

MULTI-LF: A Unified Continuous Learning Framework for Real-Time DDoS Detection in Multi-Environment Networks

Furqan Rustam, Islam Obaidat, Anca Delia Jurcut, *Member, IEEE*

Abstract—Detecting Distributed Denial of Service (DDoS) attacks in Multi-Environment (M-En) networks presents significant challenges due to diverse malicious traffic patterns and the evolving nature of cyber threats. Existing AI-based detection systems struggle to adapt to new attack strategies and lack real-time attack detection capabilities with high accuracy and efficiency. This study proposes an online, continuous learning methodology for DDoS detection in M-En networks, enabling continuous model updates and real-time adaptation to emerging threats, including zero-day attacks. First, we develop a unique M-En network dataset by setting up a realistic, real-time simulation using the NS-3 tool, incorporating both victim and bot devices. DDoS attacks with varying packet sizes are simulated using the DDoSim application across IoT and traditional IP-based environments under M-En network criteria. Our approach employs a multi-level framework (MULTI-LF) featuring two machine learning models: a lightweight Model 1 (M1) trained on a selective, critical packet dataset for fast and efficient initial detection, and a more complex, highly accurate Model 2 (M2) trained on extensive data. When M1 exhibits low confidence in its predictions, the decision is escalated to M2 for verification and potential fine-tuning of M1 using insights from M2. If both models demonstrate low confidence, the system flags the incident for human intervention, facilitating model updates with human-verified categories to enhance adaptability to unseen attack patterns. We validate the MULTI-LF through real-world simulations, demonstrating superior classification accuracy of 0.999 and low prediction latency of 0.866 seconds compared to established baselines. Furthermore, we evaluate performance in terms of memory usage (3.632 MB) and CPU utilization (10.05%) in real-time scenarios.

Index Terms—DDoS Attacks, Network Emulation, Dataset Collection, Continuous Learning, M-En Networks, Zero-Day Attacks

I. INTRODUCTION

Distributed Denial-of-Service (DDoS) attacks pose a critical challenge to network security by overwhelming target servers or networks with excessive traffic, thereby depleting their bandwidth, memory, and processing resources [1]. Attackers employ a variety of techniques to orchestrate these attacks. One prevalent method involves the deployment of botnets—networks of compromised devices such as *Internet of Things* (IoT) devices infected with malware—which are remotely controlled to generate vast amounts of traffic directed at the target [2]. In addition, adversaries exploit inherent properties of certain network protocols by involving intermediary servers (such as *Domain Name System* (DNS) resolver, Network Time Protocol server, and Simple Service Discovery Protocol services) to magnify attack traffic without the need to compromise a large number of devices directly [3], [4].

To combat these diverse and evolving DDoS attack methods, a plethora of *machine learning* (ML) models have been developed to detect and mitigate DDoS attacks for specific network types (for IoT network traffic [5], software-defined networking traffic [6], [7], and traditional network traffic, c.f., [8]). Although these approaches show good results in detecting DDoS attacks on specific network types,

they struggle in handling complex, heterogeneous environments like *multi-environment* (M-En) networks [9], [10].

M-En networks, which integrate multiple network types under a single environment, present unique challenges for DDoS detection due to multiple challenges. First, there is a lack of benchmark datasets to train models effectively. Second, the traffic patterns vary widely across IoT and traditional networks, each with unique packet characteristics; as a result, a model trained on one network type, like IoT, may not effectively identify patterns in traditional traffic. Another key challenge is the bias toward specific network traffic; if one traffic dominates, the model may overfit this traffic type, reducing performance across other networks. These challenges make it difficult to create a unified model that accurately generalizes across all network environments [11].

Several studies explore M-En network security and address these challenges through various approaches. For instance, Rustam et al. [10] propose a fully automated malicious traffic detection system (MTDS) that addresses the lack of M-En datasets by generating M-En traffic through a combined IoT and traditional network dataset (IoTID-20 [12] and UNWNB-15 [13]). They tackle traffic diversity by deploying a Moth Flame Optimizer to find optimal weights of an ML algorithm for both network traffic types. Similarly, another study by Rustam et al. [14] employs an optimization approach to create M-En traffic, including IoT and traditional network traffic. They propose a self-diverse ensemble model by combining three variations of random forests optimized with a particle swarm optimizer. Most current studies create M-En traffic by merging existing datasets, which may not capture real-time M-En network patterns, and combining features can result in the loss of meaningful traffic patterns. Furthermore, no study currently addresses continuous learning, which could lead to performance degradation of these models over time.

This study addresses the challenges in securing M-En networks against DDoS attacks in two main contributions. In the first contribution, we introduce a comprehensive approach to M-En dataset collection that leverages the DDOSHIELD-IoT testbed [5], which integrates NS-3 (a discrete-event network simulator) with Docker containers. This testbed allows us to generate real-world M-En traffic by running actual binaries in a simulated network, producing both benign traditional traffic (FTP, HTTP, and RTMP-based video) and malicious IoT traffic derived from Mirai malware binaries. To achieve a complete range of traffic types—namely benign IoT and malicious traditional traffic—we incorporate external PCAP datasets from the literature, ensuring our final dataset accurately reflects diverse, real-world network conditions.

Following dataset construction, we uniformly process all collected packets through a consistent feature extraction pipeline, capturing both packet-level and time-based statistical features. This uniformity ensures that traffic from all sources—DDOSHIELD-IoT and external PCAP datasets—can be directly compared and integrated. As a result, the dataset provides a robust foundation for training ML models capable of detecting DDoS attacks in M-En

Furqan Rustam and Anca Delia Jurcut are with the School of Computer Science, University College Dublin, Ireland, e-mail: (furqan.rustam@ucdconnect.ie; anca.jurcut@ucd.ie)

Islam Obaidat is in the Department of Computer Systems Technology, North Carolina A&T State University, NC, USA, e-mail: (iaobaidat@n)

networks, with improved reliability and generalizability compared to prior approaches that relied solely on merged feature sets without preserving raw packet-level information.

In the second contribution, we develop and implement a continuous learning technique that combines ML and human (expert) intervention. Then, we deploy a real-time, continuous learning approach using a multi-level framework involving two models: M1 and M2. M1 is a lightweight model optimized for continuous learning with low computational costs and is initially trained on a smaller selected samples dataset to allow faster adaptability to new patterns. M2, in contrast, is a more robust and complex model with higher accuracy, trained on a larger, more comprehensive dataset. When a new packets arrives, M1 attempts a prediction. If it is unable to make a confident prediction, the data is passed to M2 for further analysis. Should M2 also fail to reach a definitive prediction, the data is then flagged to human experts for manual verification. Based on the expert's feedback, M1 undergoes retraining to integrate the new insights, allowing it to enhance its predictive accuracy over time by incorporating this human-reviewed data into its model.

In summary, the key contributions of this work are as follows:

- It proposes a real-time Multi-level Framework (MULTI-LF) for DDoS attack detection in M-En networks.
- It introduces an M-En traffic dataset constructed by leveraging the DDOSHIELD-IOT testbed and integrating external PCAP datasets. This dataset encompasses both IoT and traditional (non-IoT) traffic, including benign and malicious categories, thereby achieving a comprehensive and realistic representation of M-En network conditions.
- A feature engineering approach is proposed to extract both general and statistical features by collecting packets within a specific time frame. Combining statistical features (e.g., Packet Counts, Destination Port Entropy) with general features creates a more linearly separable feature space, leading to substantial performance gains.
- MULTI-LF is tested and validated in real-time scenarios by deploying it in NS-3 simulations. This approach has demonstrated greater accuracy compared to state-of-the-art algorithms.
- The performance of all models is evaluated based on accuracy, precision, recall, F1 score, model size, memory usage, CPU utilization, and prediction time.

The rest of this paper is organized as follows: Section II presents the related work in the problem domain. Section III describes the proposed methodology. Section IV discusses the results, and Section VI provides the conclusion and future work.

II. RELATED WORK

This section discusses the related works on malicious traffic detection for traditional networks, IoT networks, and *multi-environment* (M-En) networks. Furthermore, we explore the literature on continuous learning and identify gaps addressed by our study.

A. Malicious Traffic Detection using Machine Learning

Malicious traffic, particularly *Distributed Denial-of-Service* (DDoS) attacks, poses significant threats to network security. Many researchers have worked on efficiently tackling DDoS. For example, Anley et al. [15] propose an innovative approach that utilizes *Convolutional Neural Networks* (CNNs), adaptive architectures, and transfer learning techniques to detect malicious traffic. Their method demonstrates robust detection capabilities across various attack categories and is validated on publicly available datasets

such as CIC-DDoS2019 [16], CSE-CIC-IDS2018 [17], UNSW-NB15 [13], and KDDCup'99 [18]. Their approach achieved accuracies of 93.62%, 99.92%, 99.84%, and 98.99% on each dataset, respectively. In a comparative study, Al-Eryani et al. [19] evaluate *machine learning* (ML) algorithms using the CIC-DoS2019 dataset [16], finding that *Gradient Boosting* (GB) and XGBoost achieve high accuracy (GB: 99.99%, XGBoost: 99.98%) with minimal false alarms.

Persistent challenges in DDoS defense stem from evolving attack vectors and the increasing complexity of network environments. In response, Zhao et al. [1] have developed DFNet, an approach that integrates advanced ML models with packet scheduling algorithms in the network data plane. This approach effectively forwards 99.93% of victim-desired traffic during new DDoS attacks while incurring minimal overhead. Similarly, Singh et al. [20] contribute to the detection and mitigation of DDoS attacks in *software-defined networking* (SDN) environments. Their work also presents a new DDoS dataset with over 1.7 million entries and employs two detection methods: Snort [21] (an Intrusion Detection System) and eight different ML algorithms, including Ensemble Classifiers and a Hybrid Support vector machine-Random Forest (SVM-RF) classifier. The detection methods achieved 99.1% accuracy. In addition, the authors suggested two strategies to mitigate DDoS traffic: dropping illegitimate traffic and redirecting it.

B. Malicious Traffic Detection in Traditional IP-based Network

Traditional IP-based networks—networks that primarily handle non-IoT traffic using the Internet Protocol, distinguishing them from networks that may employ protocols like MQTT for IoT devices—have been extensively explored, resulting in numerous benchmark datasets and efficient security approaches. In recent literature, several researchers have proposed methods to protect these networks from malicious actors. Talukder et al. [22] present an ML-based network intrusion detection model that integrates *Random Oversampling* (RO) to counter data imbalance, Stacking Feature Embedding derived from clustering results, and *Principal Component Analysis* (PCA) for dimension reduction. Their model achieves exceptional accuracy rates: 99.59% and 99.95% with RF and *Extra Trees* (ET) models on the UNSW-NB15 dataset [13], 99.99% on the CIC-IDS-2017 dataset [23], and 99.94% on the CIC-IDS-2018 dataset [17] with DT and RF models, respectively.

Expanding the focus on network security, Casanova et al. [24] concentrate on transforming the CIRACIC-DoHBrw-2020 time-series dataset [24] for training deep learning models in network intrusion detection. Their approach includes a two-layer network classification strategy, distinguishing *DNS over HTTPS* (DoH) from non-DoH traffic and further classifying DoH traffic into benign and malicious categories using a subset of 26 features and various types of *Recurrent Neural Networks* (RNNs), such as *Long Short-Term Memory* (LSTM), Bidirectional LSTM, *Gated Recurrent Unit* (GRU), and Deep RNN. Bi-LSTM outperforms all other methods with 99% accuracy, while GRU is the second-best performer. Hema et al. [25] propose a novel feature selection metric, CorrAUC, and develop a new feature selection algorithm using a wrapper technique. Their approach enhances traffic flow classification accuracy, evaluated using the NSL-KDD dataset [18] with three different ML algorithms such as RF, LR, and KNN, achieving 99%, 82%, and 98%, respectively. Babayigit et al. [26] propose the *Queried Adaptive Random Forests* (QARF) method—an online active learning-based approach that combines adaptive RF with an adaptive margin sampling strategy. This method queries a small number of instances from unlabeled traffic streams to obtain training data. Experimental evaluations using the NSL-KDD dataset [18] demonstrate that QARF achieves 98.20% accuracy.

C. Malicious Traffic Detection in IoT Networks

IoT networks have attracted significant research interest due to their lower security mechanisms compared to traditional networks [27]. For instance, Babayigit et al. [28] propose a novel approach to IoT malicious traffic detection using multiple-domain learning. They used Edge-IIoTSet [29], WUSTL-IIoT-2021 [30], and X-IIoTID [31] datasets in the proposed approach. Furthermore, they employ an autoencoder for feature-space fusion to convert all datasets into common feature space. Their hybrid deep learning model, combining CNN and GRU, achieves up to 97.68% accuracy and improvements in transfer learning scenarios. Zhu et al. [32] present the *Lightweight Knowledge Distillation Space-Time Neural Network* (LKD-STNN) model to address IoT security constraints by creating a compact model using knowledge distillation and adaptive temperature function dynamics. Their model achieves over 98% accuracy on ToN-IoT [33] and IoT-23 [34] datasets.

Huo et al. [35] introduce LightGuard, a lightweight malicious traffic detection model for IoT. LightGuard utilizes *lightweight residual block* (LRB) modules (inspired by ShuffleNetV2 [36]) and a novel ghost module for efficient feature map generation, achieving over 99.6% accuracy across diverse datasets (Edge-IIoTSet [29], USTCTFC2016 [37], ToN-IoT [33] and CIC-IoT [38] datasets) while maintaining low computational complexity. Babayigit et al. [28] propose a multiple-domain learning framework to improve the reliability and generalization of DL models for Industrial IoT (IIoT) traffic classification. Their work integrates Edge-IIoTSet [29], WUSTL-IIoT-2021 [30], and X-IIoTID [31] datasets using an autoencoder for dimensionality harmonization and a modified locally linear embedding for statistical alignment. They use a hybrid DL model that combines CNNs and GRUs along with Bayesian optimization for hyperparameter tuning. Their work achieves 97.68% accuracy, 97.70% recall, 97.67% precision, and 97.68% F1-score for binary classification and improves to 97.80% accuracy and 97.79% F1-score with transfer learning.

D. Continuous Learning for Malicious Traffic Detection

Continuous learning is an approach in which a model learns continuously by incorporating new data without retraining from scratch [39]. This is especially useful for adapting to evolving data in dynamic environments. Continuous learning is also important in cybersecurity due to the evolving nature of attacks. Xu et al. [40] propose an approach that uses *self-paced class incremental learning* (SPCIL). SPCIL leverages network traffic data to improve *class incremental learning* (CIL), a deep learning technique that integrates new malware classes while preserving recognition of prior categories. SPCIL uses a loss function that combines sparse pairwise loss with sparse loss. Their experimental results demonstrate that SPCIL effectively identifies both existing and new malware classes. Compared to other incremental learning methods, SPCIL excels in performance and efficiency, with a minimal parameter count of 8.35 million, achieving accuracy rates of 89.61%, 94.74%, and 97.21% in different test scenarios. Similarly, Ajjaj et al. [41] present an approach to detect black hole attacks, an attack on the *Ad hoc On-Demand Distance Vector* (AODV) routing protocol. They simulate realistic VANET scenarios using the *Simulation of Urban Mobility* (SUMO) [42] and the Network Simulator (NS-3) [43]. They evaluate the performance of two online incremental classifiers, *Adaptive Random Forest* (ARF) and KNN, using metrics such as accuracy, recall, precision, and F1-score, as well as training and testing time. Results demonstrate that ARF successfully classifies and detects black hole nodes in VANETs, outperforming KNN in all performance measures.

