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

Furqan Rustam[a], Islam Obaidat[b], Anca Delia Jurcut[a]

[a]School of Computer Science, University College Dublin, , Dublin, D06WK27, Ireland
[b]Department of Computer Systems Technology, North Carolina A&T State University, North Carolina, USA

Abstract

Multi-environment (M-En) networks integrate diverse traffic sources, including *Internet of Things* (IoT) and traditional computing systems, creating complex and evolving conditions for malicious traffic detection. Existing *machine learning* (ML)-based approaches, typically trained on static single-domain datasets, often fail to generalize across heterogeneous network environments. To address this gap, we develop a realistic *Docker–NS3-based testbed* that emulates both IoT and traditional traffic conditions, enabling the generation and capture of live, labeled network flows. The resulting *M-En Dataset* combines this traffic with curated public PCAP traces to provide comprehensive coverage of benign and malicious behaviors. Building on this foundation, we propose MULTI-LF, a real-time continuous learning framework that combines a lightweight model (M1) for rapid detection with a deeper model (M2) for high-confidence refinement and adaptation. A confidence-based coordination mechanism enhances efficiency without compromising accuracy, while weight interpolation mitigates catastrophic forgetting during continuous updates. Features extracted at 1-second intervals capture fine-grained temporal patterns, enabling early recognition of evolving attack behaviors. Implemented and evaluated within the Docker–NS3 testbed on live traffic, MULTI-LF achieves an accuracy of 0.999 while requiring human intervention for only 0.0026% of packets, demonstrating its effectiveness and practicality for real-time malicious traffic detection in heterogeneous network environments.

Keywords: Malicious Traffic Detection, Dataset Collection, M-En Networks, Continuous Learning, Zero-Day Attacks

1. Introduction

Modern networks face a broad spectrum of malicious activities that now extend far beyond volumetric denial of service floods to include data exfiltration, protocol abuse, and other adaptive threats [1]. These attacks originate from an increasingly diverse set of hosts, including cloud servers, enterprise endpoints, and resource-constrained *Internet of Things* (IoT) devices, all interconnected in complex *multi-environment* (M-En) networks [2]. The resulting variety in traffic patterns, timing characteristics, and device capabilities undermines traditional malicious detection methods that depend on static signatures or models tuned to a single operating context. The challenge is compounded by (i) the heterogeneity between IoT and traditional traffic, (ii) the absence of a realistic M-En dataset, (iii) the class imbalance that causes models to overfit the environment that dominates the training data [3], and (iv) the need for continuous monitoring with human expertise during online learning [4]. Effective defense, therefore, requires a framework that can ingest live traffic from multiple sources (e.g., traditional systems and IoT devices) and continuously learn to adapt in real time to unseen traffic, all while imposing minimal overheads [5, 6].

Researchers have addressed this emerging threat landscape by developing numerous *machine learning* (ML) models that target specific deployment contexts [7]. Dedicated models have been trained for IoT traffic [8], for flows within software-defined networking fabrics [9, 10], and for conventional enterprise and backbone traffic (cf., [11]). These models achieve high accuracy within the intended domain; however, their performance degrades when encountering traffic that mixes protocols, spans diverse device classes, or carries attack patterns it has never seen during training. Similarly, several studies have explored M-En network security and addressed these challenges through various approaches. For instance, Rustam et al. [12] propose a fully automated malicious traffic detection system that addresses the lack of M-En datasets by generating M-En traffic through a combined IoT and traditional network dataset (IoTID-20 [13] and UNWNB-15 [14]). They tackle traffic diversity by deploying a *moth flame optimizer* (MFO) to find optimal weights of an ML algorithm for different types of network traffic. Similarly, another study by Rustam et al. [15] employs an optimization approach to generate M-En traffic, which includes both 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* (PSO). Nonetheless, most current studies create M-En traffic by merging existing datasets and then training and testing their models on the resulting dataset, which may not capture real-time M-En network patterns. In addition, combining the feature sets of different networks (since each dataset provides distinct features) can lead to the loss of meaningful traffic patterns. Lastly, none of these studies employs continuous learning with proper human oversight, causing their models' performance to

degrade over time.

