remarkably well with an accuracy score of 0.77 for the IoTID-20 dataset and 0.83 for the UNSW-NB15 dataset. The models consistently exhibit significant performance improvements across all evaluation metrics, notably achieving substantial F1 scores compared to results obtained from imbalanced data. Under-sampling not only enhances accuracy and F1 scores but also reduces computational costs, prompting us to further experiment solely with under-sampled data and the deployment of FAMTDS on it.

Table 7
LB and UB values used by MFO for optimization.

Model	Values Range (LB to UB)			FMTMD		
	Hyper-parameters	Imp. Scr. Thrs	Num. of Feat.	Threshold	Features	Hyper-parameters
RF	n_estimators = {2 to 300}, max_depth = {2 to 50}, random_state = {2 to 50}	{0.01 to 0.03}	{90 to 200}	0.0100	2000	n_estimators = 113, max_depth = 12, random_state = 2
ETC	n_estimators = {2 to 300}, max_depth = {2 to 50}, random_state = {2 to 50}	{0.01 to 0.03}	{90 to 200}	0.0204	1156	n_estimators = 244, max_depth = 17, random_state = 2
LR	solver = ['saga','sag','newton-cg', 'lbfgs'], random_state = {2 to 1000}, multi_class = 'multinomial', C = {2 to 10 }	{0.01 to 0.03}	{90 to 200}	0.0262	1992	random_state = 10, solver = 'newton-cg', multi_class = 'multinomial', C = 7.0
SVM	Kernel = ['linear', 'poly', 'rbf', 'sigmoid'], random_state = {2 to 1000}, C = {2 to 10 }	{0.01 to 0.03}	{90 to 200}	0.0193	807	Kernel = 'linear', random_state = 2, C = 10.0
KNN	Algorithm = ['auto', 'ball_tree', 'kd_tree', 'brute'], n_neighbors = {2 to 10}	{0.01 to 0.03}	{90 to 200}	0.0156	1716	n_neighbors = 3, Algorithm = 'auto'

Algorithm 3 Algorithms for proposed FAMTDS

Input
IoT_t = IoTID-20 traffic samples
U_t = UNSW-NB15 traffic samples
etc_{model} = ETC models for feature importance.
NN_{model} = NN model for feature generation.
Output: Multi-environment Dataset

procedure:
def FAMTDS(I,U,M,H,T,N):
    FI_{score}= $etc_{model}(I)$
    FG_{score} = $etc_{model}(U)$
    $SF_I,\ SF_U$ = $\{FI_{score} \& FG_{score}\} > T$, where $T \epsilon \{FI_{score},\ FG_{score}\}$
    $GF_I,\ GF_U$ = $NN_{model}(N)\{SF_I,\ SF_U\}$ $N = Numberof Features(length)$
    multi-data= merge(GF_I, GF_U, axis=0)
    Training-set, Testing-set = split(multi-data, test-size=0.15)
    $Optimized_(M)$= M(H)
    $Trained_m = Optimized_(M)$.fit(Training-set)
    $Model_{pred} = Trained_m.(Testing - set)$
    score= evaluation($Model_{pred}$)
    return (score)
def MFO(P,E, PS, PR):
    Op_H = Find optimized hyper-parameters using LB to UB values
    Op_T = Find optimized threshold using LB to UB values
    Op_NoF =Find optimized number of features using LB to UB values
    Best_Solution=**Call:** FAMTDS(IoT_t, U_t, Model, Op_H, Op_T, Op_NoF)
    return (Best_Solution)
Call: MFO(problem,epoch=10, pop_size=50, pr=0.03)

Table 8
Machine learning models results using original features and without under-sampling of datasets.

Dataset	Model	Accuracy	Precision	Recall	F1 Score
IoTID-20	RF	0.75	0.75	0.75	0.75
	ETC	0.74	0.76	0.75	0.75
	LR	0.68	0.58	0.54	0.53
	KNN	0.75	0.78	0.72	0.70
	SVM	0.69	0.60	0.56	0.54
UNSW-NB15	RF	0.86	0.63	0.59	0.60
	ETC	0.85	0.61	0.56	0.57
	LR	0.61	0.37	0.26	0.25
	KNN	0.75	0.44	0.39	0.40
	SVM	0.65	0.45	0.36	0.35

Table 9
Machine learning model results using original features and under-sampling for individual datasets.

Dataset	Model	Accuracy	Precision	Recall	F1 Score
IoTID-20	RF	0.77	0.78	0.77	0.77
	ETC	0.73	0.73	0.73	0.73
	LR	0.61	0.63	0.61	0.61
	KNN	0.72	0.74	0.72	0.72
	SVM	0.65	0.67	0.65	0.65
UNSW-NB15	RF	0.83	0.84	0.83	0.83
	ETC	0.82	0.82	0.82	0.82
	LR	0.67	0.66	0.67	0.66
	KNN	0.78	0.79	0.78	0.78
	SVM	0.80	0.82	0.80	0.80

Table 10
RF and ETC results using FAMTDS for Multi-environment dataset.

RF				ETC			
Class	Precision	Recall	F1 Score	Class	Precision	Recall	F1 Score
Analysis	0.75	0.46	0.57	Analysis	0.67	0.46	0.55
Backdoor	0.80	0.92	0.86	Backdoor	0.77	0.88	0.82
DoS	0.79	0.92	0.85	DoS	0.79	0.88	0.83
DoS-Synflooding	1.00	1.00	1.00	DoS-Synflooding	1.00	1.00	1.00
Exploits	0.77	0.96	0.86	Exploits	0.79	0.88	0.83
Fuzzers	1.00	1.00	1.00	Fuzzers	1.00	1.00	1.00
Generic	0.96	1.00	0.98	Generic	1.00	1.00	1.00
MITM ARP Spoofing	0.92	0.96	0.94	MITM ARP Spoofing	0.96	1.00	0.98
Mirai-Ackflooding	0.46	0.42	0.44	Mirai-Ackflooding	0.52	0.46	0.49
Mirai-HTTP Flooding	0.47	0.65	0.55	Mirai-HTTP Flooding	0.41	0.54	0.47
Mirai-Hostbruteforceg	0.96	0.92	0.94	Mirai-Hostbruteforceg	0.96	0.92	0.94
Mirai-UDP Flooding	0.89	0.62	0.73	Mirai-UDP Flooding	0.90	0.69	0.78
Normal-U	1.00	1.00	1.00	Normal-U	1.00	0.96	0.98
Normal-I	0.93	1.00	0.96	Normal-I	0.90	1.00	0.95
Reconnaissance	0.95	0.84	0.89	Reconnaissance	0.84	0.84	0.84
Scan Hostport	0.81	0.88	0.85	Scan Hostport	0.85	0.88	0.86
Scan Port OS	0.86	0.72	0.78	Scan Port OS	0.86	0.76	0.81
Shellcode	0.96	0.88	0.92	Shellcode	1.00	0.88	0.94
Worms	0.92	0.92	0.92	Worms	0.89	0.96	0.93
Macro avg	0.85	0.85	0.84	Macro avg	0.85	0.84	0.84
Weighted avg	0.85	0.85	0.84	Weighted avg	0.85	0.84	0.84
Avg. accuracy	0.85			Avg. accuracy	0.84		

Table 11
RF and ETC confusion metrics using FAMTDS for multi-environment dataset.

RF

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
A	**12**	6	5	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	2	**24**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	1	0	**23**	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	**25**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	**24**	0	0	0	0	0	0	0	0	1	0	0	0	0	0
F	0	0	0	0	0	**26**	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	**25**	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	**24**	1	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	1	**11**	14	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	7	**17**	0	2	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	1	0	0	**23**	0	0	1	0	0	0	0	0
L	0	0	0	0	0	0	0	0	5	5	0	**16**	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	**26**	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	**26**	0	0	0	0	0
O	1	0	0	0	2	0	0	0	0	0	0	0	0	0	**21**	0	0	1	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**22**	3	0	0
Q	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	5	**18**	0	0
R	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	**23**	2
S	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	**24**

ETC

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
A	12	6	4	0	2	0	0	0	0	0	0	0	0	0	2	0	0	0	0
B	3	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	22	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1
D	0	0	0	25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	1	0	1	0	22	0	0	0	0	0	0	0	0	0	1	0	0	0	0
F	0	0	0	0	0	26	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	25	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	25	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	12	14	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	9	14	0	2	0	1	0	0	0	0	0
K	0	0	0	0	0	0	0	1	0	0	23	0	0	1	0	0	0	0	0
L	0	0	0	0	0	0	0	0	2	6	0	18	0	0	0	0	0	0	0
M	0	1	0	0	0	0	0	0	0	0	0	0	25	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	26	0	0	0	0	0
O	2	0	0	0	2	0	0	0	0	0	0	0	0	0	21	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	3	0	0
Q	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	4	19	0	0
R	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	23	2
S	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	25

7.2. Results of proposed approach for multi-environment data

This section presents the results obtained with several machine learning models when using our proposed FAMTDS. In our approach, we optimized each step using an MFO optimizer, including feature selection, feature generation, and hyperparameter tuning. The results are presented in terms of accuracy, precision, recall, F1 score, and confusion matrices.