Wang et al. [44] propose an incremental learning method for small-sample data to detect malicious traffic. Their method employs

a pruning strategy to identify and remove redundant network structures, dynamically reallocating these resources based on the proposed measurement method according to the difficulty of the new class. Their approach ensures that the network can learn incrementally without overconsuming storage and computing resources. Additionally, the proposed method utilizes knowledge transfer to mitigate forgetting old classes, alleviating the burden of training large parameters with limited data. Their experimental results on multiple datasets outperform established baselines in classification accuracy while using 50% less memory. In addition, Zhao et al. [45] propose Trident, a framework for detecting fine-grained unknown encrypted traffic. Trident transforms the identification of known and new classes into multiple independent one-class learning tasks. It comprises three modules: tSieve for traffic profiling, tScissors for outlier threshold determination, and tMagnifier for clustering. Their evaluations on four popular datasets show that Trident outperforms 16 other related works.

E. Multi-Environment Malicious Traffic Detection

Malicious traffic detection in *M-En* networks remains challenging due to the heterogeneous and complex nature of traffic patterns. M-En networks, where IoT and traditional IP-based traffic coexist, create challenges for the security system. Especially, different network types exhibit distinct traffic patterns, making it challenging to develop models that can effectively capture this diversity. To address this security concern, several studies propose different approaches to securing M-En. For example, Rustam et al. [14] introduce a system that leverages the *Synthetic Data Augmentation Technique* (S-DATE) and a *Particle Swarm Optimizer* (PSO)-based *Diverse-Self Ensemble Model* (D-SEM) to enhance the detection of malicious activities across diverse network environments. Their approach is validated on a composite dataset integrating InSDN [46], UNSW-NB15 [13], and IoTID-20 [12], achieving an accuracy score of 98.9%.

Building on this foundation, Rustam et al. [47] present a detection approach that combines a newly developed M-En traffic dataset with S-DATE to mitigate data imbalance and improve model training efficiency. This approach enhances the detection of malicious traffic, achieving a detection rate of 99.1%. In advancing cybersecurity in M-En environments, Rustam et al. [11] develop ML models trained on AI-based traffic for the M-En dataset derived from benchmark datasets UNSW-NB15 [13] and IoTID-20 [12]. Their models counter both traditional and AI-based threats, achieving accuracy rates of 98.3% for binary classification and 96.8% for multi-class problems. Similarly, Indrasiri et al. [48] propose an approach to detect malicious traffic in M-En networks through ensemble learning. By merging UNSW-NB15 and IoTID-20 datasets and reducing the feature set using *Principal Component Analysis* (PCA), they develop a stacked ensemble model termed *Extra Boosting Forest* (EBF). EBF enhances detection performance, achieving accuracy scores of 98.5% and 98.4% for binary and multi-class classifications, respectively. In addition, Zukaib et al. [49] validate a framework designed for detecting cyberattacks in dynamic *Internet of Medical Things* (IoMT) networks within M-En environments. Their approach integrates Federated Learning and Meta-learning within a multi-phase architecture, achieving an accuracy of 99.82%.

F. The Gap in the Literature

Despite significant advancements in malicious traffic detection using ML across various network environments, challenges remain. Most studies focus on specific network architectures, such as IoT [35] or traditional networks [22]. There is a lack of research

TABLE I: Summary of Related Work

Ref	Year	ML	DL	Dataset	Method	Results	Purpose	Limitation
[15]	2024	-	CNN, Transfer Learning	CIC-DDoS2019, CSE-CIC-IDS2018, UNSW-NB15 and KDDCup'99	Adaptive architectures, CNNs	93.62%, 99.92%, 99.84%, and 98.99%	DDoS Detection	Works with benchmark datasets, not real-time testing.
[19]	2023	Gradient Boosting, XGBoost	-	CICDoS2019	Comparative study	GB: 99.99%, XGBoost: 99.98%, minimal false alarms	DDoS Detection	Limited to CICDoS2019 dataset
[1]	2023	-	DFNet	Self-Collected	Preference-driven, in-network enforced traffic shaping	99.93% forwarding of victim-desired traffic with minimal overhead	DDoS Defense	False classifications and lacks real network trace evaluations
[20]	2024	LR, SVM, NB, KNN, RF, EC, SVM-RF	ANN	Self-Collected	SVM-RF Ensemble	99.1%	DDoS Attack Detection	Lack of DL models evaluation.
[22]	2024	RF, ET, DT	-	UNSW-NB15, CIC-IDS-2017, CIC-IDS-2018	RO, Stacking Feature Embedding, PCA	Accuracy: 99.59%-99.99% depending on model and dataset	Network Intrusion Detection	Focuses on specific datasets, and no real-time testing.
[50]	2023	-	LSTM, BiLSTM, GRU, Deep RNN	CIRACIC-DoHBrw-2020-time series	Two-layer network classification	99% accuracy with GRU	Network Intrusion Detection	Focuses on DoH traffic and did not focus on real-time diverse traffic.
[25]	2023	RF, LR, KNN	-	NSL-KDD	CorrAUC, feature selection algorithm	99% with RF	Feature Selection for Traffic Classification	Works with an old dataset and no real-time testing.
[26]	2023	Adaptive Random Forests	-	NSL-KDD	Online active learning, adaptive margin sampling	98.20% accuracy, outperforms other state-of-the-art methods	Complex Traffic Detection	Focuses on NSL-KDD dataset
[28]	2024	-	CNN, GRU	Edge-IIoTSet, WUSTL-IIoT-2021, X-IIoTID	Multiple-domain learning, autoencoder, modified locally linear embedding	Up to 97.68% accuracy, significant improvements in transfer learning scenarios	IoT Malicious Traffic Detection	Limited to specific IoT datasets
[32]	2023	-	LKD-STNN	ToN-IoT, IoT-23	Knowledge distillation, adaptive temperature function dynamics	Over 98% accuracy	IoT Security	Focuses on ToN-IoT and IoT-23 datasets
[35]	2023	-	-	Edge-IIoTSet, USTCTFC2016, ToN-IoT and CIC IoT Dataset 2023	Lightweight residual block (LRB) modules, ghost module	Over 99.6% accuracy for CIC IoT Dataset 2023, 99.94% USTC-TFC2016, 99.93% ToN-IoT and 99.9% Edge-IIoTSet	IoT Malicious Traffic Detection	Limited to specific IoT datasets and no real-time testing.
[40]	2023	-	SPCIL	USTC-TFC2016	Self-paced class incremental learning (SPCIL)	Accuracy rates: 89.61%, 94.74%, and 97.21% in increments of 2, 4, and 5	Incremental Learning for Malicious Traffic	Focuses on small-sample datasets
[41]	2023	ARF, KNN	-	Simulated VANET scenarios	Incremental learning, Adaptive Random Forests, K-Nearest Neighbors	ARF outperforms KNN in all performance measures	Incremental Learning for VANET Security	Higher training and testing time for ARF
[44]	2023	-	-	-	Incremental learning, pruning strategy, knowledge transfer	Superior performance, 50% less memory usage	Small-sample Malicious Traffic Classification	Focuses on small-sample datasets
[45]	2024	-	-	Four popular datasets	tSieve, tScissors, tMagnifier	Significantly outperforms 16 state-of-the-art methods	Fine-grained Unknown Encrypted Traffic Detection	Customizable framework for specific scenarios
[14]	2024	-	-	InSDN, UNSW-NB15, IoTID-20	Synthetic Data Augmentation Technique (S-DATE), Particle Swarm Optimizer (PSO)	Accuracy score: 0.989	M-En Malicious Traffic Detection	Focuses on synthetic data augmentation technique
[47]	2023	-	-	M-En traffic dataset	S-DATE, new M-En traffic dataset	Detection rate: 0.991	Mitigating Data Imbalance in M-En Networks	Focuses on specific M-En traffic dataset
[11]	2024	Extra Trees	-	UNSW-NB15, IoTID-20	AI-based traffic detection	Accuracy: 0.983 for binary, 0.968 for multi-class	M-En Network Security	Focuses on benchmark datasets
[48]	2022	Extra Tree, Gradient Boosting, Random Forest	-	UNSW-NB15, IoTID-20	Ensemble learning, PCA	Accuracy: 0.985 for binary, 0.984 for multi-class	Malicious Traffic Detection in M-En Networks	Focuses on specific datasets
[49]	2024	-	-	IoMT networks	Federated Learning, Meta-learning	Accuracy: 99.82%	Cyberattack Detection in IoMT Networks	Focuses on IoMT networks

in the M-En domain, with few studies [47], [49] relying on synthetic combinations of existing benchmark datasets, as no real M-En datasets exist. This reliance on synthetic datasets may limit generalizability to real-world M-En networks. Moreover, adapting to continuously evolving threats, particularly advanced persistent threats utilizing sophisticated attack strategies, presents ongoing challenges. Some studies adopt continuous learning [44], but they

are domain-specific, focusing either on IoT or other networks. Therefore, an approach is needed to handle M-En networks in real-time and update their knowledge with continuous learning.

III. PROPOSED METHODOLOGY

We propose a real-time approach for malicious traffic detection in M-En networks using ML. In this real-time approach, we utilize

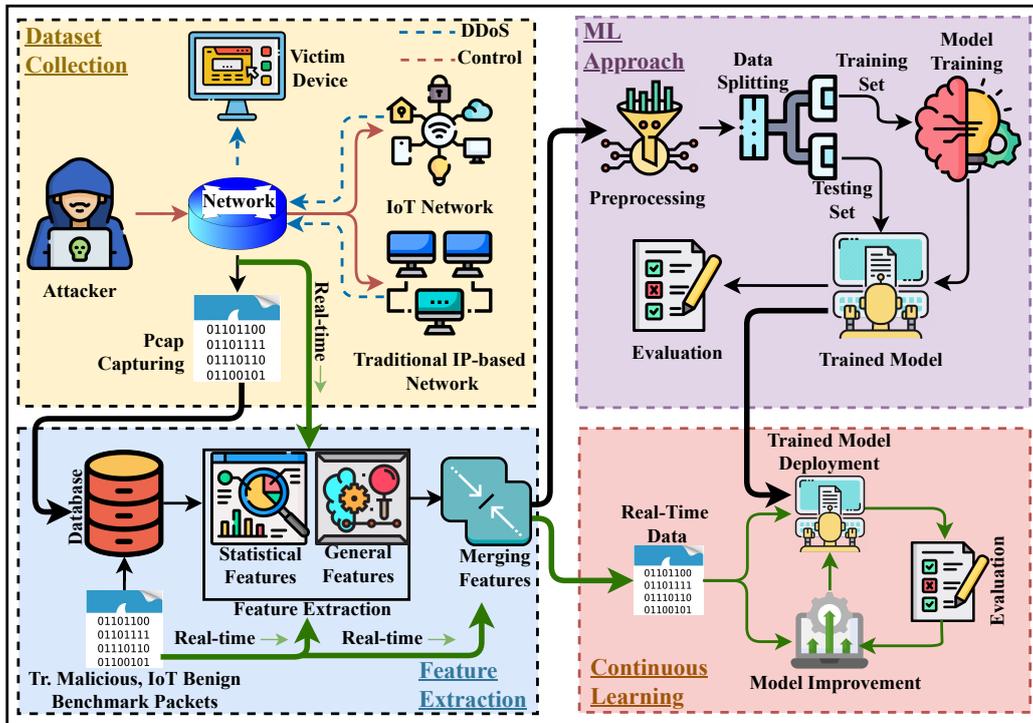


Fig. 1: Overview of the proposed framework for malicious traffic detection for M-En networks

a *continuous learning* technique that enables the ML models to continuously learn and adapt to new traffic patterns, ensuring the system remains up-to-date over time. We evaluate and validate our proposed approach in realistic scenarios by generating real-world traffic and using our models to classify this traffic in real-time.

Figure 1 shows our proposed approach and its four main phases: Dataset Collection, Feature Extraction, ML Approach, and Continuous Learning.

In the first phase (Dataset Collection), we collect a PCAP dataset (a dataset that consists of recorded data packets that have traversed in a network). We collect this dataset by leveraging DDOSHIELD-IOT [5] (a testbed that uses NS-3 and Docker containers to replicate realistic IoT environments and traffic for studying intrusion detection systems) to generate real-world malicious IoT traffic and benign traditional traffic that are supported by DDOSHIELD-IOT. Since DDOSHIELD-IOT does not support the generation of benign IoT traffic and malicious traditional traffic, we leverage existing PCAP datasets (published in the literature) that contain both of the traffic types that DDOSHIELD-IOT cannot generate. In this manner, we are able to construct a dataset that contains real IoT benign and malicious traffic as well as real traditional (non-IoT) benign and malicious traffic. We discuss our Dataset Collections phase in detail in §III-A.

In the second phase (Feature Extraction), we extract general features directly from the packets in the PCAP dataset (e.g., destination ports). We also calculate statistical features for these packets (e.g., the average number of packets received in a time window). Then, we merge these general and statistical features to form a significant feature set. These statistical and general features generate a more correlated and linearly separable feature set for malicious and benign targets to train the model accurately and efficiently. We discuss the details of the feature extraction phase in §III-B.

In the third phase (ML Approach), we start by preprocessing the merged feature set. After the preprocessing step, we split the processed feature set into training and testing sets. Then, we use the training set to train multiple ML models and subsequently test these trained models on the testing set to check their performance. We

estimate the performance of the ML models in terms of accuracy, precision, recall, F1 score, ROC curve, and computational time. We discuss the details of the ML Approach phase in III-C.

In the last phase (Continuous Learning), we deploy our trained ML models using DDOSHIELD-IOT and validate their performance in real-time. In addition, we utilize a continuous learning technique to retrain the model on new (unseen) data over time. We use continuous learning in two scenarios: with and without human (expert) involvement. We discuss the details of the Continuous Learning phase in §III-D.

A. Phase 1: Data Collection

We construct a comprehensive dataset that includes both IoT and traditional (non-IoT) traffic, encompassing benign and malicious categories. To achieve this, we employ a two-pronged approach. First, we leverage the DDOSHIELD-IOT testbed [5], which integrates NS-3 (a discrete-event network simulator) with Docker containers. This combination provides a highly realistic environment where actual binaries run in containers and communicate over a simulated network, enabling the generation and capture of real-world network traffic. We use DDOSHIELD-IOT to generate the two types of traffic it natively supports—benign traditional traffic and malicious IoT traffic. Second, to incorporate the other two required traffic categories (benign IoT traffic and malicious traditional traffic), we integrate external datasets published in the literature. Below, we discuss these two steps in detail, and then we discuss how we merge these different traffic sources in a unified manner.

- 1) **Traffic Generation Using DDOSHIELD-IOT:** DDOSHIELD-IOT natively supports the generation of certain traffic types. Specifically, it facilitates the creation of benign traditional traffic and malicious IoT traffic. For the benign traditional traffic, it generates three types of traffic, which include FTP, HTTP, and RTMP-based video streams from legitimate services within Docker containers. For malicious IoT traffic, we leverage the Mirai malware binaries hosted in the Docker environment, in which we produce IoT-based DDoS attack traffic. In particular, we

consider three classic Mirai-based attack types: ACK flood, SYN flood, and UDP flood, which reflect key threats faced by IoT devices in modern network environments.