This paper addresses the challenges in securing M-En networks against malicious attacks in two main contributions. In the first contribution, we introduce a comprehensive approach to M-En dataset collection that utilizes the DDoSHIELD-IoT testbed [8], which leverages the integration of NS-3 [16] (a network simulator) with Docker [17] (a container-based virtualization technology) to host real-world binaries that communicate over a simulated network. This testbed enables us to generate realistic M-En traffic by running actual binaries in a simulated network, producing benign and malicious traffic from the traditional and IoT networks, respectively. To achieve a complete range of traffic types (i.e., to incorporate the missing benign IoT and malicious traditional traffic), we utilize external PCAP datasets from the literature, ensuring our final dataset accurately reflects diverse and realistic M-En network conditions. Following our M-En dataset construction, we uniformly process all dataset packets through a consistent feature extraction pipeline, capturing both packet-level and time-based statistical features. This uniformity ensures that traffic from all sources can be directly compared and integrated. As a result, our M-En dataset provides a robust foundation for training ML models capable of detecting malicious attacks in M-En networks, offering 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 design and implement a continuous learning, real-time *Multi-Level Framework* (MULTI-LF). MULTI-LF combines two ML models, M1 and M2, and human (expert) intervention. Specifically, M1 is a lightweight model that is initially trained on a smaller, selected sample dataset. Furthermore, M1 is optimized for continuous learning with low computational costs, enabling adaptability to new traffic patterns and a faster detection rate with minimal resource requirements. M2, in contrast, is a more robust and complex model with higher accuracy, trained on a larger, more comprehensive dataset. In essence, MULTI-LF operates as follows: when a new packet arrives, M1 attempts the first prediction; if its prediction confidence falls below a threshold, MULTI-LF forwards the information about this packet to M2 for a second prediction attempt. Similarly, if M2 fails to reach a definitive prediction, MULTI-LF escalate it for manual inspection by human experts. Then, MULTI-LF updates M1 with the expert's label using a weight interpolation method where 75% of M1's existing weights remain intact and 25% come from a quick gradient update on the newly labeled sample. The updates to M1 introduce new knowledge while mitigating overfitting and catastrophic forgetting, which keeps M1 fast, adaptive, and accurate as traffic evolves.

We implement and deploy MULTI-LF in DDoSHIELD-IoT to assess its performance against live traffic. Using this deployment, we achieve an accuracy of 0.999. Across fifteen streaming batches covering more than 115,795 packets, only three instances required human intervention, resulting in just 0.0026% human involvement. Even with a high system depth, the latency rate remains unaffected. In addition, we measure resource usage and find that the lightweight M1 model maintains a throughput of 66,130 packets per second. In summary, the key contributions of this work are as follows.

- We propose MULTI-LF, a continuous learning, real-time framework that combines ML techniques with expert feedback for malicious attack detection in M-En networks. Code and dataset for reproducibility are available on GitHub[1].
- We construct an M-En traffic dataset by leveraging DDoSHIELD-IoT and integrating external PCAP datasets. This dataset encompasses both IoT and traditional (non-IoT) traffic, including benign and malicious categories.
- We propose transformer-based lightweight and complex model architectures, trained on a selective subset and a large corpus, respectively. Under MULTI-LF, these models achieve significant performance with high throughput while imposing minimal burden on human workload.
- We test and validate MULTI-LF in real-time scenarios by hosting its implementation in DDoSHIELD-IoT. This process demonstrates that MULTI-LF results in higher accuracy compared to recent approaches.
- We collect a wide range of performance metrics (e.g., accuracy, precision, throughput, CO_2 emissions) and resource measurements (e.g., memory usage and CPU utilization) under real operating conditions.

The rest of this paper is organized as follows. Section 2 presents the related work in the problem domain. Section 3 describes the proposed methodology. Section 4 discusses the evaluation and results, and Section 6 provides the conclusion and future work.

2. Related Work

In this section, we discuss recent literature on malicious traffic detection for traditional, IoT, and M-En networks. Furthermore, we review the literature on continuous learning and identify gaps that our study addresses.

2.1. Malicious Traffic Detection using Machine Learning

Malicious traffic, e.g., *Distributed Denial of Service* (DDoS) attacks, poses significant threats to network security. Many researchers have worked on efficiently mitigating these attacks. For example, Anley et al. [18] 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 [19], CSE-CIC-IDS2018 [20], UNSW-NB15 [14], and KDDCup'99 [21]. Their approach achieved accuracies of 93.62%, 99.92%, 99.84%, and

[1] `https://github.com/furqanrustam/MultiLF`

98.99% on each dataset, respectively. In a comparative study, Al-Eryani et al. [22] evaluate *machine learning* (ML) algorithms using the CICDoS2019 dataset [19], finding that *Gradient Boosting* (GB) and XGBoost achieve high accuracy (GB: 99.99%, XGBoost: 99.98%) with minimal false alarms.

Similarly, the study [23] proposes a *graph transformer-based autoencoder* (GTAE-IDS), an unsupervised packet-based graph intrusion detection framework designed for early and accurate anomaly detection. The system captures traffic and constructs sequential packet graphs, learning benign behavior using a graph autoencoder with transformer-based encoding to model contextual packet relationships. Unlike prior flow-based or supervised approaches, GTAE-IDS operates in near real time, requires no labeled data, and is resilient to unseen attacks. Experiments on CICIDS2017, CICIDS2018, and ACI-IoT2023 show >98% accuracy, even when observing only the first few packets of each communication. Another study [24] introduces a scalable SDN-CFN network model integrated with a GNN–based anomaly detection framework (GNN-NAD). The proposed system combines *software-defined networking* (SDN) and *compute-first networking* (CFN) to enable intelligent, real-time traffic analysis. GNN-NAD fuses static, vulnerability-aware attack graphs with dynamic traffic features, providing a unified and context-rich view of network security. At its core, a GNN model (GSAGE) performs graph representation learning, and an RF classifier conducts final detection. The hybrid design (GSAGE + RF) achieves a significant accuracy of 99.69, even with limited samples from the CICIDS2017 dataset.