Table 10 presents the results of FAMTDS for multi-environment traffic using RF and ETC. Both models are tree-based ensemble models that utilize decision tree predictions for voting and combine them based on majority voting criteria. RF achieved the highest accuracy score of 0.85 among all models in the study. However, it performed relatively poorly in the Mirai-Ackflooding, Mirai-HTTP Flooding, and Analysis categories, achieving F1 scores of only 0.44, 0.55, and 0.57, respectively. On the other hand, ETC closely followed RF with an accuracy score of 0.84. Similar to RF, ETC also showed poor performance in the same target classes such as Mirai-Ackflooding, Mirai-HTTP Flooding, and Analysis. The notable success of RF and ETC can be attributed to their ensemble architecture, which enables them to perform well on small datasets, such as the one used in our study. Additionally, these models are effective in mitigating overfitting issues due to the presence of multiple decision trees in the prediction process and the utilization of the bagging method during training.

Table 11 shows the confusion matrices for both RF and ETC models. RF gives 410 correct predictions out of 485 predictions and gives 75 wrong predictions. As we discussed above RF could not perform well for Mirai-Ackflooding, Mirai-HTTP Flooding, and Analysis classes prediction, and in terms of the confusion matrix, RF gives 14 wrong predictions for Mirai-Ackflooding out of 26 predictions. Similarly for Mirai-HTTP Flooding and Analysis classes, it gives 15 and 9 wrong predictions prediction respectively. ETC is just behind the RF with a total of 408 correct predictions and 77 wrong predictions. Overall tree-based ensemble models RF and ETC are good in terms of accuracy on a small dataset for multi-label classification, while RF is more significant for MTD. In all confusion matrics, we represent the attacks with a specific character such as A: Analysis, B: Backdoor, C: DoS, D: DoS-Synflooding, E: Exploits, F: Exploits, G: Generic, H: MITM ARP Spoofing, I: Mirai-Ackflooding, J: Mirai-HTTP Flooding, K: Mirai-Hostbruteforceg, L: Mirai-UDP Flooding, M: Normal-U, N: Normal-I, O: Reconnaissance, P: Scan Hostport, Q: Scan Port OS, R: Shellcode, R: Worms.

LR and SVM are two linear models that require a large feature set to achieve significant results and this is difficult since it requires knowing the appropriate size for the models. To resolve this problem our FAMTDS approach helps a lot as it finds the best feature length by doing optimization. MFO finds the best-optimized feature set size for LR and SVM. Hence LR and SVM can achieve significant results as well as the tree-based models. LR and SVM both give 0.84 accuracy scores as shown in Table 12. However, LR and SVM both are poor on the same target classes whereas RF and ETC show poor performance. LR gives good average results for the Analysis class with a 0.71 F1 score,

Table 12
LR and RF results using FAMTDS for multi-environment dataset.

LR				SVM			
Class	Precision	Recall	F1 Score	Class	Precision	Recall	F1 Score
Analysis	0.80	0.62	0.70	Analysis	1.00	0.42	0.59
Backdoor	0.83	0.92	0.87	Backdoor	0.74	0.96	0.83
DoS	0.86	0.96	0.91	DoS	0.75	0.96	0.84
DoS-Synflooding	1.00	0.92	0.96	DoS-Synflooding	1.00	1.00	1.00
Exploits	0.79	0.92	0.85	Exploits	0.79	0.88	0.83
Fuzzers	0.96	0.92	0.94	Fuzzers	0.96	0.96	0.96
Generic	1.00	1.00	1.00	Generic	1.00	1.00	1.00
MITM ARP Spoofing	0.92	0.92	0.92	MITM ARP Spoofing	0.96	0.92	0.94
Mirai-Ackflooding	0.50	0.42	0.46	Mirai-Ackflooding	0.46	0.46	0.46
Mirai-HTTP Flooding	0.35	0.46	0.40	Mirai-HTTP Flooding	0.41	0.50	0.45
Mirai-Hostbruteforceg	1.00	0.88	0.94	Mirai-Hostbruteforceg	1.00	0.88	0.94
Mirai-UDP Flooding	0.79	0.73	0.76	Mirai-UDP Flooding	0.77	0.65	0.71
Normal-U	1.00	0.96	0.98	Normal-U	1.00	0.96	0.98
Normal-I	0.93	1.00	0.96	Normal-I	0.96	1.00	0.98
Reconnaissance	0.91	0.84	0.87	Reconnaissance	0.85	0.88	0.86
Scan Hostport	0.78	0.84	0.81	Scan Hostport	0.78	0.84	0.81
Scan Port OS	0.83	0.80	0.82	Scan Port OS	0.83	0.80	0.82
Shellcode	0.92	0.92	0.92	Shellcode	0.96	0.88	0.92
Worms	0.96	0.96	0.96	Worms	1.00	0.96	0.98
macro avg	0.85	0.84	0.84	macro avg	0.85	0.84	0.84
weighted avg	0.85	0.84	0.84	weighted avg	0.85	0.84	0.84
Avg. accuracy	0.84			Avg. accuracy	0.84		

Table 13
LR and SVM confusion metrics using FAMTDS for multi-environment dataset.

LR																			
A	16	4	4	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
B	2	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	24	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	23	0	0	0	0	0	2	0	0	0	0	0	0	0	0
E	0	0	0	0	23	1	0	0	0	0	0	0	0	1	0	0	0	0
F	1	0	0	0	1	24	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	25	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	23	1	1	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	1	11	13	0	1	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	9	12	0	4	0	1	0	0	0	0
K	0	0	0	0	0	0	0	1	0	0	22	0	0	1	0	1	0	0
L	0	0	0	0	0	0	0	0	1	6	0	19	0	0	0	0	0	0
M	0	1	0	0	0	0	0	0	0	0	0	0	25	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	26	0	0	0	0
O	1	0	0	0	2	0	0	0	0	0	0	0	0	0	21	0	0	1
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21	4	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	20	0
R	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	24	1
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	25
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S

SVM																			
A	11	8	4	0	2	0	0	0	0	0	0	0	0	0	1	0	0	0	0
B	0	25	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
C	0	0	24	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	1	0	22	1	0	0	0	0	0	0	0	0	1	0	0	0	0
F	0	0	0	0	1	25	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	25	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	23	1	1	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	12	13	0	1	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	9	13	0	4	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	1	0	0	22	0	0	1	0	1	0	0	0
L	0	0	0	0	0	0	0	0	4	5	0	17	0	0	0	0	0	0	0
M	0	1	0	0	0	0	0	0	0	0	0	0	25	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	26	0	0	0	0	0
O	0	0	0	0	2	0	0	0	0	0	0	0	0	0	22	0	0	1	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21	4	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	20	0	0
R	0	0	2	0	0	0	0	0	0	0	0	0	0	0	1	0	0	23	0
S	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	25
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S

while RF, ETC, and SVM show poor performance as they give 0.57, 0.55, and 0.59 F1 scores, respectively. This shows that LR is poor on only two classes Mirai-Ackflooding, and Mirai-HTTP Flooding, while SVM on three, same as the RF and ETC. Overall, the performance of LR, and SVM is good but still, RF is significant with a 0.85 accuracy score.

Table 13 shows the confusion matrix values for the LR and SVM. According to the results, LR is better as compared to SVM in terms of correct and wrong predictions. LR gives 408 corrections out of 485 test predictions and 77 wrong predictions, while SVM gives 406 correct predictions out of 485. For the Analysis class where all other models are poor LR gives 16 correct predictions out of 25, while RF, ETC, and SVM are poor with 12, 12, and 11 correct predictions, respectively.

Table 14 shows the results for the KNN using the FAMTDS approach and according to the results, KNN achieved a 0.81 accuracy score. KNN performs poorly for Mirai-Ackflooding, Mirai-HTTP Flooding, Mirai-UDP Flooding, and Analysis classes as it could achieve only 0.48, 0.50, 0.74, and 0.51 F1 scores respectively. KNN is not as good as other used models because KNN performs well on categorical kind of datasets. According to the KNN confusion matrix, it gives 394 correct predictions and 91 wrong predictions out of 485 predictions as shown in Table 15. KNN is weaker as compared to other used models with FAMTDS. Fig. 6 shows the accuracy upper bound for each model using the FAMTDS approach iteration.