- 2) **Integrating External PCAP Datasets:** While DDOSHIELD-IoT allows us to generate both benign traditional traffic and malicious IoT traffic, it does not natively support benign IoT traffic or malicious traditional traffic. To address these gaps, we incorporate external PCAP datasets that have been widely used and validated in other research works. This approach ensures our final dataset is both comprehensive and realistic. For the benign IoT traffic, we use the MQTT-IoT-IDS2020 dataset¹. The MQTT-IoT-IDS2020 dataset contains benign IoT network traces, focusing on IoT devices and services employing the MQTT protocol. It includes normal operations of IoT devices in a controlled environment, providing a clear baseline of legitimate IoT behavior. By integrating this dataset, we incorporate authentic IoT benign traffic that closely represents everyday device activities. For the malicious traditional traffic, we use the CIC DDoS 2019 dataset². The CIC DDoS 2019 dataset includes a variety of DDoS attacks captured in a realistic testbed environment. These attacks are carried out on traditional (non-IoT) services and include multiple DDoS vectors that target typical enterprise or data center hosts. Incorporating this dataset enhances the representativeness of our overall traffic corpus, ensuring that our MTDS can detect malicious patterns in both IoT and traditional network segments.

To combine these different traffic sources, we employ a Python script that is running in a Docker container node, which is running in promiscuous mode within the DDOSHIELD-IoT simulation. Our Python script receives both the live traffic generated by the DDOSHIELD-IoT environment and batches of packets randomly selected from the external PCAP datasets. The Python script processes incoming packets without distinction between their origins.

All traffic is labeled according to established ground truths. Traffic generated within DDOSHIELD-IoT is inherently known to be malicious or benign based on controlled initiation of attacks or normal service activity. External PCAP datasets come with predefined annotations, indicating which portions are malicious or benign. Thus, after merging, each packet (whether IoT or traditional) is labeled accordingly, ensuring clear distinctions among benign IoT, malicious IoT, benign traditional, and malicious traditional traffic.

To ensure that traffic originating from different sources—DDOSHIELD-IoT and external PCAP files—can be meaningfully compared, we apply a uniform feature extraction and aggregation methodology. First, all packets are processed through the same feature extraction pipeline (detailed in III-B), which derives both basic packet-level features (e.g., IP addresses, port numbers, protocol types, and payload sizes) and advanced statistical features computed over fixed time intervals (e.g., the average packet size, the packet rate, and the distribution of protocols observed within a given time window).

This uniformity means that regardless of whether a packet is captured from the simulated environment or sourced from an external dataset, the same set of features is extracted using identical definitions, thresholds, and time windows. By treating all traffic in this consistent manner, we eliminate biases that could arise from applying different techniques or settings to different portions of the data. As a result, the final combined dataset presents a coherent feature space, making it possible to directly compare, combine, and evaluate data from multiple origins using a single, unified analytic framework.

¹<https://paperswithcode.com/dataset/mqtt-iot-ids2020>

²<https://www.unb.ca/cic/datasets/ddos-2019.html>

B. Phase 2: Feature Extraction

After capturing traffic into PCAP files, we extracted features for training ML models. These features consist of two types: general features and statistical features. General features are directly related to traffic characteristics, while statistical features are derived from analyzing general features over time to detect patterns indicative of DDoS attacks.

The extracted general features from the network traffic data are essential for comprehensive analysis and detection of malicious activities. The *timestamp* records the precise time of packet capture, allowing for chronological analysis. *Source and destination IP addresses* (*ip_src*, *ip_dst*) identify the communication endpoints, crucial for tracing attack origins and targets. The *protocol* field specifies the transport protocol (e.g., TCP, UDP), aiding in protocol-specific analysis. *Source and destination ports* (*src_port*, *dst_port*) indicate the application-level endpoints, useful for identifying targeted services. The presence of *TCP and UDP flags* (*tcp_flag*, *udp_flag*) denotes the protocol used, while *Time to Live* (TTL) reveals the packet's hop count, indicating network topology and potential anomalies. Flags such as *ACK*, *SYN*, *FIN*, *PSH*, *URG*, and *RST* in TCP packets provide insights into the connection states and potential malicious behaviors like SYN flooding or connection resets. *Sequence and acknowledgment numbers* are vital for reconstructing TCP sessions and detecting session hijacking or manipulation. Finally, *packet size* and *payload size* metrics help in identifying unusual packet structures and potential payload-based attacks. Together, these features provide a detailed view of network traffic, enabling effective identification and mitigation of diverse cyber threats.

```
'Timestamp', 'Source', 'Destination',
'Protocol', 'SrcPort', 'DstPort',
'TCP', 'UDP', 'TTL', 'ACK',
'SYN', 'FIN', 'PSH', 'URG',
'RST', 'SequenceNumber', 'PacketSize',
'AcknowledgmentNumber', 'PayloadSize'
```

These general features provide a detailed description of each packet and are essential for identifying the characteristics of network traffic. However, these features alone are not sufficient to detect DDoS attacks effectively, as DDoS attacks rely on continuous high-frequency packet transmission [51]. Therefore, we also extract statistical features calculated over fixed time windows to capture traffic patterns and anomalies. These 24 statistical features are crucial for detecting DDoS attacks and are described below:

- **Packet Count:** The total number of packets observed in each time window, represented as $\text{PacketCount}_t = \sum_{i=1}^n \text{Packet}_i$.
- **Destination Port Entropy:** Measures the entropy of destination ports to detect scanning activities, defined as $\text{Entropy} = -\sum_{i=1}^n p_i \log(p_i)$.
- **Most Frequent Source Port:** Identifies the most common source port in a window, SourcePort_{\max} .
- **Most Frequent Destination Port:** Identifies the most common destination port in a window, $\text{DestinationPort}_{\max}$.
- **Short-lived Connections:** Counts the number of short-lived connections, $\text{ShortLivedConnections} = \sum_{i=1}^n \delta_i$, where δ_i indicates a short-lived connection.
- **Repeated Connection Attempts:** Measures repeated connection attempts, $\text{RepeatedAttempts} = \sum_{i=1}^n \alpha_i$, where α_i indicates a repeated attempt.
- **Network Scanning Activity:** Counts instances of SYN flags without ACK flags, $\text{Scanning} = \sum_{i=1}^n (\text{SYN} - \text{ACK})$.
- **Flow Rate:** Calculated as packets per second, $\text{FlowRate} = \frac{\text{TotalPackets}}{\text{TimeInterval}}$.

- **Source Entropy:** Entropy of source addresses, $SourceEntropy = -\sum_{i=1}^n q_i \log(q_i)$, where q_i is the probability distribution of source addresses.
- **Connection Errors (RST flag):** Counts instances of RST flags, $RSTCount = \sum_{i=1}^n RST_i$.
- **Most Frequent Packet Size Frequency:** Identifies the most common packet size, $PacketSize_{max}$.
- **Abnormal Size Frequency:** Counts packets exceeding a size threshold, $AbnormalSizeFrequency = \sum_{i=1}^n Packet_i$ if $size_i > Threshold$.
- **Sequence Number Variance:** Variance in sequence numbers, $Var(SequenceNumber)$.
- **Average Packet Number:** Average packets per interval, $AvgPackets = \frac{TotalPackets}{TimeIntervals}$.
- **SYN Frequency:** Frequency of SYN flags, $SYNFrequency = \frac{TotalSYN}{TimeInterval}$.
- **ACK Frequency:** Frequency of ACK flags, $ACKFrequency = \frac{TotalACK}{TimeInterval}$.
- **TCP Frequency:** Proportion of TCP packets, $TCPFrequency = \frac{TotalTCP}{TotalPackets}$.
- **UDP Frequency:** Proportion of UDP packets, $UDPFrequency = \frac{TotalUDP}{TotalPackets}$.
- **Most Frequent Protocol:** Most used protocol, $Protocol_{max}$.
- **Packet Size Variability:** Variance in packet sizes, $Var(PacketSize)$.
- **Most Frequent Payload Size:** Most common payload size, $PayloadSize_{max}$.
- **Average Payload Size:** Mean payload size, $AvgPayloadSize = \frac{TotalPayload}{TotalPackets}$.
- **Packet Size Standard Deviation:** Standard deviation of packet sizes, $StdDev(PacketSize)$.
- **Average Packet Size:** Mean size of packets within a window, $AvgPacketSize = \frac{TotalPacketSize}{TotalPackets}$.

In our approach, we used specific thresholds and parameters to calculate these statistical features. The processing interval was set to 1 second, which is the time interval used to aggregate packets. An abnormal size threshold was defined as 1500 bytes to identify abnormal packet sizes. The port frequency threshold was set to 5 to determine frequently used ports. Additionally, a short-lived connection threshold was defined as less than 5 packets to identify short-lived connections. These features are designed to capture various aspects of network traffic and detect deviations from normal patterns that are characteristic of DDoS attacks. By combining general features and statistical features, we can effectively distinguish between normal traffic and DDoS traffic, enhancing the detection capabilities of our system. The comprehensive feature set is illustrated in Figure 2.

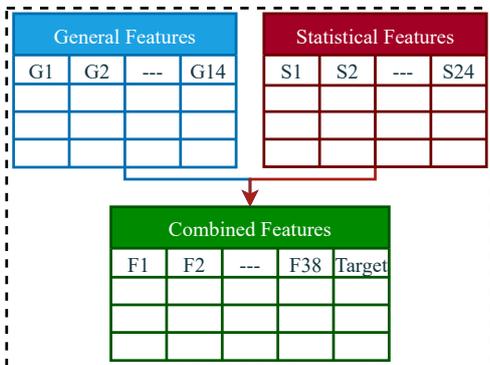


Fig. 2: Combining statistical and general features approach

Figure 3 shows the feature space for both general and statistical features, highlighting the importance of periodic statistical feature calculation. In this visualization approach, we use PCA to reduce

the dataset’s dimensionality to three principal components and then visualize these components in a 3D scatter plot. The target labels are used to differentiate between classes, with each class represented by a unique color. It is evident that with only general features, the samples overlap, making it difficult for the model to learn distinguishable patterns. However, with statistical features, the samples are separable, aiding the model in learning distinct patterns.

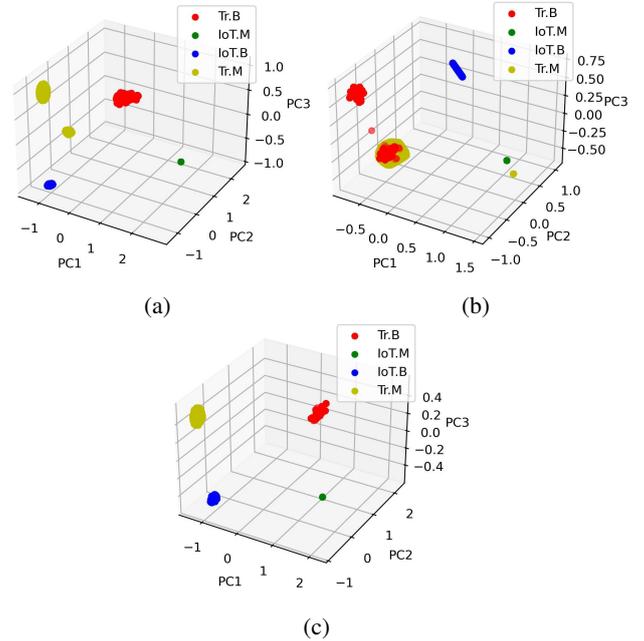


Fig. 3: Feature Space (a) Combined Features, (b) General Features, and (c) Statistical Features

C. Phase 3: ML Approach

In the ML approach, we deployed several state-of-the-art methods for malicious traffic detection. We trained two models, M1 and M2. M1 is a lightweight model trained on selected samples after preprocessing while M2 is a complex model trained on a huge dataset as shown in Figure 4. This M1 model directly deals with live traffic, and if it fails, the decision is passed to M2, which then helps to learn new data where M1 failed.

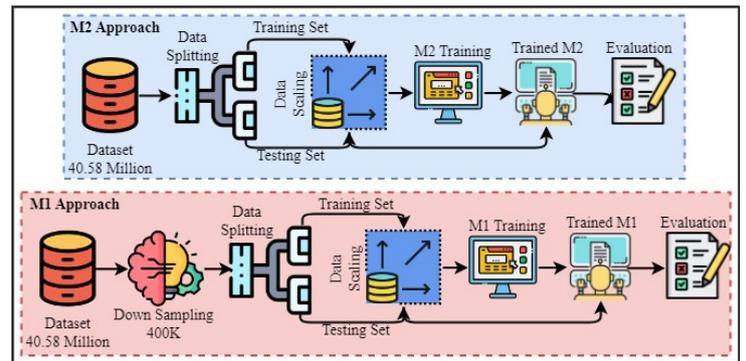


Fig. 4: ML models training approaches

M1: In the M1 training approach, we first selected a small sample of data by choosing 100,000 samples from each class, totaling 400,000 samples. This small dataset helps us obtain a lightweight model. Initially, we randomly selected 100,000 samples and trained different models. Then, we selected the best 100,000 samples from each class using the K-Means clustering approach [52]. K-Means clustering was applied to the standardized features, with an appropriate number of clusters chosen based on the dataset’s characteristics. For each sample, the distance to the nearest cluster

center was calculated, reflecting how well each sample represented its cluster. By selecting the samples with the smallest distances to their respective cluster centers, this method ensured that the most representative samples for each class were chosen. This process was repeated for each class in the dataset, and the selected top samples from each class were compiled into a new data frame. This approach effectively reduced the dataset size while maintaining the diversity and representativeness of the original data, making it suitable for training ML models or further analysis. After selecting the important samples, we applied data scaling using the MinMax method. After that, we split the dataset with a 75% and 25% ratio, where 75% was used to train the model and 25% to test the model. We trained four models, which are described below such as stochastic *gradient descent classifier* (SGDC), perceptron classifier, *multinomial naive Bayes* (MNB), and *Bernoulli naive Bayes* (BNB). We used the `partial_fit` function to train the model in the M1 approach as we used the M1 method to retrain during continuous learning. In the end, we evaluated the performance in terms of accuracy, precision, recall, and F1 score. We also used a 10-fold cross-validation approach to evaluate the performance.

We deploy the four models in the M1 approach using their best hyperparameter settings, which were identified through a combination of literature review [11], [14], [53] and tuning within specific ranges and values using a trial-and-error approach. Table II illustrates the hyperparameter ranges and values used. Some models, such as BNB and MNB, were deployed with their default hyperparameter settings because these models are known to perform well with defaults due to their simplicity and lack of sensitivity to extensive hyperparameter tuning [54].