Persistent challenges in network 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 attacks while incurring minimal overhead. Similarly, Singh et al. [25] contribute to the detection and mitigation of different attacks in *software-defined networking* (SDN) environments. Their work also presents a new dataset with over 1.7 million entries and employs two detection methods: Snort [26] (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 malicious traffic: dropping illegitimate traffic and redirecting it.

2.2. 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. [27] 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 [14], 99.99% on the CIC-IDS-2017 dataset [28], and 99.94% on the CIC-IDS-2018 dataset [20] 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 [29] 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. [30] 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 [21] with three different ML algorithms, such as RF, LR, and KNN, achieving 99%, 82%, and 98%, respectively. Babayigit et al. [31] 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 [21] demonstrate that QARF achieves 98.20% accuracy.

2.3. Malicious Traffic Detection in IoT Networks

IoT networks have attracted significant research interest due to their lower security mechanisms compared to traditional networks [32, 33]. For instance, Babayigit et al. [34] propose a novel approach to IoT malicious traffic detection using multiple-domain learning. They used Edge-IIoTSet [35], WUSTL-IIoT-2021 [36], and X-IIoTID [37] datasets in the proposed approach. Furthermore, they employ an autoencoder for feature-space fusion to convert all datasets into a 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. [38] 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 [39] and IoT-23 [40] datasets.

Huo et al. [41] introduce LightGuard, a lightweight malicious traffic detection model for IoT. LightGuard utilizes *lightweight residual block* (LRB) modules (inspired by ShuffleNetV2 [42]) and a novel ghost module for efficient feature map generation, achieving over 99.6% accuracy across diverse datasets (Edge-IIoTset [35], USTCTFC2016 [43], ToN-IoT [39] and CIC-IoT [44] datasets) while maintaining low computational com-

plexity. Babayigit et al. [34] 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 [35], WUSTL-IIoT-2021 [36], and X-IIoTID [37] 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.

2.4. 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 [45]. This approach 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. [46] 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. [47] 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) [48] and the Network Simulator (NS-3) [16]. 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. [49] 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. [50] 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.

2.5. Malicious Traffic Detection in Multi-Environment Networks

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. [15] introduce a system that leverages the *Synthetic Data Augmentation Technique* (S-DATE) and a 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 [51], UNSW-NB15 [14], and IoTID-20 [13], achieving an accuracy of 98.9%.

Building on this foundation, Rustam et al. [52] 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. [3] develop ML models trained on AI-based traffic for the M-En dataset derived from benchmark datasets UNSW-NB15 [14] and IoTID-20 [13]. 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. [53] 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. [54] 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%.

2.6. Gaps & Limitations

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 [41] or traditional networks [27]. There is a lack of research in the M-En domain, with few studies [52, 54] relying on synthetic combinations of existing benchmark datasets, as no realistic M-En datasets exist. This reliance on synthetic datasets may limit the generalizability of findings to real-world