Table 16, shows the results for the IoTID-20 and UNSW-NB15 datasets using the FAMTDS. The proposed approach also performs well for individual datasets as well as for multi-environment datasets. RF, ETC, and SVM achieved significant results with 0.83 accuracy scores for the IoTID-20 dataset while ETC is significant as compared to other models with a 0.87 accuracy score for the UNSW-NB15 dataset.

7.3. Results on multi-environment dataset using PCA, Chi2 features and machine learning models

This section presents the results of machine learning models for the multi-environment dataset with PCA and Chi2 features. Table 17 shows the results for these experiments, where no optimizer is used for feature selection and model tuning. We make the best setting according to the literature knowledge such as we select 30 features from both datasets and merging them to generate a multi-environment dataset. We deploy all models with their best hyperparameter settings. Results show that model performance is not significant with PCA and Chi2 features as compared to the proposed FAMTDS approach. RF is highest with PCA features as it achieved 0.75, 0.83, and 0.76 accuracy scores for IoTID-20, UNSW-NB15, and multi-environment datasets respectively. Similarly, RF is also highest with Chi2 features as it achieved 0.76, 0.81, and 0.79 accuracy scores for IoTID-20, UNSW-NB15, and multi-environment datasets respectively.PCA and Chi2 reduce the size of the feature set, which can potentially lead to decreased efficiency in models. However, in the FAMTDS approach, we employ a feature generation technique after feature selection, which helps mitigate the impact of feature reduction caused by feature selection. Overall, RF is

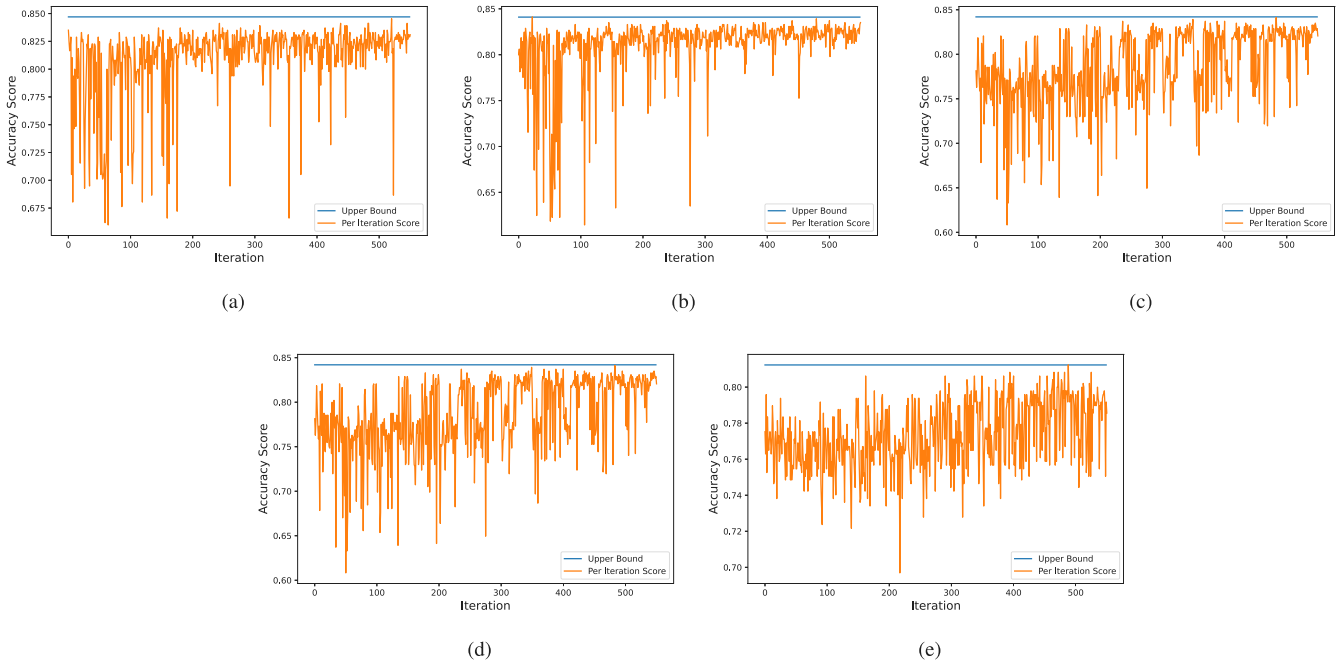


Fig. 6. Accuracy Scores Upper Bound for Each Model using FAMDTS Approach for Multi-Environments Dataset, **(a)** RF, **(b)** ETC, **(c)** LR, **(d)** SVM, and **(e)** KNN.

Table 14
KNN results using FAMTDS for multi-environment dataset.

KNN			
Class	Precision	Recall	F1 Score
Analysis	0.65	0.42	0.51
Backdoor	0.72	0.88	0.79
DoS	0.69	0.88	0.77
DoS-Synflooding	1.00	0.92	0.96
Exploits	0.78	0.84	0.81
Fuzzers	1.00	0.92	0.96
Generic	1.00	1.00	1.00
MITM ARP Spoofing	0.89	0.96	0.92
Mirai-Ackflooding	0.44	0.54	0.48
Mirai-HTTP Flooding	0.47	0.54	0.50
Mirai-Hostbruteforceg	0.87	0.80	0.83
Mirai-UDP Flooding	0.85	0.65	0.74
Normal-U	1.00	0.92	0.96
Normal-I	1.00	1.00	1.00
Reconnaissance	0.88	0.84	0.86
Scan Hostport	0.77	0.80	0.78
Scan Port OS	0.82	0.72	0.77
Shellcode	0.89	0.92	0.91
Worms	0.96	0.88	0.92
macro avg	0.82	0.81	0.81
weighted avg	0.82	0.81	0.81
Avg. accuracy	0.81		

Table 15
KNN confusion metrics using FAMTDS.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
A	11	8	4	0	2	0	0	0	0	0	0	0	0	0	1	0	0	0	0
B	3	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	1	0	22	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0
D	0	0	0	23	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
E	1	0	3	0	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	1	0	0	0	0	24	0	0	0	0	0	0	0	0	1	0	0	0	0
G	0	0	0	0	0	0	25	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	24	1	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	14	11	0	1	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	10	14	0	2	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	3	1	0	20	0	0	0	0	1	0	0	0
L	0	0	0	0	0	0	0	0	5	4	0	17	0	0	0	0	0	0	0
M	0	1	0	0	0	0	0	0	0	0	0	0	24	0	0	0	0	0	1
N	0	0	0	0	0	0	0	0	0	0	0	0	0	26	0	0	0	0	0
O	0	0	1	0	2	0	0	0	0	0	0	0	0	0	21	0	0	1	0
P	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	20	4	0	0
Q	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	5	18	0	0
R	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	24	0
S	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	23

Table 16
Models results using FAMTDS for individual datasets.

Dataset	Model	Accuracy	Precision	Recall	F1 Score
IoTID-20	RF	0.83	0.83	0.83	0.83
	ETC	0.83	0.83	0.83	0.83
	LR	0.81	0.82	0.81	0.82
	KNN	0.77	0.78	0.77	0.78
	SVM	0.83	0.83	0.83	0.83
UNSW-NB15	RF	0.86	0.86	0.86	0.86
	ETC	0.87	0.87	0.87	0.87
	LR	0.85	0.85	0.85	0.85
	KNN	0.81	0.81	0.81	0.81
	SVM	0.85	0.86	0.85	0.85

good with PCA and Chi2 features because it can perform well even on small feature sets, while linear models such as LR, and SVM both require a large feature set.

7.4. Results on multi-environments datasets using NN, image features

This section presents the results of machine learning models using the NN and Image features. For the NN feature, we pass the dataset to ETC for important feature selection. Next, we generate the features using NN architecture as shown in Fig. 3. We extract the feature importance score using ETC and set the threshold on that score in order to select the features. Fig. 7, shows the feature's importance scores and we analyze according to our good knowledge that 0.02 can be a good threshold to select the important feature.

Table 17
Machine learning models results using PCA, Chi2 features.