Below, we present a detailed discussion of each model, emphasizing their key characteristics and providing a rationale for their selection in the M-En malicious traffic detection task

TABLE II: Hyperparameter settings for the M1 models

Model	Hyperparameter	Value	Tuning Range
SGDC	loss	log_loss	-
	max_iter	1000	[200 to 1500]
	tol	1e-3	-
	random_state	42	[0, 42]
Perceptron	max_iter	1000	[200 to 1500]
	tol	1e-3	-
	random_state	42	[0, 42]
MNB	Default	Default	-
BNB	Default	Default	-

1) *SGDC*: It is a linear classifier that optimizes a linear model using stochastic gradient descent. We use it for malicious traffic detection in M-EN because it is efficient for large-scale and sparse datasets. It also supports continuous learning, which makes it suitable for our study [55]. It supports various loss functions; however, we use log loss, which helps us work with prediction probabilities to measure model confidence. The SGDC models the decision function as a linear combination of the input features as:

$$z = \mathbf{w} \cdot \mathbf{x} + b = w_1x_1 + w_2x_2 + \dots + w_nx_n + b \quad (1)$$

where, $\mathbf{w} = [w_1, w_2, \dots, w_n]$ is the weight vector, b is the bias term, \mathbf{x} is the input vector. Depending on the chosen loss function (log in our case), the SDGC optimizes the model using the gradient of the loss function, which can be described as:

$$L(\mathbf{w}, b) = -\log\left(\frac{1}{1 + \exp(-y(\mathbf{w} \cdot \mathbf{x} + b))}\right) \quad (2)$$

Then SGDC updates weights iteratively, which can be described as:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla L(\mathbf{w}, b, \mathbf{x}_i, y_i) \quad (3)$$

$$b \leftarrow b - \eta \frac{\partial L(\mathbf{w}, b, \mathbf{x}_i, y_i)}{\partial b} \quad (4)$$

where, η is the learning rate, ∇L is the gradient of the loss function.

2) *Perceptron*: It is a linear classifier that makes predictions based on a simple threshold function [56]. It is used for binary classification tasks and can be extended to multiclass classification using the *one-vs-rest* (OvR) scheme, as in our M-En malicious traffic detection. It is also effective when data is linearly separable, which is very suitable for our case, as our dataset is highly linearly separable, as shown in Figure 3. Perceptron in sci-kit-learn also provides a continuous learning function, so we use it as M1. The Perceptron classifier aims to find a hyperplane that separates the classes. Given an input vector $\mathbf{x} = [x_1, x_2, \dots, x_n]$, the Perceptron computes the output y using:

$$z = \mathbf{w} \cdot \mathbf{x} + b = w_1x_1 + w_2x_2 + \dots + w_nx_n + b \quad (5)$$

where, $\mathbf{w} = [w_1, w_2, \dots, w_n]$ is the weight vector, b is the bias term, \mathbf{x} is the input vector. The output y is then determined by applying the activation function:

$$y = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases} \quad (6)$$

3) *MNB*: It is suitable for large datasets where fast computation is needed, as in our case [57]. It also provides the *'partial_fit'* function for continuous learning, which we use in our approach as the M1 model in comparison with others [58]. It estimates the probability of a class given the feature values by combining the prior probability of the class and the likelihood of the observed features given the class using Bayes' theorem:

$$P(C_k|\mathbf{x}) = \frac{P(C_k) \cdot P(\mathbf{x}|C_k)}{P(\mathbf{x})} \quad (7)$$

where, $P(C_k|\mathbf{x})$ is the posterior probability of class C_k , $P(C_k)$ is the prior probability of class C_k , $P(\mathbf{x}|C_k)$ is the likelihood of \mathbf{x} given C_k , $P(\mathbf{x})$ is the marginal likelihood of \mathbf{x} . The likelihood is estimated based on the frequency of features:

$$P(\mathbf{x}|C_k) = \prod_{i=1}^n P(x_i|C_k)^{x_i} \quad (8)$$

where, $P(x_i|C_k)$ is the probability of feature x_i in C_k , x_i is the count of feature i in \mathbf{x} . Parameters are estimated using Laplace smoothing:

$$P(x_i|C_k) = \frac{N_{C_k, x_i} + \alpha}{N_{C_k} + \alpha n} \quad (9)$$

where, N_{C_k, x_i} is the count of x_i in C_k , N_{C_k} is the total count of features in C_k , α is the smoothing parameter. Class prediction is made by selecting the class with the highest posterior probability:

$$\hat{C} = \arg \max_{C_k} P(C_k) \prod_{i=1}^n P(x_i|C_k)^{x_i} \quad (10)$$

4) *GNB*: We use it because network traffic features such as packet sizes, inter-arrival times, and other statistical metrics are continuous. GNB is well-suited for continuous data as it assumes that the features follow a Gaussian (normal) distribution [59]. It also uses Bayes' theorem like MNB:

$$P(C_k|\mathbf{x}) = \frac{P(C_k) \cdot P(\mathbf{x}|C_k)}{P(\mathbf{x})} \quad (11)$$

where, $P(C_k|\mathbf{x})$ is the posterior probability of class C_k , $P(C_k)$ is the prior probability of class C_k , $P(\mathbf{x}|C_k)$ is the likelihood of \mathbf{x} given C_k , $P(\mathbf{x})$ is the marginal likelihood of \mathbf{x} . In the GNB model, the likelihood is assumed to be normally distributed:

$$P(x_i|C_k) = \frac{1}{\sqrt{2\pi\sigma_{C_k,i}^2}} \exp\left(-\frac{(x_i - \mu_{C_k,i})^2}{2\sigma_{C_k,i}^2}\right) \quad (12)$$

where, $\mu_{C_k,i}$ is the mean of x_i in C_k , $\sigma_{C_k,i}^2$ is the variance of x_i in C_k . Parameters are estimated using maximum likelihood estimation (MLE):

$$\mu_{C_k,i} = \frac{1}{N_{C_k}} \sum_{j=1}^{N_{C_k}} x_{j,i} \quad (13)$$

$$\sigma_{C_k,i}^2 = \frac{1}{N_{C_k}} \sum_{j=1}^{N_{C_k}} (x_{j,i} - \mu_{C_k,i})^2 \quad (14)$$

where, N_{C_k} is the number of samples in C_k , $x_{j,i}$ is the value of x_i for the j -th sample in C_k . Class prediction is made by selecting the class with the highest posterior probability:

$$\hat{C} = \arg \max_{C_k} P(C_k) \prod_{i=1}^n P(x_i|C_k) \quad (15)$$

M2: In the M2 training approach, we trained models on the full dataset, which consists of 40 million samples. We applied data scaling and then split the dataset with a 75% and 25% ratio, where 75% was used to train the model and 25% to test the model. We trained several models such as MLP [60], LR [61], SGDC [62], BNB [63], GNB [64], *random forest* (RF) [65], *AdaBoost* (ADA) [66], *support vector classifier* (SVC) [67], and *k-nearest neighbors* (KNN) [68]. We evaluated the performance of these models in terms of accuracy, precision, recall, and F1 score. After training both M1 and M2 models, we saved the models using the pickle library and stored them with a .pkl extension to use them during real-time testing. Similar to the M1 approach, we also deploy the M2 approach models using their best hyperparameter settings, identified through a combination of literature review and tuning within specific ranges and values using an iterative experimentation approach [10], [53], [69]. Table III illustrates the hyperparameter ranges and values used. Other models, such as BNB, GNB, and MNB, were deployed with their default hyperparameter settings.

TABLE III: Hyperparameter settings for the M2 models

Model	Hyperparameter	Value	Tuning Range
LR	max_iter	1000	[200 to 1500]
	random_state	42	[0, 42]
SGDC	loss	log	-
	max_iter	1000	[200 to 1500]
	tol	1e-3	-
	random_state	42	[0, 42]
Perceptron	max_iter	1000	[200 to 1500]
	tol	1e-3	-
	random_state	42	[0, 42]
RF	n_estimators	100	[10 to 300]
	random_state	42	[0, 42]
	max_depth	80	[2 to 100]
ADA	n_estimators	100	[10 to 300]
	random_state	42	[0, 42]
	max_depth	80	[2 to 100]
SVC	probability	True	-
	random_state	42	[0, 42]
KNN	n_neighbors	5	[3 to 30]
MLP	hidden_layer_sizes	(100,)	[50 to 500]
	max_iter	1000	[200 to 1500]
	random_state	42	[0, 42]
MNB	Default	Default	-
BNB	Default	Default	-
GNB	Default	Default	-

D. Phase 4: MULTI-LF with Continuous Learning

In MULTI-LF, we use two models: M1 (a lightweight model trained on a small sample of the dataset) and M2 (a complex model trained on the full dataset). Initially, data is fed into the M1 model, which predicts the traffic label. If M1's prediction confidence is 100%, the model proceeds to the next sample for prediction or ends the process. If M1's confidence is below 100%, the control shifts to M2 and the real-time traffic is passed to M2.

M2 then makes its prediction. If M2's confidence is above 90%, M1 is retrained under a continuous learning mechanism using the predicted label from M2. If M2's confidence is also below 90%, the control is handed over to a human expert for interaction. The expert monitors the specific traffic and predicts the label. Then, M1 is retrained using the new traffic data and the label provided by the human expert. Similarly, human interaction is maintained with M2 to update it periodically. This process ensures continuous monitoring and improvement of both models' performance as shown in Figure 5.

Algorithm 1 describes the continuous learning process. Here, D represents the dataset, and CT denotes the confidence threshold set at 0.9. PL refers to the predicted label's output by the models, while T indicates the incoming real-time traffic. The algorithm employs two models: $M1$, a lightweight model trained on a small sample (SS) of the dataset for quick initial predictions, and $M2$, a complex model trained on the full dataset of 40.58 million instances for secondary predictions. During prediction, S represents the current traffic packet. $p1$ and $c1$ are the prediction and confidence from $M1$, respectively. If $c1 > CT$, the prediction ($p1$) is accepted as the output (O). If not, the sample is passed to $M2$ for prediction. $p2$ and $c2$ are the prediction and confidence from $M2$. If $c2 > CT$, $M1$ is retrained incrementally with S and $p2$. If $c2 \leq CT$, a human expert provides a prediction (p_h), and both models are retrained with S and p_h . This process ensures continuous model improvement and adaptation.

Algorithm 1 Continuous Learning with M1 and M2 Models

```

1: Input:  $D$ ,  $CT = 0.9$ 
2: Output:  $PL$  for incoming  $T$ 
3: Initialize: Train  $M1$  on  $SS$  of 400K instances using K-Means clustering; Train  $M2$  on the full dataset of 40.58M instances
4: while  $T$  exists do
5:    $S = \text{next\_sample}(T)$ 
6:    $p1, c1 = M1.\text{predict}(S)$ 
7:   if  $c1 > CT$  then
8:      $O = p1$ 
9:   else
10:     $p2, c2 = M2.\text{predict}(S)$ 
11:    if  $c2 > CT$  then
12:       $O = p2$ 
13:       $M1.\text{retrain}(S, p2)$ 
14:    else
15:       $p_h = \text{get\_human\_label}(S)$ 
16:       $O = p_h$ 
17:       $M1.\text{retrain}(S, p_h)$ 
18:       $M2.\text{retrain}(S, p_h)$        $\triangleright$  Periodic human updates
19:    end if
20:  end if
21:  Store or utilize  $O$ 
22: end while

```

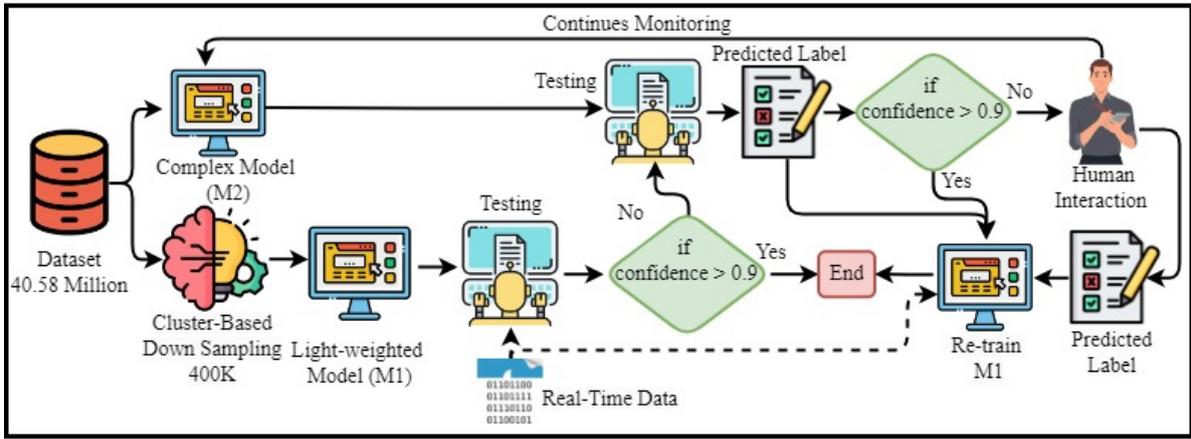


Fig. 5: MULTI-LF Flow Diagram

IV. RESULTS & DISCUSSION

In our experimentation, the evaluation and training of models were conducted on a Core i7 11th generation machine with a Windows operating system. The system has 64GB of RAM and a 1 TB SSD. We use a VMWare workstation to run Ubuntu 24.04, where we run all experiments. We used the scikit-learn library for the implementation of models and NS3 version 3.36.1.

We evaluated all models in terms of accuracy, precision, recall, F1 score, computational time, and memory usage. Accuracy is defined as the proportion of correctly predicted samples to the total number of samples, calculated as follows:

$$Acc. = \frac{TP + TN}{TP + TN + FP + FN} \quad (16)$$

In the context of malicious attack detection, precision is the ratio of correctly predicted malicious traffic (true positives) to the total traffic instances predicted as malicious (true positives plus false positives). It measures how many of the detected malicious activities are actually malicious. It is calculated as follows:

$$Precision = \frac{TP}{TP + FP} \quad (17)$$

Recall, or sensitivity, in malicious attack detection, is the ratio of correctly predicted malicious traffic (true positives) to all actual malicious traffic instances (true positives plus false negatives). It measures how many of the actual malicious activities are detected by the model. It is calculated as follows:

$$Recall = \frac{TP}{TP + FN} \quad (18)$$

The F1 Score is the harmonic mean of precision and recall, providing a balance between the two metrics, especially useful in scenarios with imbalanced datasets. Malicious attack detection gives a single score that represents the balance between precision and recall. It is calculated as follows:

$$F1Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (19)$$

A. Lightweight-M1 Models Results Used for Continuous Learning

Table IV presents the results for the ML models, referred to as M1, which were trained on a small sample consisting of 400,000 instances. We used the partial fit method to train four models, with and without min-max scaling. Without scaling, the performance of the models was poor, except for BNB, which achieved an accuracy score of 1.000, and MNB, which reached 0.972. In contrast, SGDC and Perceptron completely failed. This is because

SGDC and Perceptron are linear models that are sensitive to the scale of features. When features have different scales, the gradient descent optimization process becomes inefficient, leading to poor convergence. The models struggle to learn effectively from unscaled data, resulting in poor performance across most classes.

On the other hand, the performance of all models with scaling was excellent across all evaluation metrics, with all models achieving scores of 1.000. Scaling ensures that all features contribute proportionately to the model, enhancing the training efficiency and effectiveness of SGDC and Perceptron classifiers. It normalizes the feature space, resulting in more balanced weight updates and better model performance. For instance, if packet sizes range from 0 to 10,000 and another feature like traffic rate ranges from 0 to 1, the larger feature disproportionately influences the model's parameters, leading to suboptimal performance. Scaling mitigates this problem by normalizing the features. Similarly, MNB and BNB models assume feature independence and use probabilities based on feature occurrences. Without scaling, the probability estimates can become skewed, affecting the model's ability to correctly classify instances. After scaling, these models show improved performance due to the normalized feature space.