Table 1: Summary of Related Work

Ref	Year	M-En	ML	DL	Dataset	Method	Results	RT	RE	CL	Early	Raw
[53]	2022	✓	ETC, GBM, RF	-	UNSWNB15, IoTID20	EL, PCA	98.50, 98.40	⊗	✓	⊗	⊗	⊗
[22]	2023	⊗	GBM, XGB	-	CICDoS2019	-	99.99, 99.98	⊗	✓	⊗	⊗	⊗
[1]	2023	⊗	-	DFNet	SC	-	99.93	⊗	⊗	⊗	□	⊗
[55]	2023	⊗	-	LSTM, BiLSTM, GRU, RNN	CIRACIC-DoHBrw2020	TNC	99.00	⊗	⊗	⊗	⊗	⊗
[30]	2023	⊗	RF, LR, KNN	-	NSLKDD	CorrAUC, FS	RF: 99.00	⊗	✓	⊗	□	⊗
[31]	2023	⊗	ARF	-	NSLKDD	OAL, AMS	98.20	⊗	⊗	✓	⊗	⊗
[38]	2023	⊗	-	LKD-STNN	ToNIoT, IoT23	KD, ATFD	98.	⊗	⊗	⊗	⊗	✓
[46]	2023	⊗	-	SPCIL	USTCTFC2016	AA	89.61, 94.74, 97.21	⊗	⊗	✓	⊗	⊗
[47]	2023	⊗	ARF, KNN	-	VANET	AA	-	⊗	-	✓	⊗	⊗
[49]	2023	⊗	-	1D-CNN	-	PS, KT	91.15	-	-	✓	⊗	✓
[52]	2023	✓	ETC	-	IoTID20, UNSWNB15	S-DATE	99.10	⊗	✓	⊗	⊗	⊗
[18]	2024	⊗	-	CNN, TL	CICDDoS2019, CICIDS2018, UNSWNB15, KDDCup99	AA, CNN	93.62, 99.92, 99.84, 98.99	⊗	⊗	⊗	⊗	⊗
[25]	2024	⊗	LR, SVM, NB, KNN, RF, EC, SVM-RF	ANN	SC	SVM-RF	99.1	⊗	✓	⊗	⊗	⊗
[27]	2024	⊗	RF, ET, DT, XGB	-	UNSWNB15, CICIDS2017, CICIDS2018	RO, SFE, PCA	99.59, 99.99	⊗	✓	⊗	⊗	⊗
[34]	2024	⊗	-	CNN, GRU	EdgeIIoTSet, WUSTLI-IoT2021, XIIoTID	MDL, AE, MLLE	97.68	⊗	⊗	⊗	⊗	⊗
[50]	2024	⊗	-	AE, RNN, GNN	-	tSieve, tScissors	96.71	⊗	⊗	✓	⊗	□
[15]	2024	✓	RF	-	InSDN, UNSWNB15, IoTID20	S-DATE, PSO	98.90	⊗	✓	⊗	⊗	⊗
[3]	2024	✓	ETC	-	UNSWNB15, IoTID20	AI-based MTC	98.30	⊗	✓	⊗	⊗	⊗
[54]	2024	⊗	-	-	IoMT	FL, MeL	99.82	⊗	⊗	⊗	⊗	⊗
[23]	2025	⊗	OCSVM, IF, HBOS, INNE	GAE, TE	CICIDS2017, CICIDS2018, ACIIoT2023	GTAE-IDS	>98	□	⊗	⊗	✓	✓
[24]	2025	⊗	RF	GNN	CICIDS2017	GSAGE+RF	99.69	⊗	⊗	⊗	⊗	⊗
Our	2025	✓	SGDC, PER, MNB, BNB	LSTM, LwT	SC M-En	AA, MULTI-LF	99.99	✓	✓	✓	✓	✓

RTE—(Real-Time Evaluation) whether the method evaluated in real-time; **M-En**—(Multi-Environment) whether multiple network domains are used; **RE**— (Resources Efficiency);whether the method is resource efficient; **CL**— (Continuous Learning); whether the method get update over time with recent attacks; **Raw**—Whether raw input data is used; **Early**—Whether early traffic classification is used. —— **Acronyms**: Transfer Learning (TL), Adaptive Architectures (AA), Self-Collected (SC), Stacking Feature Embedding (SFE), Two-layer Network Classification (TNC), Feature Selection (FS), Adaptive Random Forests (ARF), Online Active Learning (OAL), Adaptive Margin Sampling (AMS), Multiple-Domain Learning (MDL), Autoencoder (AE), Modified Locally Linear Embedding (MLLE), Knowledge Distillation (KD), Adaptive Temperature Function Dynamics (ATFD), Lightweight Residual Block (LRB), Ghost Module (GM), Self-Paced Class Incremental Learning (SPCIL), Pruning Strategy (PS), Knowledge Transfer (KT), Ensemble Learning (EL), Federated Learning (FL), Meta Learning (MeL), Light-weighted Transformer (LwT), Graph-Autoencoder (G-AE), Transformer-based Encoding (TE), Isolation-based Anomaly Detection using Nearest Neighbor Ensembles (INNE), Histogram-based Outlier Score (HBOS), Isolation forest (IF), One-class Support Vector Machines (OCSVM), —— **Symbols**: ✓ present, □ partial, ⊗ lacking; "-" indicates that the information is not provided in the paper.

M-En networks. Most studies performed M-En experiments by merging datasets from different domains [12, 56], which can introduce unrealistic traffic patterns and lose critical semantics due to disrupted context and feature reduction [57]. Moreover, adapting to continuously evolving threats, particularly advanced persistent threats utilizing sophisticated attack strategies, presents ongoing challenges. Some studies adopt continuous learning [49], but they are domain-specific, focusing either on IoT or other networks without human oversights [4]. Therefore, an approach is needed to handle M-En networks in real-time and update their knowledge with continuous learning.

3. Proposed Methodology

We introduce a real-time malicious traffic detection framework for M-En networks that combines machine learning with online, continuous learning, allowing the models to adapt to emerging traffic patterns as they emerge. Figure 1 outlines the workflow, which proceeds through four phases: (i) Dataset Collection, (ii) Feature Extraction, (iii) ML Approach, and (iv) Continuous Learning. We discuss the details of each phase in the following subsections.