Dataset	PCA					Chi2				
	Model	Accuracy	Precision	Recall	F1 Score	Model	Accuracy	Precision	Recall	F1 Score
IoTID-20	RF	0.75	0.75	0.75	0.75	RF	0.76	0.77	0.76	0.76
	ETC	0.74	0.74	0.74	0.74	ETC	0.73	0.75	0.73	0.73
	LR	0.60	0.62	0.60	0.60	LR	0.56	0.57	0.56	0.56
	KNN	0.72	0.74	0.72	0.72	KNN	0.71	0.71	0.71	0.71
	SVM	0.66	0.69	0.66	0.67	SVM	0.61	0.63	0.61	0.61
UNSW-NB15	RF	0.83	0.83	0.83	0.83	RF	0.81	0.81	0.81	0.81
	ETC	0.82	0.82	0.82	0.82	ETC	0.80	0.80	0.80	0.79
	LR	0.66	0.67	0.66	0.65	LR	0.68	0.68	0.68	0.67
	KNN	0.78	0.79	0.78	0.78	KNN	0.76	0.77	0.76	0.76
	SVM	0.79	0.81	0.79	0.79	SVM	0.75	0.75	0.75	0.74
Multi-Environment	RF	0.76	0.76	0.76	0.76	RF	0.79	0.79	0.79	0.79
	ETC	0.76	0.77	0.77	0.76	ETC	0.78	0.78	0.78	0.77
	LR	0.55	0.55	0.55	0.54	LR	0.65	0.66	0.65	0.65
	KNN	0.75	0.76	0.76	0.76	KNN	0.76	0.76	0.76	0.76
	SVM	0.75	0.76	0.75	0.75	SVM	0.74	0.74	0.74	0.74

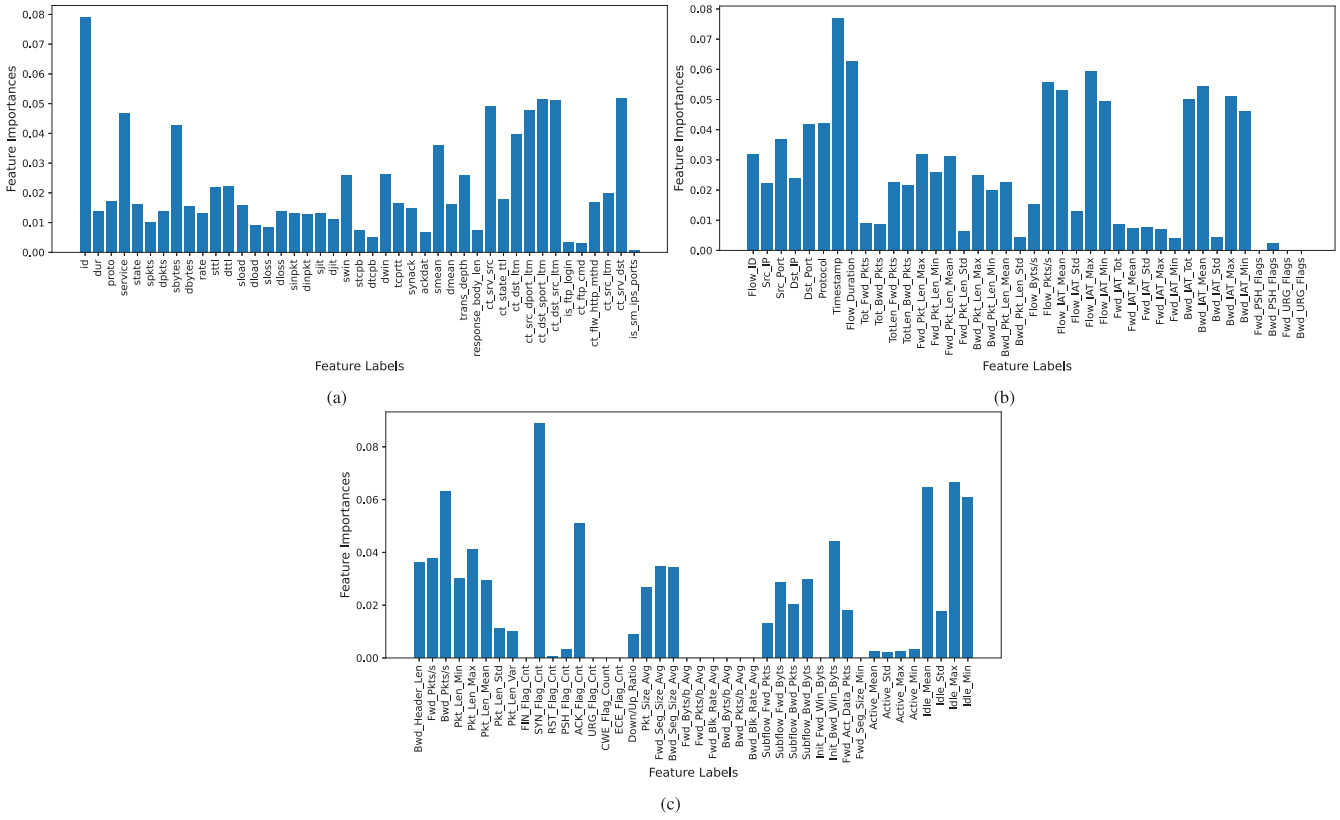


Fig. 7. Feature importance scores using ETC, **(a)** Feature Importance Scores for UNSW-NB15, **(b)** Feature Importance Scores for IoTID-20, First 40 Features, and **(c)** Feature Importance Scores for IoTID-20, Remaining Features.

For image features, we generate images after performing important feature selection and obtaining neural network (NN) features. Using the NN, we generate 4096 features and then convert them into images with dimensions of (64 × 64 × 1). Sample-generated images are shown in Fig. 8. Although the images may appear visually similar, there are differences in their underlying features, which contribute to variations in model performance. To utilize these generated image features, we convert the images into NumPy arrays and pass them to the learning models.

Table 18 presents the results for various machine learning models applied to the IoTID-20, UNSW-NB15, and multi-environment datasets, using both NN and image features. In some cases, RF and ETC perform well, while LR and SVM also exhibit good performance when using image feature sets. This can be attributed to the fact that image data generates a large feature set, which is suitable for linear models like LR and SVM. Based on the results, RF demonstrates good performance

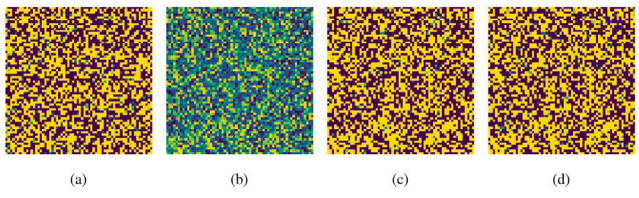


Fig. 8. Generated Images, **(a)** Analysis, **(b)** Normal, **(c)** Generic, and **(d)** Shellcode.

when utilizing NN features for the IoTID-20 dataset, achieving an accuracy score of 0.77. For the UNSW-NB15 dataset, RF, ETC, and SVM performed well, achieving an accuracy score of 0.80. In the case of the multi-environment dataset, RF stands out with an accuracy score of 0.80 when using NN features. When employing image features,

Table 18
Machine learning models results using NN and generated image features.

Dataset	NN					Image				
	Model	Accuracy	Precision	Recall	F1 Score	Model	Accuracy	Precision	Recall	F1 Score
IoTID-20	RF	0.77	0.78	0.77	0.77	RF	0.75	0.75	0.75	0.75
	ETC	0.76	0.76	0.76	0.76	ETC	0.76	0.76	0.76	0.76
	LR	0.73	0.75	0.73	0.74	LR	0.73	0.74	0.73	0.73
	KNN	0.70	0.72	0.70	0.71	KNN	0.70	0.71	0.69	0.70
	SVM	0.76	0.76	0.76	0.76	SVM	0.77	0.76	0.76	0.76
UNSW-NB15	RF	0.80	0.80	0.80	0.80	RF	0.82	0.82	0.82	0.82
	ETC	0.80	0.80	0.80	0.80	ETC	0.80	0.80	0.80	0.80
	LR	0.76	0.76	0.76	0.76	LR	0.80	0.79	0.80	0.79
	KNN	0.78	0.79	0.78	0.78	KNN	0.77	0.88	0.77	0.77
	SVM	0.80	0.80	0.80	0.79	SVM	0.79	0.80	0.79	0.79
Multi-Environment	RF	0.80	0.80	0.80	0.80	RF	0.80	0.80	0.80	0.80
	ETC	0.79	0.80	0.80	0.80	ETC	0.79	0.80	0.79	0.80
	LR	0.79	0.80	0.79	0.79	LR	0.80	0.80	0.80	0.80
	KNN	0.75	0.76	0.75	0.75	KNN	0.75	0.75	0.75	0.75
	SVM	0.77	0.78	0.77	0.77	SVM	0.76	0.77	0.77	0.77

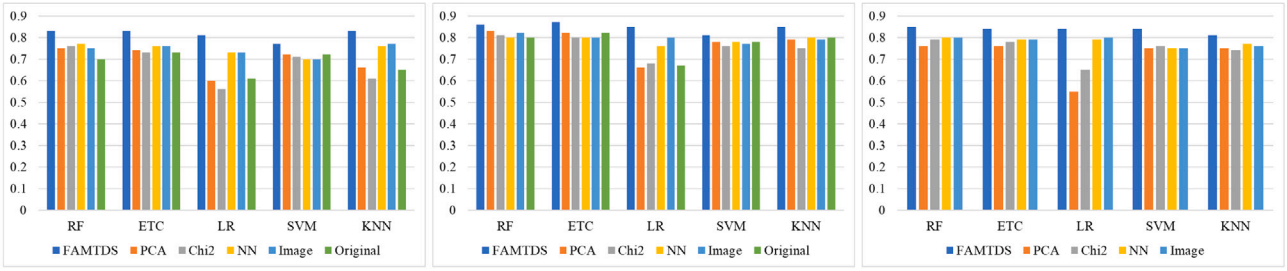


Fig. 9. Accuracy scores comparisons, **(a)** IoTID-20, **(b)** UNSW-NB15, **(c)** Multi-Environment.