Table V presents the performance of various ML models, referred to as M1, trained on a 400,000-sample dataset using 10-fold cross-validation. Each model was evaluated with and without min-max scaling. Without scaling, the SGDC and Perceptron models performed poorly, achieving an accuracy of only 0.25 with a *standard deviation* (Std) of 0.0. This suboptimal performance is due to the models' sensitivity to feature scale; the gradient descent optimization process becomes inefficient when features have different scales, leading to poor convergence and low precision (0.0625 ± 0.0), recall (0.25 ± 0.0), and F1 scores (0.1 ± 0.0). Without scaling, even though the Std is low, the accuracy and other evaluation metrics are also low.

In contrast, with min-max scaling, both SGDC and Perceptron achieved perfect accuracy scores of 1.0 with a Std of 0.0 across all metrics. Scaling ensures that all features contribute proportionately, normalizing the feature space and resulting in balanced weight updates and better model performance. For instance, features like packet size and traffic rate need normalization to prevent one from disproportionately influencing the model parameters. Similarly, the MNB and BNB models also showed significant improvements with scaling. Without scaling, MNB achieved a mean accuracy of 0.9719 with a Std of 0.0328, while BNB reached a mean accuracy of 0.9992 with a Std of 0.0023. After scaling, both models achieved perfect scores (1.0 ± 0.0). The results show the significance of BNB in both cases, with and without scaling, so we use it as the M1 model in our proposed approach.

TABLE IV: Results for M1 Models using M-En Data

With Scaling						Without Scaling				
Model	Accuracy	Class	Precision	Recall	F1-score	Accuracy	Class	Precision	Recall	F1-score
SGDC	1.000	IoT.B	1.00	1.00	1.00	0.250	IoT.B	0.25	1.00	0.40
		IoT.M	1.00	1.00	1.00		IoT.M	0.00	0.00	0.00
		Tr.B	1.00	1.00	1.00		Tr.B	0.00	0.00	0.00
		Tr.M	1.00	1.00	1.00		Tr.M	0.00	0.00	0.00
		Macro Avg.	1.00	1.00	1.00		macro avg	0.06	0.25	0.10
Perceptron	1.000	IoT.B	1.00	1.00	1.00	0.250	IoT.B	0.00	0.00	0.00
		IoT.M	1.00	1.00	1.00		IoT.M	0.00	0.00	0.00
		Tr.B	1.00	1.00	1.00		Tr.B	0.25	1.00	0.40
		Tr.M	1.00	1.00	1.00		Tr.M	0.00	0.00	0.00
		Macro Avg.	1.00	1.00	1.00		macro avg	0.06	0.25	0.10
MNB	1.000	IoT.B	1.00	1.00	1.00	0.972	IoT.B	0.99	1.00	1.00
		IoT.M	1.00	1.00	1.00		IoT.M	1.00	1.00	1.00
		Tr.B	1.00	1.00	1.00		Tr.B	1.00	0.90	0.95
		Tr.M	1.00	1.00	1.00		Tr.M	0.91	0.99	0.95
		Macro Avg.	1.00	1.00	1.00		macro avg	0.97	0.97	0.97
BNB	1.000	IoT.B	1.00	1.00	1.00	1.000	IoT.B	1.00	1.00	1.00
		IoT.M	1.00	1.00	1.00		IoT.M	1.00	1.00	1.00
		Tr.B	1.00	1.00	1.00		Tr.B	1.00	1.00	1.00
		Tr.M	1.00	1.00	1.00		Tr.M	1.00	1.00	1.00
		Macro Avg.	1.00	1.00	1.00		macro avg	1.00	1.00	1.00

TABLE V: K-Fold Cross Validation Results for M1 Models using M-En Data

Model	Metric	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10	Mean \pm Std
SGDC + Scaling	Accuracy	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0 \pm 0.0
	Precision	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0 \pm 0.0
	Recall	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0 \pm 0.0
	F1-score	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0 \pm 0.0
SGDC	Accuracy	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25 \pm 0.0
	Precision	0.0625	0.0625	0.0625	0.0625	0.0625	0.0625	0.0625	0.0625	0.0625	0.0625	0.0625 \pm 0.0
	Recall	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25 \pm 0.0
Perceptron + Scaling	Accuracy	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0 \pm 0.0
	Precision	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0 \pm 0.0
	Recall	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0 \pm 0.0
	F1-score	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0 \pm 0.0
Perceptron	Accuracy	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25 \pm 0.0
	Precision	0.0625	0.0625	0.0625	0.0625	0.0625	0.0625	0.0625	0.0625	0.0625	0.0625	0.0625 \pm 0.0
	Recall	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25 \pm 0.0
	F1-score	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1 \pm 0.0
MNB + Scaling	Accuracy	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0 \pm 0.0
	Precision	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0 \pm 0.0
	Recall	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0 \pm 0.0
	F1-score	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0 \pm 0.0
MNB	Accuracy	0.9922	0.9311	0.9736	0.9018	0.9991	0.9995	0.9990	0.9440	0.9978	0.9815	0.9719 \pm 0.0328
	Precision	0.9924	0.9453	0.9759	0.9289	0.9991	0.9995	0.9990	0.9539	0.9978	0.9825	0.9774 \pm 0.0246
	Recall	0.9922	0.9311	0.9736	0.9018	0.9991	0.9995	0.9990	0.9440	0.9978	0.9815	0.9719 \pm 0.0328
	F1-score	0.9922	0.9299	0.9735	0.8979	0.9991	0.9994	0.9990	0.9433	0.9977	0.9814	0.9714 \pm 0.0339
BNB + Scaling	Accuracy	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0 \pm 0.0
	Precision	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0 \pm 0.0
	Recall	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0 \pm 0.0
	F1-score	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0 \pm 0.0
BNB	Accuracy	0.9923	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.9992 \pm 0.0023
	Precision	0.9925	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.9993 \pm 0.0022
	Recall	0.9923	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.9992 \pm 0.0023
	F1-score	0.9923	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.9992 \pm 0.0023

B. Complex-M2 Models Results

The performance of M2 models is shown in Table VI. These results are based on a full dataset consisting of approximately 40 million samples, and we experimented only after data scaling because, in the M1 case, we obtained the best results with data scaling. The performance of all models is significant, with RF achieving a perfect 1.000 accuracy score and outperforming all other evaluation metrics. RF manages many features and determines their importance, which is particularly beneficial in network traffic data that often contains diverse and numerous features. By focusing on the most relevant features, RF enhances predictive accuracy for identifying malicious traffic in our M-En network. Furthermore, RF scales well with large datasets, as in our case, and handles

extensive data efficiently, a common requirement in network traffic analysis. While BNB performed poorly with an accuracy score of 0.8316, all models took considerable time to train but achieved significant results. Our proposed approach used RF as M2 because of its significant performance.

The performance metrics for the M2 models in terms of *correct prediction* (CP), *wrong prediction* (WP), and error rate are presented in Table VII. RF model demonstrated the most significant performance with the highest CP of 10,146,052 and the lowest error rate of 0.0000, which justifies its selection as M2 in our proposed approach. LR and Perceptron models both performed admirably, with error rates of 0.0005, indicating their reliability in malicious traffic detection. The GNB and ADA models also showed strong

TABLE VI: Results for M2 Models using M-En Data

Model	Accuracy	Class	Precision	Recall	F1-score
LR	0.9995	IoT.B	1.00	1.00	1.00
		IoT.M	1.00	1.00	1.00
		Tr.B	1.00	1.00	1.00
		Tr.M	1.00	1.00	1.00
		Macro Avg	1.00	1.00	1.00
SGDC	0.9982	IoT.B	1.00	1.00	1.00
		IoT.M	1.00	1.00	1.00
		Tr.B	1.00	0.99	0.99
		Tr.M	1.00	1.00	1.00
		Macro Avg	1.00	1.00	1.00
Perceptron	0.9995	IoT.B	1.00	1.00	1.00
		IoT.M	1.00	1.00	1.00
		Tr.B	1.00	1.00	1.00
		Tr.M	1.00	1.00	1.00
		Macro Avg	1.00	1.00	1.00
MNB	0.9943	IoT.B	1.00	1.00	1.00
		IoT.M	0.99	0.99	0.99
		Tr.B	0.99	0.99	0.99
		Tr.M	0.99	1.00	0.99
		Macro Avg	0.99	0.99	0.99
BNB	0.8316	IoT.B	0.82	1.00	0.90
		IoT.M	0.99	0.74	0.85
		Tr.B	0.80	1.00	0.89
		Tr.M	0.94	0.21	0.34
		Macro Avg	0.89	0.73	0.74
GNB	0.9984	IoT.B	1.00	1.00	1.00
		IoT.M	1.00	1.00	1.00
		Tr.B	1.00	0.99	1.00
		Tr.M	1.00	1.00	1.00
		Macro Avg	1.00	1.00	1.00
RF	1.0000	IoT.B	1.00	1.00	1.00
		IoT.M	1.00	1.00	1.00
		Tr.B	1.00	1.00	1.00
		Tr.M	1.00	1.00	1.00
		Macro Avg	1.00	1.00	1.00
ADA	0.9987	IoT.B	1.00	1.00	1.00
		IoT.M	1.00	0.99	0.99
		Tr.B	0.99	0.99	0.99
		Tr.M	1.00	1.00	1.00
		Macro Avg	1.00	1.00	1.00

performance with error rates of 0.0016 and 0.0013, respectively. On the other hand, the BNB model had a significantly higher error rate of 0.1684, indicating its limited effectiveness in this context. The MNB model also showed a relatively higher error rate of 0.0057 compared to other models. These results highlight the robustness of the RF model and the varying degrees of effectiveness among different ML models in detecting malicious traffic.

TABLE VII: M2 Performance Metrics

Model	CP	WP	Error Rate
LR	10140676	5395	0.0005
SGDC	10128182	17889	0.0018
Perceptron	10140625	5446	0.0005
MNB	10088644	57427	0.0057
BNB	8437084	1708987	0.1684
GNB	10130116	15955	0.0016
RF	10146052	19	0.0000
ADA	10132709	13362	0.0013

Table VIII shows the computational time required for the training and testing of models used at the M1 and M2 levels, measured in seconds. The M2 models were trained on a larger dataset and thus exhibited higher computational times compared to the M1 models. In the M2 category, LR and ADA took significantly longer, with times of 13,815.6875 seconds and 14,106.234375 seconds, respectively. RF also required considerable time, 12,841.046875 seconds, reflecting its complex ensemble nature. Comparatively, models like SGDC, Perceptron, MNB, and BNB showed much lower computational times, highlighting their relative efficiency for the same tasks. However, RF is significant in terms of accuracy and also average computational cost, so we chose it for the proposed

approach from M2 and BNB from M1.

TABLE VIII: M1 & M2 Computational Time

Model	M1	M2
LR	-	138.68
SGDC	1.17	325.17
Perceptron	0.93	253.45
MNB	1.12	113.29
BNB	1.17	124.85
GNB	-	71.0
RF	-	12841.04
ADA	-	14106.23

C. Results in Real-Time Environment

In this section, we present the results of MULTI-LF in real-time scenarios. For real-time testing, we used NS-3 simulation to generate sample data. The data was collected centrally and then passed to our traffic analyzer, the MTDS framework. We conducted experiments under different scenarios to evaluate the framework's performance. We collected packets over specific time windows and then passed the batch of packets to the framework to determine the accuracy. This process was repeated over several iterations, and the average scores were reported.

We select the best models from online testing, evaluate their performance, and compare them across different scenarios. *Scenario 1*: M1 is used for attack detection without continuous learning capabilities and evaluated with DDoS and benign traffic. *Scenario 2*: M1 is used for attack detection with continuous learning capabilities. *Scenario 3*: M2 is used independently for testing. *Scenario 4*: M1 and M2 are deployed together with MULTI-LF without human involvement. *Scenario 5*: M1 and M2 are deployed with MULTI-LF and human involvement. The implementation of Scenarios 4 and 5 in MULTI-LF is illustrated in Figure 6.

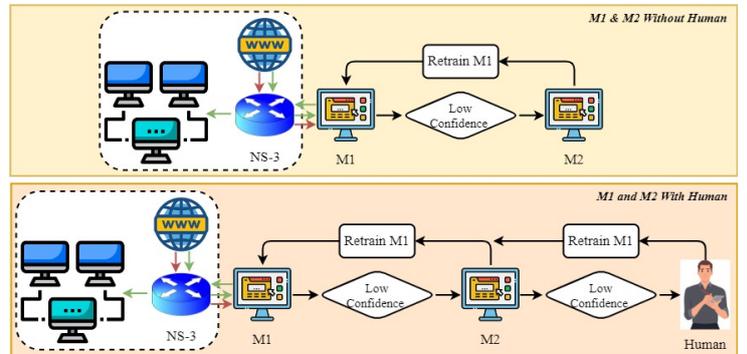


Fig. 6: Scenario 4 & 5 visualization for easy understanding

Table IX provides an evaluation of *Scenario 1*, where the M1 model is used without continuous learning for attack detection. KNN demonstrates the highest accuracy (0.688) but at the cost of substantial resource usage, including a large MS (90,236 KB), high memory consumption (187.741 MB), and significant CPU utilization (58.34%). In contrast, models like LR show a much smaller size (2 KB), lower MU (122.3 MB), and minimal CPU impact (45.61%) with a faster prediction time (0.0007 seconds), though with reduced accuracy (0.536). This highlights the trade-off between model complexity and computational efficiency, making KNN suitable for accuracy-critical scenarios, while lighter models like LR are better for resource-constrained environments.

Scenario 2 evaluates the M1 model with continuous learning capabilities, showcasing improvements in both efficiency and performance as shown in Table X. Among the models tested, Perceptron achieves the highest accuracy (0.704) while maintaining a minimal MS (3 KB) and low prediction time (0.0007 seconds).

TABLE IX: **Scenario 1:** M1 Without Continuous Learning

Model	ADA	GNB	KNN	LR	RF
MS (KB)	57	4	90236	2	103
Accuracy	0.32	0.422	0.688	0.536	0.661
PT (S)	0.071	0.002	0.723	0.0007	0.032
CPU (%)	52.91	49.12	58.34	45.61	51.29
MU (MB)	122.467	121.571	187.741	122.3	112.81

This demonstrates the model’s ability to quickly adapt to new data without a significant increase in computational costs. Other models like MNB and SGDC also display competitive accuracies (0.594 and 0.581, respectively) and maintain a low memory footprint (around 117-120 MB). These results highlight the effectiveness of continuous learning in maintaining high accuracy while optimizing resource usage for real-time scenarios as it achieved 0.704 high accuracy as compared to *Scenario 1* which has 0.688 accuracy.