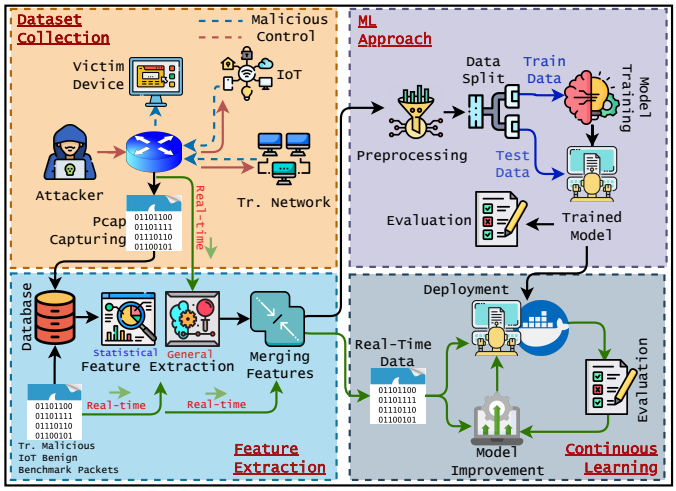


Figure 1: Proposed Framework For M-En Malicious Traffic Detection

3.1. Phase 1: Data Collection

We construct a comprehensive dataset that encompasses both IoT and traditional (non-IoT) traffic, including both benign and malicious categories. To achieve this goal, we employ a two-pronged approach. First, we leverage DDoShield-IoT [8], a testbed that leverages the integration of Docker with NS-3 to host actual binaries that communicate over a simulated network, mimicking a realistic network environment. This setup enables the generation and capturing of real-world network traffic. We use DDoShield-IoT to generate the two types of traffic it natively supports: benign traditional network traffic and malicious IoT network traffic. Second, to incorporate the other two missing traffic categories (benign IoT traffic and malicious traditional traffic), we integrate external PCAP datasets (published in the literature) into DDoShield-IoT. 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 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 different types of traffic (e.g., HTTP/HTTPS and video traffic) from legitimate services within Docker containers. For malicious IoT traffic, we utilize the Mirai malware [58] (a notorious IoT botnet) binaries hosted in a Docker container in DDoShield-IoT, from which we generate IoT-based attack traffic. In particular, we generate three Mirai-based attack classes: ACK flood, SYN flood, and UDP flood, which reflect key threats by IoT botnets in modern network environments [59].

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 proposed by other research works. This approach ensures our final dataset is both comprehensive and realistic.

   For the benign IoT traffic, we use the MQTTIoTIDS2020 dataset[2]. The MQTT-IoT-IDS2020 dataset contains benign IoT network traces, focusing on IoT devices and services that utilize 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 PCAP dataset, we incorporate authentic IoT benign traffic that closely represents everyday device activities.

   For the malicious traditional traffic, we use the CICDDoS2019 PCAP dataset[3]. This dataset includes a variety of attacks captured in a realistic testbed environment. These attacks are carried out on traditional (non-IoT) services and include multiple attack vectors that target typical enterprise or data center hosts. Incorporating this dataset enhances the representativeness of our overall traffic corpus, ensuring that our traffic detection can identify malicious patterns in both IoT and traditional network segments.

To combine these heterogeneous traffic sources, we deploy a Python sniffer running in a promiscuous Docker node inside DDoShield-IoT so it sees every frame traversing the simulated network. At each injection interval, we select a PCAP file from the external corpus and draw a random offset anywhere between its first and last packet. Rather than replaying traffic from that arbitrary byte position (which could split a session midstream), we step back to the first packet of the same flow (identified by the customary five-tuple of source IP, destination IP, source port, destination port, and transport-layer protocol [60]) and begin replaying from that clean boundary. The packets emitted from that point forward are merged with the live DDoShield-IoT traffic and passed, without source labels, into the feature-extraction pipeline (discussed in §3.2). Because both the PCAP file and the insertion point change every time, the resulting stream provides an unbiased, temporally coherent mix of IoT and traditional traffic segments.

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

[2]https://paperswithcode.com/dataset/mqtt-iot-ids2020
[3]https://www.unb.ca/cic/datasets/ddos-2019.html

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.

Attack Coverage: Across the two malicious sources, our M-En dataset spans fifteen distinct attack classes. DDoSHIELD-IoT (Mirai) contributes three IoT-centric floods (SYN, ACK, and UDP) that reflect the malware's most common attacks [59]. The external CICDDoS2019 traces add twelve additional traditional attacks: NTP, PortScan, DNS, LDAP, MSSQL, NetBIOS, SNMP, SSDP, UDP, UDP-Lag, WebDDoS, SYN, and TFTP floods. In addition, the MQTTIoTIDS2020 dataset provides two reconnaissance scenarios (Sparta SSH brute force and a UDP scan), which we include to capture low-volume probing behavior. This combination gives our corpus both high-rate floods and short-burst probing activity, providing broad coverage for M-En detection research. Because DDoSHIELD-IoT Docker and NS-3 architecture lets us plug in new binaries or replay additional PCAP, future attack classes can be integrated with minimal effort, either by implementing the attack inside the testbed or by importing an annotated PCAP and following the same injection procedure described above.

Attack Coverage

```
SYN flood, ACK flood, and UDP flood, NTP, PortScan, DNS,
LDAP, MSSQL, NetBIOS, SNMP, SSDP, UDP, UDP-Lag, WebDDoS,
SYN, and TFTP flood, Sparta SSH brute-force, UDP scan
```

3.2. Phase 2: Feature Extraction

After capturing all M-En traffic, we apply a uniform feature extraction and aggregation methodology. This uniformity means that regardless of whether a packet is captured from the simulated environment or sourced from an external PCAP 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, this process presents a coherent feature space, making it possible to directly compare, combine, and evaluate M-En data from multiple origins using a single, unified analytic framework.