SVM performs well with an accuracy score of 0.77 for the IoTID-20 dataset, while RF achieves an accuracy score of 0.82 for the UNSW-NB15 dataset. Both RF and LR exhibit good performance, achieving an accuracy score of 0.80 for the multi-environment dataset. These results indicate that tree-based models perform well on both large and small feature sets, while linear models like LR and SVM excel primarily on large feature sets.

The significance of feature selection prior to applying an NN is evident from the results obtained when using only the NN without feature selection. Our proposed FAMTDS approach, which incorporates feature selection using ETC, outperforms the results achieved without feature selection. Specifically, our experiments conducted on a multi-environment dataset generated without feature selection, where only an NN was utilized for feature generation, yielded a lower accuracy value of 0.80 (as presented in Table 18). In contrast, the proposed approach with feature selection, as demonstrated in Table 10, achieved an accuracy of 0.85. These findings highlight the importance and effectiveness of feature selection in generating a multi-environment dataset

Fig. 9 provides a comparison of the different approaches used, and the graphs indicate that the FAMTDS approach outperforms all others, demonstrating its significance in terms of performance.

7.5. Deep learning models

Deep learning models are deployed using the multi-environment datasets in comparison with our proposed FAMTDS. We deploy several models such as long short-term memory (LSTM) [48,49], convolutional neural networks (CNN) [49], and recurrent neural networks (GRU) [50]. Recently, these models have been mostly used for MTD. LSTM and GRU both are recurrent network-based models which keep the sequence of data from the previous state, while CNN is mostly considered best for the image dataset but it also has lots of applications in the network security domain. The architecture of these used deep learning models is shown in Table 19.

Table 19
Architecture of deep learning models.

Model	Architecture	
GRU	Embedding(50000,100,) Dropout(0.5) GRU(256) SimpleRNN(128) Dense(64) Dense(19, activation = 'softmax')	loss = 'category_crossentropy', optimizer = 'adam', epochs = 100
CNN	Embedding(50000,100,) Dropout(0.5)) Conv1D(128, 4, activation = 'relu') MaxPooling1D(pool_size = 4) Dropout(0.5) Flatten() Dense(64) Dense(19, activation = 'softmax')	
LSTM	Embedding(50000,100,) Dropout(0.5)) LSTM(100) Dense(64) Dense(19, activation = 'softmax')	

All models are used with similar kinds of layers such as embedding layers, dense layers, dropout layers, and sigmoid functions. The embedding layer consists of a 50 000 vocabulary size and a 100 output dimension. After embedding layers, we use a dropout layer for each model with a 0.5 dropout rate, which will help to reduce the model's complexity by deleting neurons randomly during the training. We use the model's own layers after dropout layers such as the GRU layer with 256 units, the LSTM layer with 100 units, and CNN layers used with 128 filters, 5 by 5 kernel size, and ReLU activation function. CNN is also used with a max-pooling layer with a 4 by 4 pool size and flattened layer which will help to change the dimension of data into 1D. A dense

Table 20
Deep learning results for multi-environment Dataset.

Class	LSTM			CNN			GRU		
	Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
Analysis	0.14	0.06	0.09	0.56	0.31	0.40	0.21	0.19	0.20
Backdoor	0.83	1.00	0.91	0.78	1.00	0.88	0.83	1.00	0.91
DoS	0.64	0.59	0.62	0.64	0.67	0.65	0.60	0.78	0.68
DoS-Synflooding	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.96	0.98
Exploits	0.31	0.53	0.39	0.65	0.68	0.67	0.44	0.37	0.40
Fuzzers	0.63	0.41	0.50	1.00	0.72	0.84	0.65	0.52	0.58
Generic	1.00	1.00	1.00	1.00	1.00	1.00	0.96	1.00	0.98
MITM ARP Spoofing	0.78	0.84	0.81	0.97	0.89	0.93	0.97	0.79	0.87
Mirai-Ackflooding	0.45	0.33	0.38	0.42	0.27	0.33	0.50	0.87	0.63
Mirai-HTTP Flooding	0.39	0.62	0.48	0.32	0.42	0.36	0.80	0.17	0.28
Mirai-Hostbruteforceg	0.54	0.35	0.42	0.82	0.70	0.76	0.67	0.50	0.57
Mirai-UDP Flooding	1.00	0.71	0.83	0.70	0.88	0.78	0.86	0.79	0.83
Normal-U	0.69	0.94	0.79	0.97	0.97	0.97	0.91	0.91	0.91
Normal-I	0.71	0.81	0.76	0.97	0.90	0.93	0.82	0.90	0.86
Reconnaissance	0.39	0.39	0.39	0.61	0.61	0.61	0.40	0.44	0.42
Scan Hostport	0.24	0.37	0.29	0.39	0.58	0.47	0.38	0.53	0.44
Scan Port OS	0.57	0.40	0.47	0.63	0.57	0.60	0.66	0.70	0.68
Shellcode	0.45	0.50	0.47	0.77	0.77	0.77	0.55	0.69	0.61
Worms	0.38	0.21	0.27	0.76	0.93	0.84	0.94	0.54	0.68
macro avg	0.59	0.58	0.57	0.74	0.73	0.73	0.69	0.66	0.66
weighted avg	0.61	0.61	0.60	0.76	0.75	0.74	0.72	0.69	0.69
Avg. accuracy	0.61			0.75			0.69		

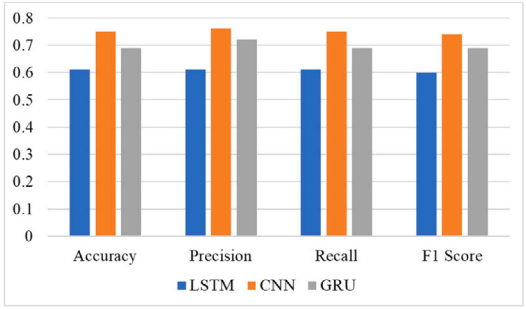


Fig. 10. Results comparison of deep learning models.

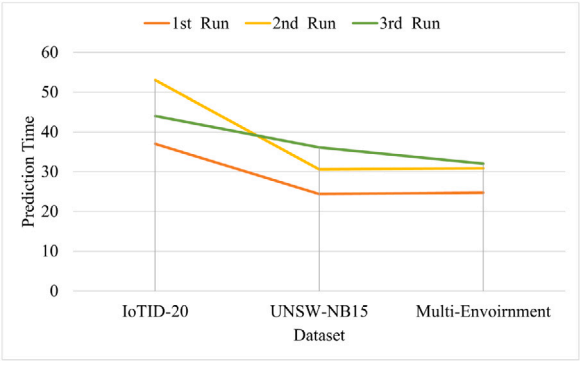


Fig. 11. Prediction time.

layer is used after the model's layers with 64 neurons. The ending layers of models consist of 19 neurons and softmax functions.

CNN demonstrates relative significance compared to LSTM and GRU, achieving a 0.75 accuracy score as shown in Fig. 10. However, it still falls short when compared to our FAMTDS coupled with RF, which attained an accuracy score of 0.85 for the MTD system.

Table 20 presents the performance metrics in detail, including accuracy, precision, recall, and F1 score, for deep learning models. While deep learning models exhibit lower significance compared to machine learning models, we utilized them without the MFO approach due to their longer processing times. Implementing the MFO iteration with deep learning models would prolong the processing time significantly. Consequently, we deployed these models after the feature engineering stage.

We conducted an analysis of prediction times to determine whether it is preferable to deploy separate models for IoT and traditional IP-based networks or implement a unified multi-environment security system. Disregarding the training time, our approach's prediction times (in milliseconds) were compared as depicted in Fig. 11. Surprisingly, the model exhibits nearly equal prediction times for each dataset, although the prediction time for the IoTID data is higher. This difference is attributed to the larger feature set size compared to UNSWNB-15. These results indicate that employing a multi-environment model does not significantly increase prediction times, and the accuracy remains efficient when compared to individual datasets with our proposed approach. So, deploying a single multi-environment security system might hold more significance than implementing separate security systems for distinct networks.