TABLE X: **Scenario 2:** M1 With Continuous Learning

Model	BNB	MNB	Perceptron	SGDC
MS (KB)	4	4	3	3
Accuracy	0.575	0.594	0.704	0.581
PT (S)	0.001	0.001	0.0007	0.001
CPU (%)	49.69	50.01	55.52	51.74
MU (MB)	110.248	117.812	124.405	120.932

In *Scenario 3*, the performance of M2 is evaluated without continuous learning across multiple ML models, as shown in Table XI. The results indicate a significant variation in accuracy, prediction time, CPU utilization, and MU among the models. MLP achieves the highest accuracy (0.885), demonstrating strong prediction capabilities, but at the cost of increased CPU utilization (57.14%) and MU (140.119 MB). On the other hand, LR provides a balanced performance with good accuracy (0.818) and low computational costs, making it also a suitable choice for applications with limited resources. RF model, while offering moderate accuracy (0.738), has the largest MS (33,009 KB) and the highest memory consumption (267.679 MB), indicating a trade-off between performance and resource efficiency. Models like BNB and GNB, with accuracies of 0.725 and 0.721, respectively, exhibit low prediction time and MU, but their high CPU utilization makes them less practical for real-time applications. Compared to *Scenario 2*, where continuous learning was employed, *Scenario 3* shows that M2 without continuous learning achieves higher accuracy in some models, such as MLP, but with increased computational overhead, indicating the benefits of continuous learning for resource optimization and real-time applicability. Considering these factors, we utilize M1 with continuous learning at the first level and then deploy the best-performing model from M2 at the second level in MULTI-LF.

TABLE XI: **Scenario 3:** M2 Without Continuous Learning

Model	MS (KB)	Accuracy	PT (S)	CPU (%)	MU (MB)
ADA	57	0.723	0.085	39.91	96.179
BNB	4	0.725	0.001	60.47	129.028
GNB	4	0.721	0.005	56.28	137.565
LR	2	0.818	0.001	48.16	123.635
MLP	106	0.885	0.016	57.14	140.119
MNB	4	0.792	0.0009	53.57	133.187
Perceptron	3	0.737	0.0009	61.79	151.869
RF	33009	0.738	0.0313	55.18	267.679
SGDC	3	0.830	0.002	46.00	137.654

Table XII shows the performance comparison of four models, BNB, MNB, Perceptron, and SGDC, under *Scenario 4*, where both M1 and M2 models are deployed in MULTI-LF without human involvement. The first set of results evaluates the models with MLP as the secondary model (M2). The Perceptron model achieves the highest accuracy at 0.999, but at the cost of significantly higher CPU utilization (23.22%) and MU (2.623 MB), indicating its com-

putational intensity. In contrast, BNB maintains high accuracy at 0.975 with a much lower CPU usage (2.582%) and MU (1.513 MB), suggesting a good trade-off between performance and resource efficiency.

The second set of results involves the same models but with RF as the M2. The accuracy of all models drops compared to the MLP scenario, with the highest accuracy achieved by the Perceptron model (0.983). However, the computational overhead is significantly reduced, as evidenced by lower CPU utilization (3.221%) and MU (0.956 MB). This suggests that using RF as the secondary model is less resource-intensive, but it comes at the cost of reduced prediction performance. The results highlight the trade-offs between model performance and resource efficiency when using different secondary models (MLP and RF) within MULTI-LF. The Perceptron model consistently achieves high accuracy but at a higher computational cost, while BNB and other models offer a balanced approach between accuracy and resource usage.

TABLE XII: **Scenario 4:** M1 and M2 Under MULTI-LF Without Human Involvement

MLP as M2 & Without Human Check				
Matrix	BNB	MNB	Perceptron	SGDC
Accuracy	0.975	0.331	0.999	0.371
PT (S)	0.332	0.270	1.259	0.144
CPU (%)	2.582	0.027	23.22	0.034
MU (MB)	1.513	0.041	2.623	0.030
RF as M2 & Without Human Check				
Accuracy	0.941	0.330	0.983	0.186
PT (S)	0.216	0.270	0.278	0.164
CPU (%)	1.240	0.023	3.221	0.042
MU (MB)	1.338	0.017	0.956	0.001

In *Scenario 5*, M1 and M2 models were evaluated under the MULTI-LF with human involvement. The models used were BNB, MNB, Perceptron, and SGDC, with MLP and RF as the secondary models (M2), same as *Scenario 4*. This scenario demonstrated the impact of incorporating human oversight on model performance. When MLP was used as the secondary model, the Perceptron achieved the highest accuracy of 0.999, but it required significant CPU utilization (17.84%) and MU (35.48 MB). The prediction time of 0.866 seconds indicates that while human involvement boosts the model’s accuracy, it also increases computational costs. In contrast, the BNB model showed good performance with an accuracy of 0.979, but its MU (56.53 MB) remained notably high compared to other models. When RF was used as the secondary model, Perceptron again demonstrated high accuracy (0.983) but with a reduced CPU utilization of 10.05% and lower MU (3.632 MB). Prediction time was also reduced to 0.434 seconds, indicating that RF as the secondary model results in a more resource-efficient configuration while maintaining high accuracy. BNB’s performance with RF was similar to its performance with MLP in terms of accuracy, but it achieved lower CPU utilization (2.564%) and higher MU (72.69 MB).

Comparing *Scenario 5* to *Scenario 4* MULTI-LF, we observe that human intervention slightly increases computational resource usage but improves model accuracy, demonstrating the value of human oversight. Furthermore, *Scenario 5* outperformed *Scenario 1* and *Scenario 2* (M1 configurations) in accuracy across most models, validating the robustness of the proposed human-involved framework. Finally, compared to *Scenario 3* (M2 without continuous learning), *Scenario 5* exhibited a more balanced trade-off between accuracy and computational efficiency, especially when using RF as the secondary model, highlighting the advantage of human intervention in refining the model’s performance.

Figure 7 shows the different iteration scores for *Scenarios 4* and *5*. It is evident that model accuracy improves with continuous learning as more data is provided. However, the prediction time remains consistent over the iterations.

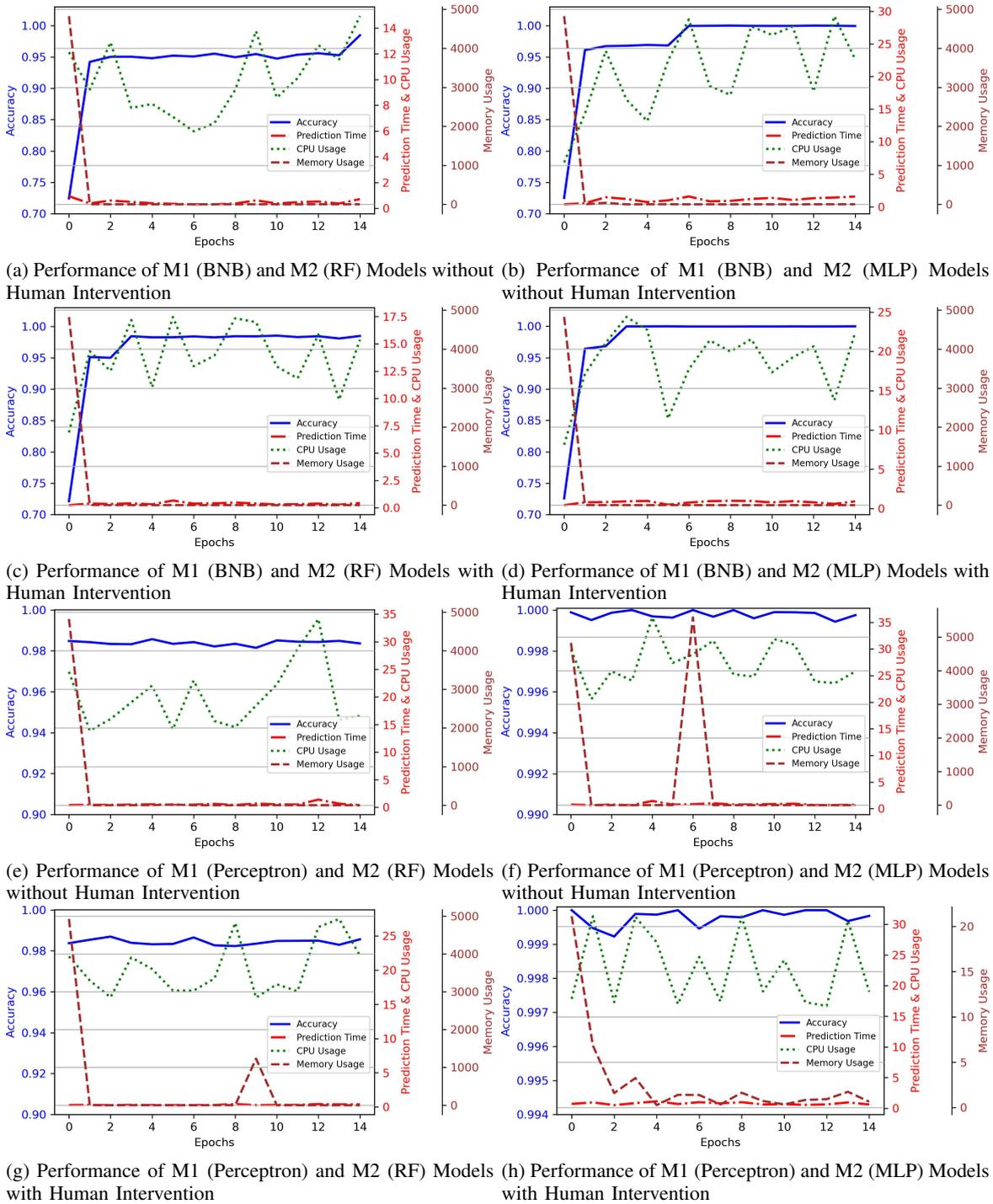


Fig. 7: Scenario 4 and Scenario 5: Best performing models' results per batch

TABLE XIII: Scenario 5: M1 and M2 Under MULTI-LF With Human Involvement

MLP as M2 & Human Check				
Matrix	BNB	MNB	Preceptron	SGDC
Accuracy	0.979	0.331	0.999	0.371
PT (S)	0.374	0.270	0.866	0.174
CPU (%)	3.158	0.022	17.84	0.035
MU (MB)	56.53	0.009	35.48	0.018
RF as M2 & Human Check				
Accuracy	0.975	0.331	0.983	0.384
PT (S)	0.331	0.270	0.434	0.178
CPU (%)	2.564	0.019	10.05	0.102
MU (MB)	72.69	0.020	3.632	0.020

D. Comparison With Benchmark Studies

Many studies validate their MTDS approaches in offline settings, often achieving significant results. However, these systems

frequently experience performance degradation when deployed in real-time environments due to the dynamic nature of network traffic and the emergence of new attack patterns. The datasets used to train these models are typically static, which limits their effectiveness in real-world applications.

There are no existing studies that evaluate their work on real-time online M-En traffic. Therefore, for a fair comparison, we implemented existing M-En studies on our newly collected real-time M-En dataset, deploying them according to the methodologies outlined in their respective published work. Since most existing studies conducted only offline testing, we replicated their models in the same offline setting to maintain consistency in evaluation. For instance, Rustam et al. [47] deployed their approach on an M-En dataset using the synthetic data augmentation technique (S-DATE). While the performance appeared strong after applying the S-DATE data balancing method, the results indicated potential data leakage

issues, which could explain the observed high accuracy. Similarly, other studies [10], [14] proposed optimization-based approaches to handle diverse traffic patterns in M-En networks, employing *Particle Swarm Optimization* (PSO) and *Moth Flame Optimization* (MFO), respectively. These studies introduced *PSO-diverse self-ensemble model* (PSO-D-SEM) and a *Fully Automated Malicious Traffic Detection System* (FAMTDS). Another study [70] utilized a *dual-data trained LightGBM* (DDT-LightGBM) model, achieving significant results in the M-En network within an offline testing framework. Zukaib et al. [49] integrated Federated Learning and Meta-learning to propose the Meta-Fed IDS, which was tested on the M-En dataset. However, their evaluation was limited to offline settings and did not incorporate real-time data collection or adaptation. In contrast, our study conducted both online and offline testing and was built on a real-time M-En dataset.

In Table XIV, we compare our approach, MULTI-LF, to relevant M-En benchmark studies. Most existing approaches [10], [14], [47], [49], [70] use static datasets and evaluate performance only in offline scenarios, reporting accuracy values between 0.94 and 0.991. While these systems can be effective in controlled settings, they often lack the mechanisms to continuously learn from new traffic patterns or adapt to real-time fluctuations in M-En networks.

In contrast, MULTI-LF is the only method evaluated in both offline and online modes, achieving near-optimal accuracy scores of 1.00 (offline) and 0.999 (online). These results highlight MULTI-LF’s resilience to dynamic traffic behavior. By continuously re-training on fresh data and utilizing multi-level validation checks, MULTI-LF maintains high accuracy even under unpredictable network conditions, offering a robust solution for real-time MTDS in M-En environments.

TABLE XIV: Comparison With Existing Studies

Ref.	Year	Approach	Testing		Results
			offline	online	
[47]	2023	ETC, S-DATE	✓	✗	0.986
[14]	2024	PSO-D-SEM	✓	✗	0.978
[10]	2024	FAMTDS	✓	✗	0.991
[70]	2024	DDT-LightGBM	✓	✗	0.98
[49]	2024	Meta-Fed IDS	✓	✗	0.94
-	2025	MULTI-LF	✓	✓	1.00, 0.999

V. DISCUSSION

This study collected a dataset in the M-En environment and proposed the MULTI-LF framework for attack detection, testing it in both offline and real-time scenarios. We deployed MULTI-LF to reduce computational costs and enhance accuracy over time. The computational cost is reduced because the initial traffic is evaluated by the lightweight M1 model, which has a faster prediction time, as shown in Table VIII. Most of the traffic is filtered by M1, and only a small portion is forwarded to M2 and, if necessary, to a human expert. Additionally, the approach’s performance improved by incorporating an extra layer of security using M2 and human involvement. This is reflected in the improvement in accuracy from 0.70 to 0.99, as shown in Figure 8.

The improvement in model performance was achieved through continuous training with new data, allowing the model to efficiently adapt to new traffic patterns as well as through our feature engineering approach. The use of statistical features in our methodology significantly contributed to the model’s success, as shown in Figure 9. Statistical features such as *ConnectionErrors*, *DstPortEntropy*, *MostFreqPayloadSize*, *SourceEntropy*, *MostFreqPacketSizeFreq*, *FlowRate*, *PacketCount*, *PacketSizeVar*, and *AvgPayloadSize* played a major role in detecting DDoS attacks. For example, a sudden increase in packet count from a specific IP within

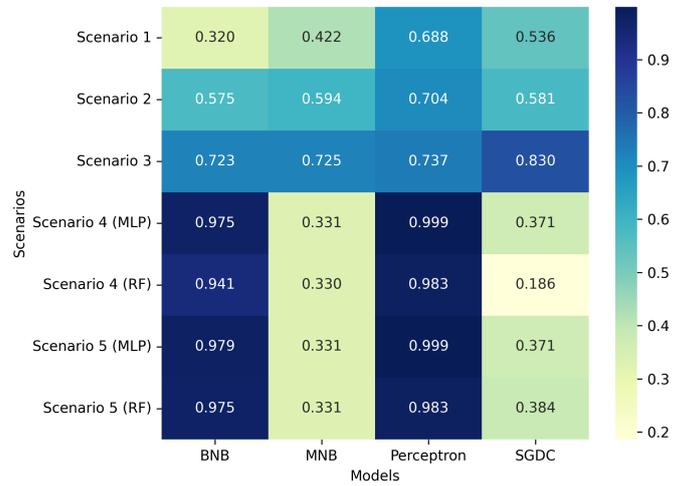


Fig. 8: Accuracy comparison with difference scenarios

a time window can indicate an attack. These statistical features, combined with general features, generated highly correlated inputs, as shown in Figure 3, thus improving the performance of the framework.