We categorize the extracted features into two types: general features (e.g., protocol types and payload sizes) and statistical features computed over fixed time intervals (e.g., the average packet size and the packet rate). General features are directly related to traffic characteristics, while statistical features are derived from analyzing general features over time to detect patterns indicative of malicious attacks.

The general features are essential for a 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 insight into connection states and potential malicious behaviors, including SYN flooding or connection resets. *Sequence and acknowledgment numbers* are vital for reconstructing TCP sessions and detecting session hijacking or manipulation. Finally, the *packet size* and *payload size* metrics help identify 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.

General Feature

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

Despite the value of general features (per-packet descriptors), they alone cannot flag malicious activity reliably, because many network attacks reveal themselves through temporal patterns (sustained bursts of traffic, rapid-fire connection attempts, or repeated request sequences) that only emerge when packets are viewed in aggregate [61]. Therefore, we also extract statistical features over fixed windows to expose such patterns and anomalies. These 24 statistical features are crucial for identifying the diverse range of attacks. Our feature extraction pipeline is modular, allowing for the easy integration of additional features whenever new threats or protocols emerge. Below, we list the selected statistical features and explain the rationale behind the key thresholds; the full calculation details are provided in Appendix Appendix A.

Statistical Feature

```
'Packet Count', 'Destination Port Entropy', 'Most Frequent
Source Port', 'Most Frequent Destination Port',
'Short-lived Connections', 'Repeated Connection Attempts',
'Network Scanning Activity', 'Flow Rate, Source Entropy',
'Connection Errors~(RST)', 'Most Frequent Packet
Size Frequency', 'Abnormal Packet Size Frequency', 'Sequence
Number Variance', 'SYN Frequency', 'ACK Frequency',
'TCP Frequency', 'UDP Frequency', 'Packet Size Variability',
'Average Packet Size', 'Average Payload Size'
```

Thresholds & Parameters For Simulation: In our approach, we use specific thresholds and parameters to calculate these statistical features as shown in Table 2. We set the *processing interval* to 1 s, a value that strikes a balance between latency and statistical stability; in a preliminary sweep of 100 ms–2 s windows, 1 s produced the highest F_1 score while keeping detection delay sub-second. We adopt 1500 B as the *abnormal size threshold* because it aligns with the default IP *maximum-transmission-unit* (MTU) on typical networks [62, 63]; packets larger than this value ordinarily require fragmentation, so their presence is a useful indicator of atypical or crafted traffic, both of which warrant closer inspection. The

port-frequency threshold is set to five occurrences per window, an empirically derived cutoff that isolates the heavy hitters responsible for a large portion of connection attempts in our training set. Finally, we label any TCP flow containing fewer than five packets as a short-lived connection. Previous work treats such ultra-short sessions as abnormal traffic that ends before any meaningful application-layer exchange can occur [64, 65].

Table 2: Thresholds and Parameters Used in Simulation

Parameter	Value	Description
PI	1s	Balances latency and statistical stability; yielded the highest F1-score in a 100 ms–2 s sweep.
APST	1500 B	Corresponds to the typical IP MTU; packets exceeding this value may indicate fragmentation or crafted traffic.
PFT	5 occurrences/ window	Empirical cutoff isolating frequently targeted ports responsible for most connection attempts.
SLCT	< 5 packets/flow	Identifies ultra-short TCP sessions considered abnormal or incomplete exchanges.

PI– Processing Interval; **APST**– Abnormal Packet Size Threshold; **PFT**— Port-Frequency Threshold; **SLCT**— Short-Lived Connection Threshold;

These choices follow best practice in flow-analysis literature (e.g., NetFlow guidelines [60]) and are validated on a held-out subset of our dataset; nevertheless, all of these parameters remain tunable knobs in our implementation. Practitioners can re-estimate them with a grid search or Bayesian optimization to suit higher-speed links or encrypted-traffic mixes. By combining the general features with these carefully parameterized statistical features, we distinguish normal from malicious traffic more effectively.

3.3. Phase 3: ML Approach

In the ML approach, we employ several state-of-the-art methods for detecting malicious traffic. We train 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 large dataset, as shown in Figure 2. M1 directly processes live traffic, and if the confidence value of the detection is below a threshold, the decision is passed to M2, which attempts to perform the prediction (and retrain M1 based on its prediction).