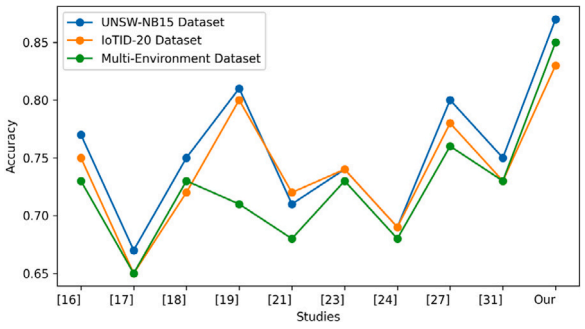


Fig. 12. Comparison with other studies.

7.6. Comparison with other studies

In this section, we compare our proposed approach with other works. We conducted experiments on a new multi-environment dataset and to perform a fair comparison we selected recent studies which concern the same types of network security problems. We deployed the selected studies on our used dataset in the same environment where we deployed our approach. We deployed the systems from selected studies on individual datasets as well as the multi-environment dataset. The accuracy comparison is illustrated in Fig. 12. Our method outperforms all other approaches for each dataset, highlighting its effectiveness in terms of accuracy over the other methods.

Table 21
Detailed comparison of results with other studies.

Ref.	Year	Features	Algorithm	Scores			
				Accuracy	Precision	Recall	F1 Score
UNSW-NB15 Dataset							
[16]	2022	Information gain & Gain ratio	Ensemble	0.77	0.77	0.77	0.77
[17]	2022	–	RNN	0.67	0.67	0.67	0.67
[18]	2022	Packet header+traffic features	Voting	0.75	0.75	0.74	0.74
[19]	2022	PCA	LNN	0.81	0.84	0.81	0.80
[21]	2022	Information Gain	CNN	0.71	0.71	0.71	0.71
[23]	2023	–	Bagging-GBM	0.74	0.73	0.73	0.73
[24]	2023	–	Bi-LSTM	0.69	0.69	0.0.69	0.68
[27]	2022	IF & Merge IF+Chi2	RF	0.80	0.80	0.79	0.79
[31]	2022	Integer encoding scheme	NN	0.75	0.75	0.75	0.74
Our Study	**2024**	**FAMTDS**		**0.87**	**0.87**	**0.87**	**0.87**
IoTID-20 Dataset							
[16]	2022	Information gain & Gain ratio	Ensemble	0.75	0.76	0.75	0.75
[17]	2022	–	RNN	0.65	0.65	0.65	0.65
[18]	2022	Packet header+traffic features	Voting	0.72	0.73	0.72	0.72
[19]	2022	PCA	LNN	0.80	0.80	0.79	0.78
[21]	2022	Information Gain	CNN	0.72	0.72	0.72	0.72
[23]	2023	–	Bagging-GBM	0.74	0.73	0.73	0.73
[24]	2023	–	Bi-LSTM	0.69	0.69	0.68	0.69
[27]	2022	IF & Merge IF+Chi2	RF	0.78	0.78	0.78	0.78
[31]	2022	Integer encoding scheme	NN	0.73	0.73	0.73	0.73
Our Study	**2024**	**FAMTDS**		**0.83**	**0.83**	**0.83**	**0.83**
Multi-Environment Dataset							
[16]	2022	Information gain & Gain ratio	Ensemble	0.73	0.73	0.73	0.73
[17]	2022	–	RNN	0.65	0.65	0.65	0.65
[18]	2022	Packet header+traffic features	Voting	0.73	0.73	0.73	0. 73
[19]	2022	PCA	LNN	0.71	0.69	0.68	0.67
[21]	2022	Information Gain	CNN	0.68	0.69	0.68	0.68
[23]	2023	–	Bagging-GBM	0.73	0.73	0.72	0.72
[24]	2023	–	Bi-LSTM	0.68	0.68	0.68	0.68
[27]	2022	IF & Merge IF+Chi2	RF	0.76	0.76	0.75	0.75
[31]	2022	Integer encoding scheme	NN	0.73	0.73	0.73	0.73
Our Study	**2024**	**FAMTDS**		**0.85**	**0.85**	**0.85**	**0.84**

Table 21 contains the detailed experimental results from previous studies. The studies that contributed to both feature engineering and model development achieved significant results compared to those that focused solely on complex deep learning models. For example, studies such as [19,27] worked on both feature engineering and model development, leading to significant results. In contrast, studies such as [23,24] focused on complex deep learning models without prioritizing feature engineering, resulting in poor accuracy. Our study, on the other hand, contributed to feature engineering, model training, and process optimization, which has been lacking in recent studies in this domain. This makes our proposed approach noteworthy.

7.7. Performance validation of proposed approach for multi-environment-2 dataset

To demonstrate the significance of our proposed approach, we expanded our experiments to include additional, contemporary datasets to address the latest attack vectors. For this purpose, we incorporated the CICDDOS2019 dataset into our study to generate multi-environment traffic [13]. This dataset encompasses 17 attack types, including "UDP", "MSSQL", "Syn", "DrDoS_DNS", "NetBIOS", "DrDoS_LDAP", "DrDoS_NetBIOS", "TFTP", "DrDoS_NTP", "Portmap", "UDPLag", "LDAP", "UDP-lag", "WebDDoS", "DrDoS_SNMP", "DrDoS_UDP", and "DrDoS_MSSQL". We combined this dataset with the IoTID-20 dataset to create another multi-environment dataset, termed multi-environment-2. The performance of our proposed approach for the multi-environment-2 dataset is presented in Table 22.

Table 22
Performance validation of proposed approach using multi-environment-2 dataset.

Model	Accuracy	Precision	Recall	F1 Score
RF	0.82	0.82	0.82	0.81
ETC	0.82	0.81	0.82	0.81
LR	0.81	0.81	0.81	0.80
SVM	0.81	0.80	0.81	0.80
KNN	0.80	0.80	0.80	0.80

The results indicate that our approach is significantly effective across both the multi-environment and multi-environment-2 datasets. Specifically, the RF, and ETC models exhibited remarkable performance, achieving accuracy scores of 0.82 and F1 scores of 0.81, respectively. The LR, KNN, and SVC models also performed well, closely following the RF, ETC models. They achieved accuracy scores of 0.81, 0.80, and 0.81, respectively.

7.8. Generalization validation of proposed approach for zero-day attacks in multi-environment networks

In our multi-environment dataset, we excluded "Generic Attack" and "Normal" data from the UNSW-NB dataset, treating them as totally unseen data while passing the other 17 attack samples to the model for training. We conducted these experiments with our proposed approach, and the results are presented in Table 23. Similarly, we predicted zero-day attacks using the IoTID-20 and CICDDOS2019

Table 23
Proposed approach generalization validation for multi-environment datasets.

Dataset	Accuracy	Class	Precision	Recall	F1 Score
Multi-environment	0.84	Attack	0.78	0.95	0.85
		Normal	0.94	0.72	0.82
		Macro avg	0.86	0.84	0.84
		Weighted avg	0.86	0.84	0.84
Multi-environment-2	0.81	Attack	0.95	0.83	0.89
		Normal	0.32	0.64	0.43
		Macro avg	0.64	0.74	0.66
		Weighted avg	0.88	0.81	0.84

(multi-environment-2) datasets. For model training, we used "Benign", "UDP", "MSSQL", "Syn", "DrDoS_DNS", "NetBIOS", "DrDoS_LDAP", "DrDoS_NetBIOS", "TFTP", and "DrDoS_NTP", and identified the following as zero-day attacks: "Portmap", "UDPLag", "LDAP", "UDP-lag", "WebDDoS", "DrDoS_SNMP", "DrDoS_UDP", and "DrDoS_MSSQL" (Attack).

Through our experiments, we found that the model demonstrated strong performance in attack prediction, even on unseen data. The performance of our proposed approach was significant, achieving 0.84 accuracy. Although there was some fluctuation in the "Normal" class, the results for "Attack" were significant. Conversely, the multi-environment-2 proposed approach, while not as significant, was still acceptable in terms of zero-day attack detection, achieving an accuracy score of 0.81 and an F1 score of 0.66. These results demonstrate the generalizability of our proposed approach.

7.9. Limitations

This study significantly contributes to the field of network security by introducing a novel approach for multi-environment MTD. While the proposed approach offers valuable insights, it is essential to acknowledge its limitations:

- **Limited network architecture support:** Our study focuses on simple networks like smart homes or small offices, predominantly handling IoT or IP-based conventional network traffic (refer to Fig. 2). The datasets used in our experiments do not encompass all network types, such as Software Defined Networking (SDN), wireless, and Radio Access Networks (RAN), leading to potential compatibility issues due to the diverse nature of each network.
- **Absence of benchmark dataset:** The lack of an available multi-environment dataset necessitated our creation of one by amalgamating two distinct open-source datasets. Consequently, a benchmark dataset becomes imperative to validate our proposed approach, a development we plan for future studies.
- **Small dataset size:** In addressing data imbalance, our utilization of under-sampling led to a reduction in dataset size. This reduction impacts the accuracy of the proposed approach.