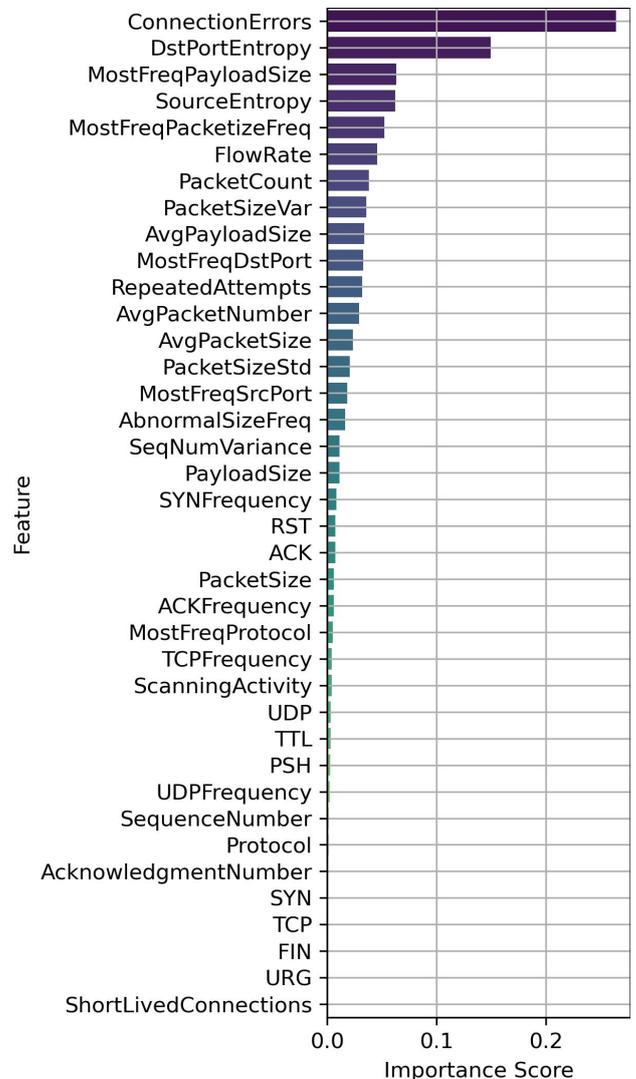


Fig. 9: Feature importance score

Table XV illustrates the performance of different scenarios, highlighting variations in network traffic and the significance of a unified approach using the M-En dataset. In our experiments, we first trained the model with IoT data tested it with traditional

(trad.) data, and then trained it on trad. data and tested it with IoT data. The models performed poorly in both cases due to the distinct traffic patterns, demonstrating that existing SOTA datasets are inadequate for creating a unified framework for M-En networks. Subsequently, we conducted experiments using the M-En dataset, training models on it, and testing with both IoT and trad. traffic. The models showed significant improvements in handling both types of traffic, underscoring the effectiveness of our M-En dataset for training models in diverse scenarios.

TABLE XV: Results using Different Training and Testing Datasets

Training	Testing	Accuracy	Class	Precision	Recall	F1-Score
IoT	Trad.	0.0066	0	0.01	0.03	0.01
			1	0.00	0.00	0.00
Trad.	IoT	0.0142	0	0.13	0.02	0.03
			1	0.00	0.00	0.00
M-En	IoT and Trad.	0.9993	0	1.00	1.00	1.00
			1	1.00	1.00	1.00

Furthermore, Figure 10 demonstrates the performance of our testing in a real-time scenario. Figure 10a, illustrates the connection between the server and the gateway of M-En networks, which facilitates the real-time ingestion of network packets for processing. Once the connection is established, the models begin analyzing the traffic. Initially, the performance is suboptimal, showing accuracy levels around 50-60%. However, as the system continues to operate, the performance progressively improves due to the continuous learning approach implemented in our framework. The continuous learning mechanism enables the models to adapt to the evolving traffic patterns by retraining on newly collected data in real-time as shown in Figure 10b. Over time, the accuracy significantly increases, stabilizing at 99-100%, as seen in the later stages of the testing process. This improvement highlights the robustness and adaptability of our framework, as well as the importance of incorporating real-time data and continual learning for addressing the dynamic nature of M-En network environments. These results further validate the effectiveness of our unified framework and its ability to handle diverse and evolving traffic scenarios in real-time applications.

Significance: This study has several significance in network security for DDoS attack detection and shows strong contribution in the given domain:

- This study introduces a comprehensive benchmark dataset specifically tailored for M-En networks, providing a foundational resource for researchers to advance security solutions in complex, M-En network settings.
- The availability of our research resources in an open repository not only ensures transparency and reproducibility but also encourages further development, validation, and deployment of novel methodologies in the domain of M-En network security.
- By incorporating continuous learning and human intervention, the framework demonstrates an ability to adapt dynamically to new and unseen traffic patterns. This adaptability is essential for dealing with zero-day attacks and evolving threats, providing a sustainable approach to long-term network security.
- MULTI-LF ability to leverage a lightweight model (M1) for initial detection significantly reduces computational overhead, which is crucial for real-time applications. This ensures that the framework is not only accurate but also efficient, making it suitable for deployment in resource-constrained environments.

Limitations: Despite the significance of this study, there are several limitations to consider:

- This study focuses primarily on a few DDoS attacks and does not explore other types of attacks, such as *Man-in-the-Middle* (MiM) and other sophisticated network attacks, which restricts the comprehensiveness of the dataset.
- This study focuses solely on IoT and traditional networks within the M-En framework, which is limited. Numerous other networks, including SDN and industrial IoT, can be incorporated into M-En networks, which could be helpful in bigger networks.
- MULTI-LF requires human intervention, which can slow down the decision-making process. On average, a human expert takes approximately 1 to 2 minutes to make a decision. To address this, a faster and more efficient processing system is needed for timely decision-making.

VI. CONCLUSION

This study effectively addresses DDoS attack detection in M-En networks using a continuous learning approach. A pivotal contribution of this research is the creation of a novel benchmark dataset specifically designed for M-En networks, integrating both IoT and traditional IP-based traffic patterns. Utilizing this comprehensive dataset, we developed a highly reliable methodology optimized for real-time operational scenarios. Our extensive analysis demonstrates that the M-En network dataset exhibits distinct characteristics compared to existing domain-specific benchmark datasets, such as greater variability in packet size and frequency. In contrast, domain-specific datasets tend to have consistent traffic patterns. Consequently, ML models trained on domain-specific datasets underperform when applied to diverse traffic domains, highlighting the inherent challenges in developing a unified detection framework. However, the inherent diversity within the M-En dataset enables the development of a unified system that can reduce computational costs, resource usage, and management overhead. Furthermore, the study concludes that models trained without a continuous learning paradigm suffer significant performance degradation over time due to evolving traffic patterns and dynamic network topologies, such as the addition or removal of devices. In contrast, our proposed MULTI-LF, incorporating continuous learning mechanisms, effectively mitigates these issues by enabling the model to adapt to zero-day attacks and scale efficiently within M-En networks. This results in enhanced reliability and sustained accuracy over time.

```

lobaidat@lobaidat-VirtualBox:~/DdosShield-IoT
Success: mirror Traffic from 'tap-emu3' to 'sl-emu3'
Success: Checking NS3 directory
About to start NS3 RUN with total emulation time of 600
NS3_HOME=/network/ns-allinone-3.42/ns-3.42 && cd $NS3_HOME && ./ns3 run -i 3 "s
cratch/tap-vn --NumNodes=5 --TotalTime=600 --Churn=0 --FileLog=0 --TapBaseName=e
mu --WriteDirectory=/home/lobaidat/DdosShield-IoT/results --NoneDevsNodes=3 --Ani
mationOn=false"
Sudo password:
[0/2] Re-checking globbed directories...
ninja: no work to do.
Sudo password:
*****
Target Server IPv4: 10.0.0.1
Target Server MAC:02:06:00:00:00:00:01
*****
Press the Enter key to continue...
proc1 = 31396
Running NS3 in the background [ Date now: 2024-12-03 17:53:00.449492
lobaidat@lobaidat-VirtualBox:~/DdosShield-IoT

```

(a) Server connection establishment

```

root@24e8c6dfb36:/ml
Accuracy = 0.5689, Pred. Time = 0.011098623275756836
Accuracy = 0.3052, Pred. Time = 0.023389816284179688
Accuracy = 0.4870, Pred. Time = 0.0941765308388127
Accuracy = 0.9293, Pred. Time = 0.13257431983947724
Accuracy = 0.6954, Pred. Time = 0.07377362251281738
Accuracy = 0.2408, Pred. Time = 0.11352992057800293
Accuracy = 0.6232, Pred. Time = 0.27958083152770996
Accuracy = 1.0000, Pred. Time = 0.018991267402919156
Accuracy = 1.0000, Pred. Time = 0.08561857235107422
Accuracy = 1.0000, Pred. Time = 0.0806253719329833984
Accuracy = 1.0000, Pred. Time = 0.0022313594018115234
Accuracy = 1.0000, Pred. Time = 0.0026223195318603510
Accuracy = 1.0000, Pred. Time = 0.0065157413482666010
Accuracy = 1.0000, Pred. Time = 0.00826416778564453125
Accuracy = 1.0000, Pred. Time = 0.0003054149982910156
Accuracy = 1.0000, Pred. Time = 0.003715735352978156
Accuracy = 1.0000, Pred. Time = 0.0038588089447021844
Accuracy = 1.0000, Pred. Time = 0.008871978491210938
root@006e872a7a1:/ml
Model 2: 'mlclassifierfor40.pkl' successfully loaded.
Scaler 1: 'scaler5.pkl' successfully loaded.
Scaler 2: 'scalerfor40.pkl' successfully loaded.
Accuracy = 1.0000, Pred. Time = 0.006399631508244141
Accuracy = 0.0049, Pred. Time = 0.004557481273630766
Accuracy = 1.0000, Pred. Time = 0.0026805376739591953
Accuracy = 1.0000, Pred. Time = 0.007260794079559884
Accuracy = 1.0000, Pred. Time = 0.00038546502118164006
Accuracy = 1.0000, Pred. Time = 0.00020616257133672
Accuracy = 0.0755, Pred. Time = 0.0085507469177246094
Accuracy = 0.0060, Pred. Time = 0.001514011631013201
Accuracy = 0.0104, Pred. Time = 0.0005233519439097046
Accuracy = 0.0000, Pred. Time = 0.013351448426675
Accuracy = 0.0000, Pred. Time = 0.000184016304073033
Accuracy = 1.0000, Pred. Time = 0.000899735181033359
Accuracy = 0.6556, Pred. Time = 0.05356006164459708
Accuracy = 1.0000, Pred. Time = 0.00019073486328125
Accuracy = 0.0706, Pred. Time = 0.000206537922332812
Accuracy = 1.0000, Pred. Time = 0.0053202045849669375

```

(b) Live performance of models and continuous improvement

Fig. 10: Real-time testing snapshots

Additionally, MULTI-LF achieves high classification accuracy and low prediction latency in real-time testing environments, although offline testing exhibits marginally superior performance metrics. Importantly, MULTI-LF continuously improves its real-time performance by fine-tuning with newly acquired data, thereby maintaining robustness against previously unseen attack vectors. This study also concludes that human involvement during the continuous learning process is highly effective in avoiding bias and improving transparency and trust in the framework.

However, despite the significant contributions of this study, we acknowledge certain limitations and outline directions for future work. We intend to explore additional unique attack types to further enhance the system's efficiency. Additionally, we consider integrating *Large Language Models* (LLMs) to replace human intervention, enabling faster and more reliable decision-making with increased trust.

ACKNOWLEDGMENT

This work is funded by the School of Computer Science and CHIST-ERA ERA-NET - SPiDDS Topic and Irish Research Council (IRC) for funding support

AUTHOR CONTRIBUTIONS

Furqan Rustam: Writing – original draft, Visualization, Software, Validation, Methodology, Investigation, Formal analysis, Data curation, Conceptualization, Writing – review & editing. Islam Obaidat: Visualization, Validation, Investigation, Formal analysis, Data Curation, Writing – review & editing. Anca Delia Jurcut: Writing – review & editing, Validation, Supervision, Software, Project administration, Formal analysis, Conceptualization.