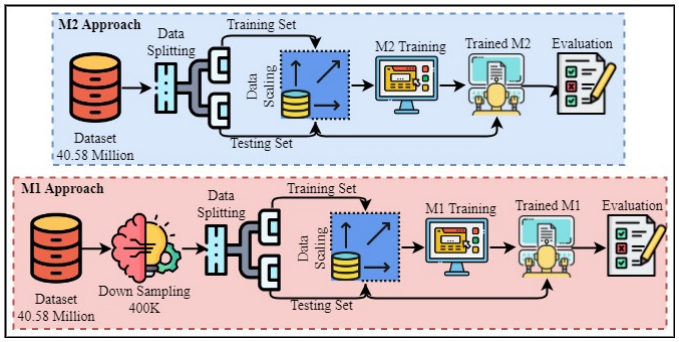


Figure 2: M1 & M2 Models Training Approaches

*3.3.1. **Baseline M1 & M2:***

In the M1 training strategy, we begin by selecting a balanced subset of the data, 100,000 samples per class, totaling 400,000 samples, to create a lightweight yet representative dataset. Initially, we train models using randomly selected samples. To refine the dataset, we then apply K-Means clustering [66] to the standardized features, selecting the top 100,000 samples per class that are closest to their respective cluster centers. This process ensures each class is represented by its most central and diverse instances. After selection, we scale the features using the MinMax method and split the dataset into 75% training and 25% testing. Then, we train four models: *stochastic gradient descent classifier* (SGDC), PER, *multinomial naive Bayes* (MNB), and *Bernoulli naive Bayes* (BNB). We train all models using the partial_fit function to allow continual learning. We evaluate the model's performance using accuracy, precision, recall, and F1 score, along with 10-fold cross-validation for robustness. We select the model hyperparameters based on literature [3, 15, 67] and tune them within specific ranges and values using a trial-and-error approach. We use BNB and MNB with their default settings, which are effective due to their simplicity and low sensitivity to hyperparameters [68]. The following sections provide further analysis of each model and its suitability for the M-En malicious traffic detection task.

Table 3: Hyperparameter Settings For Baseline 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]
PER	max_iter	1000	[200 to 1500]
	tol	1e-3	-
	random_state	42	[0, 42]
MNB	Default	Default	-
BNB	Default	Default	-

⇒**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 [69]. 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: $z = \mathbf{w}\cdot\mathbf{x}+b = w_1x_1+w_2x_2+\ldots+w_nx_n+b$ where, $\mathbf{w} = [w_1, w_2, \ldots, w_n]$ 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:

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

Then SGDC updates weights iteratively, which can be described as: $\mathbf{w} \leftarrow \mathbf{w} - \eta\nabla L(\mathbf{w}, b, \mathbf{x}_i, y_i)$; $b \leftarrow b - \eta\frac{\partial L(\mathbf{w},b,\mathbf{x}_i,y_i)}{\partial b}$ where, η is the learning rate, ∇L is the gradient of the loss function.

⇒**PER:** It is a linear classifier that makes predictions based on a simple threshold function [70]. It is used for binary clas-

sification 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. PER in sci-kit-learn also provides a continuous learning function, so we use it as M1. The PER classifier aims to find a hyperplane that separates the classes. Given an input vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]$, the PER computes the output y using: $z = \mathbf{w} \cdot \mathbf{x} + b = w_1x_1 + w_2x_2 + \ldots + w_nx_n + b$ where, $\mathbf{w} = [w_1, w_2, \ldots, w_n]$ 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:

$$y = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases} \tag{2}$$

⇒**MNB:** It is suitable for large datasets where fast computation is needed, as in our case [71]. It also provides the *'partial_fit'* function for continuous learning, which we use in our approach as the M1 model in comparison with others [72]. 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:

$$P(C_k|\mathbf{x}) = \frac{P(C_k) \cdot P(\mathbf{x}|C_k)}{P(\mathbf{x})} \tag{3}$$

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, and parameters for MNB are estimated using Laplace smoothing.

⇒**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 [73]. It also uses Bayes' theorem like MNB:

$$P(C_k|\mathbf{x}) = \frac{P(C_k) \cdot P(\mathbf{x}|C_k)}{P(\mathbf{x})} \tag{4}$$

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:

$$P(x_i|C_k) = \frac{1}{\sqrt{2\pi\sigma^2_{C_k,i}}} \exp\left(-\frac{(x_i - \mu_{C_k,i})^2}{2\sigma^2_{C_k,i}}\right) \tag{5}$$

where, $\mu_{C_k,i}$ is the mean of x_i in C_k, $\sigma^2_{C_k,i}$ is the variance of x_i in C_k. Further, parameters are estimated using maximum likelihood estimation in GNB.

⇒**Baseline M2:** In the M2 training approach, models were trained on the full dataset of 40 million samples. After applying data scaling, the dataset was split into 75% for training and 25% for testing. We trained various models, including MLP [74], LR [75], SGDC [76], BNB [77], GNB [78], random forest (RF) [79], AdaBoost (ADA) [80], support vector classifier (SVC) [81], and k-nearest neighbors (KNN) [82]. Trained M1 and M2 models were saved using the `pickle` library with a `.pkl` extension for real-time testing. As with M1, M2 models were deployed with their best hyperparameter configurations, determined through literature review and iterative tuning [83, 12, 67]. Table 4 lists the hyperparameter ranges and values used. Models such as BNB, GNB, and MNB were deployed using default settings due to their robustness and minimal sensitivity to hyperparameters.