8. Conclusion & future direction

In this paper, we focused on MTD for multi-environment networks. We highlighted the gap and discussed the importance of security in multi-environment networks. Our proposed approach, FAMTDS, involves generating a novel multi-environment traffic dataset by leveraging traditional IP-based networks and IoT network-based datasets. We deployed machine learning models and optimized the system using an MFO optimizer. FAMTDS achieved a significant accuracy score of 0.85, outperforming state-of-the-art methods and studies. Through extensive literature and experiment analysis, we concluded several points which are listed below:

- Existing datasets can be used to generate multi-environment traffic, but they are limited to specific types of architectures. In this study, we generated a novel dataset for multi-environment MTD experiments by utilizing the UNSW-NB15 and IoTID-20 datasets, specifically to support smart home multi-environment architecture.
- We concluded that good feature engineering, coupled with appropriate hyperparameter settings for the machine learning model, can result in higher accuracy and efficiency. For instance, in this study, the methods without optimization achieved a maximum accuracy score of only 0.79 for MTD. However, when these same methods were applied in an optimized environment, the accuracy score increased to 0.85.
- To provide a much-optimized environment in terms of feature selection and model hyper-parameters settings, optimizers can help generate significant results. In this study, MFO helped to select the best feature, generate significant numbers of features, tune the models according to the dataset nature, and help to achieve significant results for MTD.
- Despite the challenges posed by the dataset size, our FAMTDS approach demonstrated promising results on this small and complex dataset, achieving a good accuracy score compared to state-of-the-art approaches. We are committed to addressing the limitations raised by the dataset size in our future work and to ensuring a more comprehensive and representative dataset for improved accuracy and generalization of the attack detection model.

We also highlighted certain limitations in Section 7.9. In future research, we aim to address these limitations. For instance, we plan to generate a benchmark dataset for multi-environment traffic that will support both large and small networks. Additionally, we intend to test our approach in a real-time environment and validate its performance. Furthermore, we will take into consideration different types of architectures, such as SDN, RAN, etc., within the scope of FAMTDS.

Funding statement

This work was funded by the Irish Research Council for the CHIST-ERA-22-SPiDDS-07 project.

CRediT authorship contribution statement

Furqan Rustam: Writing – original draft, Visualization, Software, Validation, Methodology, Investigation, Formal analysis, Data curation, Conceptualization, Writing – review & editing. **Wajdi Aljedaani:** Visualization, Validation, Investigation, Formal analysis. **Mahmoud Said Elsayed:** Writing – review & editing, Validation, Investigation, Formal analysis. **Anca Delia Jurcut:** Writing – review & editing, Validation, Supervision, Software, Project administration, Formal analysis, Conceptualization.

Declaration of competing interest

We declare that there are no conflicts of interest that could potentially bias the interpretation of the findings or influence the impartiality of this work. This manuscript has been prepared and submitted with the utmost transparency and adherence to ethical standards. We have no financial or personal relationships with individuals or organizations that might have influenced the content or conclusions presented in this submission.

Dataset & code availability

The code and data related to the experiments are available at the provided link: https://github.com/furqanrustam/FAMTD-Malicious-Traffic-Detection.

References

[1] A.K. Blaskovic, J.-D. Rusk, V.C. Parker, B.R. Payne, Cybercrime and intellectual property theft: An analysis of modern digital forensics, in: Proceedings of the Future Technologies Conference, Springer, 2023, pp. 536–542.

[2] N. Capuano, G. Fenza, V. Loia, C. Stanzione, Explainable artificial intelligence in CyberSecurity: A survey, IEEE Access 10 (2022) 93575–93600.

[3] O. Powell, The biggest data breaches and leaks of 2022, 2022, https://www.cshub.com/attacks/articles/the-biggest-data-breaches-and-leaks-of-2022. (Accessed 11 December 2022).

[4] M. Mclean, 2022 Must-know cyber attack statistics and trends, 2022, https://www.embroker.com/blog/cyber-attack-statistics/. (Accessed 11 December 2022).

[5] L. Cvetkovska, 30 Smart home statistics for all high-tech enthusiasts, 2022, https://comfyliving.net/smart-home-statistics/. (Accessed 11 December 2022).

[6] B. Morel, Artificial intelligence and the future of cybersecurity, in: Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, AISec '11, Association for Computing Machinery, New York, NY, USA, 2011, pp. 93–98.

[7] E.T. Ogidan, K. Dimililer, Y. Kirsal-Ever, Chapter two - machine learning for cyber security frameworks: a review, in: F. Al-Turjman (Ed.), Drones in Smart-Cities, Elsevier, 2020, pp. 27–36, URL https://www.sciencedirect.com/science/article/pii/B9780128199725000021.

[8] Z. Wang, K.W. Fok, V.L. Thing, Machine learning for encrypted malicious traffic detection: Approaches, datasets and comparative study, Comput. Secur. 113 (C) (2022).

[9] E.G. Dada, J.S. Bassi, H. Chiroma, S.M. Abdulhamid, A.O. Adetunmbi, O.E. Ajibuwa, Machine learning for email spam filtering: review, approaches and open research problems, Heliyon 5 (6) (2019) e01802, URL https://www.sciencedirect.com/science/article/pii/S2405844018353404.

[10] Balbix, Using artificial intelligence in cybersecurity, 2022, https://www.balbix.com/insights/artificial-intelligence-in-cybersecurity/. (Accessed 11 December 2022).

[11] I. Ullah, Q.H. Mahmoud, A scheme for generating a dataset for anomalous activity detection in iot networks, in: Canadian Conference on Artificial Intelligence, Springer, 2020, pp. 508–520.

[12] N. Moustafa, J. Slay, UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set), in: 2015 Military Communications and Information Systems Conference, MilCIS, IEEE, 2015, pp. 1–6.

[13] I. Sharafaldin, A.H. Lashkari, S. Hakak, A.A. Ghorbani, Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy, in: 2019 International Carnahan Conference on Security Technology, ICCST, IEEE, 2019, pp. 1–8.

[14] A.N. Jahromi, H. Karimipour, A. Dehghantanha, K.-K.R. Choo, Toward detection and attribution of cyber-attacks in IoT-enabled cyber–physical systems, IEEE Internet Things J. 8 (17) (2021) 13712–13722.

[15] N. Sridhar, L. Yang, J. Joshi, V. Piotrowski, Cybersecurity education in the age of artificial intelligence, in: Proceedings of the 52nd ACM Technical Symposium on Computer Science Education, SIGCSE '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 1365.

[16] K. Albulayhi, Q. Abu Al-Haija, S.A. Alsuhibany, A.A. Jillepalli, M. Ashrafuzzaman, F.T. Sheldon, IoT intrusion detection using machine learning with a novel high performing feature selection method, Appl. Sci. 12 (10) (2022) 5015.

[17] M. Baz, SEHIDS: Self evolving host-based intrusion detection system for IoT networks, Sensors 22 (17) (2022) 6505.

[18] P. Illy, G. Kaddoum, K. Kaur, S. Garg, ML-based IDPS enhancement with complementary features for home IoT networks, IEEE Trans. Netw. Serv. Manag. 19 (2) (2022) 772–783.

[19] R. Zhao, G. Gui, Z. Xue, J. Yin, T. Ohtsuki, B. Adebisi, H. Gacanin, A novel intrusion detection method based on lightweight neural network for internet of things, IEEE Internet Things J. 9 (12) (2022) 9960–9972.

[20] A. Makkar, S. Garg, N. Kumar, M.S. Hossain, A. Ghoneim, M. Alrashoud, An efficient spam detection technique for IoT devices using machine learning, IEEE Trans. Ind. Inform. 17 (2) (2021) 903–912.

[21] B.I. Hairab, M. Said Elsayed, A.D. Jurcut, M.A. Azer, Anomaly detection based on CNN and regularization techniques against zero-day attacks in IoT networks, IEEE Access 10 (2022) 98427–98440.

[22] U. Islam, R.Q. Malik, A.S. Al-Johani, M.R. Khan, Y.I. Daradkeh, I. Ahmad, K.A. Alissa, Z. Abdul-Samad, E.M. Tag-Eldin, A novel anomaly detection system on the internet of railways using extended neural networks, Electronics 11 (18) (2022) 2813.