REFERENCES

- [1] Z. Zhao, Z. Liu, H. Chen, F. Zhang, Z. Song, and Z. Li, "Effective ddos mitigation via ml-driven in-network traffic shaping," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–18, 2024.
- [2] A. Nazir, J. He, N. Zhu, A. Wajahat, X. Ma, F. Ullah, S. Qureshi, and M. S. Pathan, "Advancing iot security: A systematic review of machine learning approaches for the detection of iot botnets," *Journal of King Saud University - Computer and Information Sciences*, vol. 35, no. 10, p. 101820, 2023.
- [3] A. Wang, W. Chang, S. Chen, and A. Mohaisen, "A data-driven study of ddos attacks and their dynamics," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 3, pp. 648–661, 2020.
- [4] S. Ismail, H. R. Hassen, M. Just, and H. Zantout, "A review of amplification-based distributed denial of service attacks and their mitigation," *Computers & Security*, vol. 109, p. 102380, 2021.
- [5] S. De Vivo, I. Obaidat, D. Dai, and P. Liguori, "DDoShield-IoT: A testbed for simulating and lightweight detection of IoT botnet DDoS attacks," in *Proceedings of the 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pp. 1–8, 2024.
- [6] Y. Cao, H. Jiang, Y. Deng, J. Wu, P. Zhou, and W. Luo, "Detecting and mitigating ddos attacks in sdn using spatial-temporal graph convolutional network," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 6, pp. 3855–3872, 2022.
- [7] J. Bhayo, S. A. Shah, S. Hameed, A. Ahmed, J. Nasir, and D. Draheim, "Towards a machine learning-based framework for ddos attack detection in software-defined iot (sd-iot) networks," *Engineering Applications of Artificial Intelligence*, vol. 123, p. 106432, 2023.
- [8] M. Najafimehr, S. Zarifzadeh, and S. Mostafavi, "Ddos attacks and machine-learning-based detection methods: A survey and taxonomy," *Engineering Reports*, vol. 5, no. 12, p. e12697, 2023.
- [9] R. C. Paffenroth and C. Zhou, "Modern machine learning for cyber-defense and distributed denial-of-service attacks," *IEEE Engineering Management Review*, vol. 47, no. 4, pp. 80–85, 2019.
- [10] F. Rustam, W. Aljedaani, M. S. Elsayed, and A. D. Jurcut, "Famtds: A novel mfo-based fully automated malicious traffic detection system for multi-environment networks," *Computer Networks*, vol. 251, p. 110603, 2024.
- [11] F. Rustam, P. S. Ranaweera, and A. D. Jurcut, "Ai on the defensive and offensive: Securing multi-environment networks from ai agents," in *Proceedings of the IEEE International Conference on Communications (ICC)*, (USA), p. To be assigned, March 2024.
- [12] I. Ullah and Q. H. Mahmoud, "A scheme for generating a dataset for anomalous activity detection in iot networks," in *Advances in Artificial Intelligence* (C. Goutte and X. Zhu, eds.), (Cham), pp. 508–520, Springer International Publishing, 2020.
- [13] N. Moustafa and 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)*, pp. 1–6, 2015.
- [14] F. Rustam and A. D. Jurcut, "Malicious traffic detection in multi-environment networks using novel s-date and pso-d-sem approaches," *Computers & Security*, vol. 136, p. 103564, 2024.
- [15] M. B. Anley, A. Genovese, D. Agostinello, and V. Piuri, "Robust ddos attack detection with adaptive transfer learning," *Computers & Security*, p. 103962, 2024.
- [16] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (ddos) attack dataset and taxonomy," in *2019 International Carnahan Conference on Security Technology (ICCST)*, pp. 1–8, 2019.
- [17] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *International Conference on Information Systems Security and Privacy*, 2018.
- [18] M. Tavallaei, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*, pp. 1–6, Ieee, 2009.
- [19] A. M. Al-Eryani, E. Hossny, and F. A. Omara, "Efficient machine learning algorithms for ddos attack detection," in *2024 6th International Conference on Computing and Informatics (ICCI)*, pp. 174–181, 2024.
- [20] A. Singh, H. Kaur, and N. Kaur, "A novel ddos detection and mitigation technique using hybrid machine learning model and redirect illegitimate traffic in sdn network," *Cluster Computing*, vol. 27, no. 3, pp. 3537–3557, 2024.
- [21] A. Waleed, A. F. Jamali, and A. Masood, "Which open-source ids? snort, suricata or zeek," *Computer Networks*, vol. 213, p. 109116, 2022.
- [22] M. A. Talukder, M. M. Islam, M. A. Uddin, K. F. Hasan, S. Sharmin, S. A. Alyami, and M. A. Moni, "Machine learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction," *Journal of Big Data*, vol. 11, no. 1, p. 33, 2024.
- [23] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "A detailed analysis of the cicsids2017 data set," in *Information Systems Security and Privacy: 4th International Conference, ICISSP 2018, Funchal-Madeira, Portugal, January 22-24, 2018, Revised Selected Papers 4*, pp. 172–188, Springer, 2019.
- [24] M. H. M. Yusof, A. A. Almomhammedi, V. Shepelev, and O. Ahmed, "Visualizing realistic benchmarked ids dataset: Cira-cic-dohbrw-2020," *IEEE Access*, vol. 10, pp. 94624–94642, 2022.
- [25] V. S. V. Hema, S. Devadharshini, and P. Gowsalya, "Malicious traffic flow detection in iot using ml based algorithms," *International Research Journal on Advanced Science*, vol. 3, no. 5, pp. 68–76, 2023.
- [26] Z. Niu, J. Xue, Y. Wang, T. Lei, W. Han, and X. Gao, "Qarf: A novel malicious traffic detection approach via online active learning for evolving traffic streams," *Chinese Journal of Electronics*, vol. 33, no. 3, pp. 645–656, 2024.
- [27] A. E. Omolara, A. Alabdulatif, O. I. Abiodun, M. Alawida, A. Alabdulatif, H. Arshad, et al., "The internet of things security: A survey encompassing unexplored areas and new insights," *Computers & Security*, vol. 112, p. 102494, 2022.
- [28] B. Babayigit and M. Abubaker, "Towards a generalized hybrid deep learning model with optimized hyperparameters for malicious traffic detection in the industrial internet of things," *Engineering Applications of Artificial Intelligence*, vol. 128, p. 107515, 2024.
- [29] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-iiotset: A new comprehensive realistic cyber security dataset of iot and iiot applications for centralized and federated learning," *IEEE Access*, vol. 10, pp. 40281–40306, 2022.
- [30] M. Zolanvari, M. A. Teixeira, L. Gupta, K. M. Khan, and R. Jain, "WUSTL-IIOT-2021 Dataset for IIoT Cybersecurity Research," October 2021.
- [31] M. Al-Hawawreh, E. Sitnikova, and N. Aboutorab, "X-iiotid: A connectivity-agnostic and device-agnostic intrusion data set for industrial internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3962–3977, 2022.
- [32] S. Zhu, X. Xu, J. Zhao, and F. Xiao, "Lkd-stnn: A lightweight malicious traffic detection method for internet of things based on knowledge distillation," *IEEE Internet of Things Journal*, vol. 11, no. 4, pp. 6438–6453, 2024.
- [33] N. Moustafa, "A new distributed architecture for evaluating ai-based security systems at the edge: Network ton_iiot datasets," *Sustainable Cities and Society*, vol. 72, p. 102994, 2021.
- [34] S. Garcia, A. Parmisano, and M. J. Erquiaga, "Iot-23: A labeled dataset with malicious and benign iot network traffic," *Stratosphere Lab., Praha, Czech Republic, Tech. Rep.*, 2020.
- [35] Y. Huo, W. Liang, J. Chen, S. Zhuang, and J. Sun, "Lightguard: A lightweight malicious traffic detection method for internet of things," *IEEE Internet of Things Journal*, pp. 1–1, 2024.
- [36] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *Computer Vision – ECCV 2018* (V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, eds.), (Cham), pp. 122–138, Springer International Publishing, 2018.
- [37] W. Wei and L. David, "USTC-TFC2016: A Dataset for Traffic Classification," 2016. Accessed: 2023-10-25.
- [38] U. o. N. B. Canadian Institute for Cybersecurity, "CIC IoT Dataset Collection," 2023. Accessed: 2023-10-25.
- [39] X. Fan, C. Li, and X. Dong, "A real-time network security visualization system based on incremental learning (chinavis 2018)," *Journal of Visualization*, vol. 22, pp. 215–229, 2019.

- [40] X. Xu, X. Zhang, Q. Zhang, Y. Wang, B. Adebisi, T. Ohtsuki, H. Sari, and G. Gui, "Advancing malware detection in network traffic with self-paced class incremental learning," *IEEE Internet of Things Journal*, vol. 11, no. 12, pp. 21816–21826, 2024.
- [41] S. Ajjaj, S. El Houssaini, M. Hain, and M.-A. El Houssaini, "Incremental online machine learning for detecting malicious nodes in vehicular communications using real-time monitoring," in *Telecom*, vol. 4, pp. 629–648, MDPI, 2023.
- [42] D. Krajzewicz, "Traffic simulation with sumo—simulation of urban mobility," *Fundamentals of traffic simulation*, pp. 269–293, 2010.
- [43] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in *Modeling and tools for network simulation*, pp. 15–34, Springer, 2010.
- [44] R. Wang, J. Fei, R. Zhang, M. Guo, Z. Qi, and X. Li, "Drnet: Dynamic retraining for malicious traffic small-sample incremental learning," *Electronics*, vol. 12, no. 12, p. 2668, 2023.
- [45] Z. Zhao, Z. Li, Z. Song, W. Li, and F. Zhang, "Trident: A universal framework for fine-grained and class-incremental unknown traffic detection," in *Proceedings of the ACM on Web Conference 2024*, pp. 1608–1619, 2024.
- [46] M. S. Elsayed, N.-A. Le-Khac, and A. D. Jurcut, "Insdn: A novel sdn intrusion dataset," *IEEE Access*, vol. 8, pp. 165263–165284, 2020.
- [47] F. Rustam, A. D. Jurcut, W. Aljedaani, and I. Ashraf, "Securing multi-environment networks using versatile synthetic data augmentation technique and machine learning algorithms," in *2023 20th Annual International Conference on Privacy, Security and Trust (PST)*, pp. 1–10, IEEE, 2023.
- [48] P. L. Indrasari, E. Lee, V. Rupapara, F. Rustam, and I. Ashraf, "Malicious traffic detection in iot and local networks using stacked ensemble classifier," *Computers, Materials and Continua*, vol. 71, no. 1, pp. 489–515, 2022.
- [49] U. Zukaib, X. Cui, C. Zheng, D. Liang, and S. U. Din, "Meta-fed ids: Meta-learning and federated learning based fog-cloud approach to detect known and zero-day cyber attacks in iomt networks," *Journal of Parallel and Distributed Computing*, p. 104934, 2024.
- [50] L. F. G. Casanova, P.-C. Lin, *et al.*, "Malicious network traffic detection for dns over https using machine learning algorithms," *APSIPA Transactions on Signal and Information Processing*, vol. 12, no. 2, 2023.
- [51] R. F. Fouladi, T. Seifpoor, and E. Anarim, "Frequency characteristics of dos and ddos attacks," in *2013 21st Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, IEEE, 2013.
- [52] K. R. Żalik, "An efficient k'-means clustering algorithm," *Pattern Recognition Letters*, vol. 29, no. 9, pp. 1385–1391, 2008.
- [53] M. Gada, Z. Haria, A. Mankad, K. Damania, and S. Sankhe, "Automated feature engineering and hyperparameter optimization for machine learning," in *2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS)*, vol. 1, pp. 981–986, IEEE, 2021.
- [54] S. Ismail and H. Reza, "Evaluation of naïve bayesian algorithms for cyber-attacks detection in wireless sensor networks," in *2022 IEEE world AI IoT congress (AllIoT)*, pp. 283–289, IEEE, 2022.
- [55] A. Angelopoulos, A. E. Giannopoulos, N. C. Kapsalis, S. T. Spantideas, L. Sarakis, S. Voliotis, and P. Trakadas, "Impact of classifiers to drift detection method: a comparison," in *International conference on engineering applications of neural networks*, pp. 399–410, Springer, 2021.
- [56] B. Sudqi Khater, A. W. B. Abdul Wahab, M. Y. I. B. Idris, M. Abdulla Hussain, and A. Ahmed Ibrahim, "A lightweight perceptron-based intrusion detection system for fog computing," *applied sciences*, vol. 9, no. 1, p. 178, 2019.
- [57] M. Panda, A. Abraham, and M. R. Patra, "Discriminative multinomial naïve bayes for network intrusion detection," in *2010 Sixth International Conference on Information Assurance and Security*, pp. 5–10, IEEE, 2010.
- [58] A. Abdulboriy and J. S. Shin, "An incremental majority voting approach for intrusion detection system based on machine learning," *IEEE Access*, 2024.
- [59] A. J. Meerja, A. Ashu, and A. Rajani Kanth, "Gaussian naïve bayes based intrusion detection system," in *Proceedings of the 11th International Conference on Soft Computing and Pattern Recognition (SoCPar 2019) 11*, pp. 150–156, Springer, 2021.
- [60] M. A. Setitra, M. Fan, B. L. Y. Agbley, and Z. E. A. Bensalem, "Optimized mlp-cnn model to enhance detecting ddos attacks in sdn environment," *Network*, vol. 3, no. 4, pp. 538–562, 2023.
- [61] S. Chalichalamala, N. Govindan, and R. Kasarapu, "Logistic regression ensemble classifier for intrusion detection system in internet of things," *Sensors*, vol. 23, no. 23, p. 9583, 2023.
- [62] J. Azimjonov and T. Kim, "Stochastic gradient descent classifier-based lightweight intrusion detection systems using the efficient feature subsets of datasets," *Expert Systems with Applications*, vol. 237, p. 121493, 2024.
- [63] B. Sharmila and R. Nagapadma, "Intrusion detection system using naïve bayes algorithm," in *2019 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)*, pp. 1–4, IEEE, 2019.
- [64] A. Nugroho, R. Wahyuwidayat, S. T. Sianturi, M. Fauzi, M. F. Ramadhan, B. A. Pratomo, A. M. Shiddiqi, *et al.*, "Ensemble methods classifier comparison for anomaly based intrusion detection system on cids-002 dataset," in *2021 13th International Conference on Information & Communication Technology and System (ICTS)*, pp. 62–67, IEEE, 2021.
- [65] M. A. M. Hasan, M. Nasser, S. Ahmad, and K. I. Molla, "Feature selection for intrusion detection using random forest," *Journal of information security*, vol. 7, no. 3, pp. 129–140, 2016.
- [66] W. Hu, W. Hu, and S. Maybank, "Adaboost-based algorithm for network intrusion detection," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 2, pp. 577–583, 2008.
- [67] L. Khan, M. Awad, and B. Thuraisingham, "A new intrusion detection system using support vector machines and hierarchical clustering," *The VLDB journal*, vol. 16, pp. 507–521, 2007.
- [68] M. Govindarajan and R. Chandrasekaran, "Intrusion detection using k-nearest neighbor," in *2009 First International Conference on Advanced Computing*, pp. 13–20, IEEE, 2009.
- [69] Y. A. Ali, E. M. Awwad, M. Al-Razgan, and A. Maarouf, "Hyperparameter search for machine learning algorithms for optimizing the computational complexity," *Processes*, vol. 11, no. 2, p. 349, 2023.
- [70] F. Rustam, R. Shafique, S. K. Posa, and A. D. Jurcut, "Malicious traffic detection in multi-environment network using dual-data trained lightgbm approach," in *2024 IEEE 21st International Conference on Mobile Ad-Hoc and Smart Systems (MASS)*, pp. 598–603, IEEE, 2024.

APPENDIX A BIOGRAPHY SECTION

Furqan Rustam is currently pursuing a Ph.D. degree in Computer Science from University College Dublin, Ireland. He received his MCS degree in the Department of Computer Science, Islamia University of Bahawalpur, Pakistan (Oct-2015 to Oct-2017). In Nov-2018, he got himself enrolled in the Master of Computer Science, Department of Computer Science, Khwaja Fareed University of Engineering and Information Technology (KFUEIT), Rahim Yar Khan, 64200, Pakistan. He also served as a Research Assistant at Fareed Computing & Research Center, KFUEIT, Pakistan. His recent research interests are related to Data Mining, Machine Learning, and Artificial Intelligence, mainly working on Creative Computing and Supervised Machine Learning.

Islam Obaidat an Assistant Professor at North Carolina A&T State University, specializing in cybersecurity, IoT software security, and computer networks. He earned his Ph.D. in Computer Science and his research includes developing testbeds for simulating IoT botnet DDoS attacks, enhancing DDoS attack simulations through full system emulation, and creating large-scale memory error IoT botnets using NS3DockerEmulator. Dr. Obaidat's work aims to advance the understanding and mitigation of security threats in IoT environments.

ANCA D. JURCUT received a Bachelor of Mathematics and Computer Science degree from the West University of Timisoara, Romania, in 2007, and a Ph.D. degree from the University of Limerick, Ireland, in 2013. From 2008 to 2013, she was a Research Assistant with the Data Communication Security Laboratory, University of Limerick. From 2013 to 2015, she was working as a Postdoctoral Researcher with the Department of Electronic and Computer Engineering, University of Limerick. She was a Software Engineer with IBM, Ireland. Since 2015, she has been an Assistant Professor with the School of Computer Science, University College Dublin (UCD), Ireland. She is currently an Assistant Professor at the School of Computer Science, UCD. Her research interests include network and data security, security for the Internet of Things (IoT), security protocols, formal verification techniques, and applications of blockchain technologies in cybersecurity.