[23] B.A.T. Maya Hilda Lestari Louk, Dual-IDS: A bagging-based gradient boosting decision tree model for network anomaly intrusion detection system, Expert Syst. Appl. 213 (2023) 119030.

[24] Y. Yang, S. Tu, R.H. Ali, H. Alasmary, M. Waqas, M.N. Amjad, Intrusion detection based on bidirectional long short-term memory with attention mechanism, Comput., Mater. Continua 74 (1) (2023) 801–815, URL http://www.techscience.com/cmc/v74n1/49832.

[25] J. Lan, X. Liu, B. Li, J. Sun, B. Li, J. Zhao, MEMBER: A multi-task learning model with hybrid deep features for network intrusion detection, Comput. Secur. 123 (2022) 102919, URL https://www.sciencedirect.com/science/article/pii/S016740482200311X.

[26] R.A. Disha, S. Waheed, Performance analysis of machine learning models for intrusion detection system using Gini Impurity-based Weighted Random Forest (GIWRF) feature selection technique, Cybersecurity 5 (1) (2022) 1–22.

[27] B. Habib, F. Khursheed, Performance evaluation of machine learning models for distributed denial of service attack detection using improved feature selection and hyper-parameter optimization techniques, Concurr. Comput.: Pract. Exper. 34 (26) (2022) e7299, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.7299.

[28] C. Davis, Home network segmentation: a must in the iot era, 2018, https://www.ckd3.com/blog/2018/10/15/home-network-segmentation-a-must-in-the-iot-era. (Accessed 11 December 2022).

[29] W. Wei, Casino gets hacked through its internet-connected fish tank thermometer, 2018, https://thehackernews.com/2018/04/iot-hacking-thermometer.html. (Accessed 11 December 2022).

[30] A. Chohra, P. Shirani, E.B. Karbab, M. Debbabi, Chameleon: Optimized feature selection using particle swarm optimization and ensemble methods for network anomaly detection, Comput. Secur. 117 (2022) 102684, URL https://www.sciencedirect.com/science/article/pii/S0167404822000827.

[31] M. Rani, et al., Effective network intrusion detection by addressing class imbalance with deep neural networks multimedia tools and applications, Multimedia Tools Appl. 81 (6) (2022) 8499–8518.

[32] M.S. Elsayed, N.-A. Le-Khac, A.D. Jurcut, InSDN: A novel SDN intrusion dataset, IEEE Access 8 (2020) 165263–165284.

[33] P.L. Indrasiri, E. Lee, V. Rupapara, F. Rustam, I. Ashraf, Malicious traffic detection in iot and local networks using stacked ensemble classifier, Comput. Mater. Contin. 71 (1) (2022) 489–515.

[34] F. Rustam, A.D. Jurcut, Malicious traffic detection in multi-environment networks using novel S-DATE and PSO-D-SEM approaches, Comput. Secur. 136 (2024) 103564.

[35] D. Baby, S.J. Devaraj, J. Hemanth, et al., Leukocyte classification based on feature selection using extra trees classifier: atransfer learning approach, Turk. J. Electr. Eng. Comput. Sci. 29 (8) (2021) 2742–2757.

[36] V. Gaur, R. Kumar, FSMDAD: Feature selection method for DDoS attack detection, in: 2022 International Conference on Electronics and Renewable Systems, ICEARS, 2022, pp. 939–944.

[37] ML | Extra tree classifier for feature selection, 2022, Geeks for Geeks. Online; https://www.geeksforgeeks.org/ml-extra-tree-classifier-for-feature-selection/. (Accessed 25 November 2022).

[38] M.A. Khan, M.I. Sharif, M. Raza, A. Anjum, T. Saba, S.A. Shad, Skin lesion segmentation and classification: A unified framework of deep neural network features fusion and selection, Expert Syst. 39 (7) (2022) e12497.

[39] F. Rustam, A. Ishaq, K. Munir, M. Almutairi, N. Aslam, I. Ashraf, Incorporating CNN features for optimizing performance of ensemble classifier for cardiovascular disease prediction, Diagnostics 12 (6) (2022) 1474.

[40] S.A.I. Alfarozi, K. Pasupa, M. Sugimoto, K. Woraratpanya, Local sigmoid method: Non-iterative deterministic learning algorithm for automatic model construction of neural network, IEEE Access 8 (2020) 20342–20362.

[41] S. Seth, K.K. Chahal, G. Singh, A novel ensemble framework for an intelligent intrusion detection system, IEEE Access 9 (2021) 138451–138467.

[42] L. Abhishek, Optical character recognition using ensemble of SVM, MLP and extra trees classifier, in: 2020 International Conference for Emerging Technology, INCET, 2020, pp. 1–4.

[43] S. Tufail, S. Batool, A.I. Sarwat, A comparative study of binary class logistic regression and shallow neural network for DDoS attack prediction, in: SoutheastCon 2022, 2022, pp. 310–315.

[44] H. Wang, Y. Shao, S. Zhou, C. Zhang, N. Xiu, Support vector machine classifier via $L_{0/1}$L0/1 soft-margin loss, IEEE Trans. Pattern Anal. Mach. Intell. 44 (10) (2022) 7253–7265.

[45] X. Gao, C. Shan, C. Hu, Z. Niu, Z. Liu, An adaptive ensemble machine learning model for intrusion detection, IEEE Access 7 (2019) 82512–82521.

[46] M. Shehab, L. Abualigah, H. Al Hamad, H. Alabool, M. Alshinwan, A.M. Khasawneh, Moth–flame optimization algorithm: variants and applications, Neural Comput. Appl. 32 (14) (2020) 9859–9884.
[47] S. Mirjalili, Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm, Knowl.-Based Syst. 89 (2015) 228–249, URL https://www.sciencedirect.com/science/article/pii/S0950705115002580.
[48] Y. Yu, X. Zeng, X. Xue, J. Ma, LSTM-based intrusion detection system for VANETs: A time series classification approach to false message detection, IEEE Trans. Intell. Transp. Syst. 23 (12) (2022) 23906–23918.
[49] A. Halbouni, T.S. Gunawan, M.H. Habaebi, M. Halbouni, M. Kartiwi, R. Ahmad, CNN-LSTM: hybrid deep neural network for network intrusion detection system, IEEE Access 10 (2022) 99837–99849.
[50] M.S. Ansari, V. Bartoš, B. Lee, GRU-based deep learning approach for network intrusion alert prediction, Future Gener. Comput. Syst. 128 (2022) 235–247.

Furqan Rustam received the M.S.C.S. degree from the Department of Computer Science, Khwaja Fareed University of Engineering and Information Technology (KFUEIT), Rahim Yar Khan, Pakistan. He worked as a Research Assistant at the Fareed Computing and Research Center, KFUEIT. He also worked as a Lecturer at UMT, Lahore. He is currently pursuing a Ph.D. degree with the School of Computer Science, University College Dublin, Ireland. His recent research interests include data mining, machine learning, and artificial intelligence, mainly working on creative computing, and supervised machine learning.

Wajdi Aljedaani received a bachelor's degree in software engineering from the Athlone Institute of Technology, Ireland, in 2014, and a master's degree in software engineering from the Rochester Institute of Technology, New York, in 2016. He is currently pursuing a Ph.D. degree in computer science and engineering at the University of North Texas. He worked as a Lecturer at the Al-Khari College of Technology, Saudi Arabia, from 2017 to 2020. His research interests include software engineering, mining software repository, accessibility, machine learning, and text mining.

Mahmoud Said Elsayed received a B.E. degree in electronics and communication engineering from Zagazig University, Egypt, in 2007, and an M.E. degree in information security from Nile University, Egypt, in 2018. He is currently pursuing a Ph.D. degree with the School of Computer Science, University College Dublin (UCD), Dublin, Ireland. He has worked for several years in the industry through Huawei and IBM Company in the area of computer networks and security. His research interests include computer networks, network security, deep learning, and cloud computing.

Anca Delia Jurcut received a B.Sc. degree in computer science and mathematics from the West University of Timisoara, Romania, in 2007, and a Ph.D. degree in security engineering from the University of Limerick (UL), Ireland, in 2013, funded by the Irish Research Council for Science Engineering and Technology. She has been an Assistant Professor at the School of Computer Science, University College Dublin (UCD), Ireland, since 2015. She worked as a Postdoctoral Researcher at UL, a member of the Data Communication Security Laboratory, and as a Software Engineer in IBM, Dublin, Ireland, in the areas of data security and formal verification. Her research interests include security protocols design and analysis, automated techniques for formal verification, network security, attack detection, and prevention techniques, security for the Internet of Things, and applications of blockchain for security and privacy. She has several key contributions to research focusing on the detection and prevention techniques of attacks over networks, the design and analysis of security protocols, automated techniques for formal verification, and security for mobile edge computing (MEC).