Table 4: Hyperparameter Settings For Baseline 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]
PER	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	-

3.3.2. *Proposed DL-Based M1 & M2:*

In this study, we propose two DL-based architectures for M1 and M2: a lightweight Transformer (*T1* → M1), a complex Transformer (*T2* → M2), a lightweight LSTM (*L1* → M1), and a complex LSTM (*L2* → M2). We evaluate these models against several baselines. The key differences between the lightweight and complex architectures lie in the size of the training datasets and the number of layers in each architecture.

⇒**Transformer-based Approach:** The *T1* architecture consists of a single dense layer followed by a transformer block with multi-head self-attention and a simple feedforward network. After flattening the attention output, it uses one dense layer before the final softmax classification layer. This design ensures minimal computational overhead, making it suitable for resource-constrained or real-time environments. In contrast, the *T2* architecture includes an additional dense layer at the input, a deeper classification head with three fully connected layers (64 → 32 → 16), and two dropout layers with a rate of 0.5 to mitigate overfitting. Both models share the same transformer

block structure, but the complex version significantly increases representational capacity and depth, enabling improved learning from complex data patterns at the cost of increased training time and memory usage. Figure 3 illustrates the architecture configuration of *T2*, the complex version of *T1*.

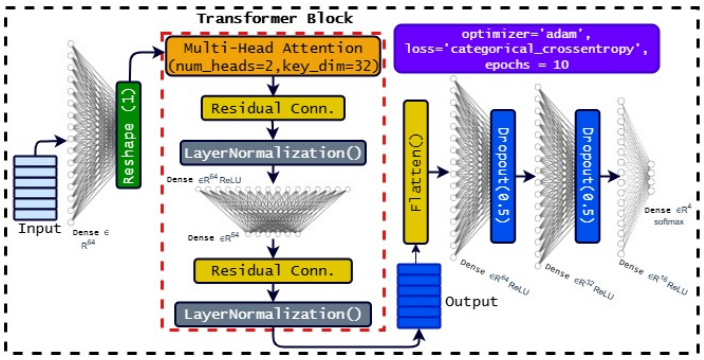


Figure 3: Proposed DL-based T2 Model Architecture

⇒**LSTM-based Approach:** Similarly, the *L1* architecture consists of two LSTM layers: the first with 64 units and the second with 32 units. The final LSTM output is passed through a dense layer with 64 units and a ReLU activation, followed by a dropout layer with a rate of 0.3 before the final softmax classification layer. This configuration is lightweight and well-suited for real-time or resource-constrained environments. In contrast, the *L2* architecture is deeper and more regularized. It begins with an LSTM layer with 64 units, followed by another LSTM layer with 32 units (both with `return_sequences=True`), and a third LSTM layer with 16 units. Between the LSTM layers, dropout layers with a rate of 0.5 are applied to reduce overfitting. The final recurrent output is passed through two dense layers (64 and 32 units, respectively), with an additional dropout of 0.3 before the final softmax output. This deeper and more complex architecture is designed to better capture sequential dependencies in more challenging or high-volume traffic data scenarios.

All models in this study were compiled using the `Adam` optimizer and the `categorical_crossentropy` loss function, which is suitable for multi-class classification tasks. The models were trained for 10 epochs with a batch size of 128. These settings were consistently applied across all Transformer- and LSTM-based architectures.

3.4. Phase 4: Multi-LF *Framework*

In the Multi-LF, we utilize two models: M1 (a lightweight model trained on a small subset of the dataset) and M2 (a more complex model trained on the full dataset). Initially, incoming data is passed to M1 for traffic label prediction. If M1's prediction confidence is 90%, it proceeds to the next sample or terminates the process. If M1's confidence is below 90%, the control shifts to M2, which then performs its own prediction. If M2's confidence exceeds 90%, M1 is retrained using the predicted label from M2 through a continuous learning mechanism. To mitigate the catastrophic forgetting problem during this retraining, we employ a weight interpolation technique, where the updated weights $\mathbf{w}_{\text{new}}$ are computed as: $\mathbf{w}_{\text{new}} = \alpha \mathbf{w}_{\text{updated}} + (1-\alpha)\mathbf{w}_{\text{old}}$,with $\alpha = 0.75$, balancing between the new and old weights. The parameter α controls the trade-off between model adaptability and knowledge retention. In this work, $\alpha = 0.75$ is selected to preserve 75% of prior information while allowing adaptation to new traffic patterns [4]. This value was determined empirically through a series of trials with $\alpha \in 0.25, 0.50, 0.75, 1.0$. Lower values (0.25 and 0.50) caused the model to overfit recent samples and forget previously learned patterns, whereas $\alpha = 1.0$ completely inhibited adaptation to new data. Since our continuous learning setup receives relatively small batches of incoming traffic, $\alpha = 0.75$ provided the most stable and consistent performance, and this configuration is reported in our results. Further, if M2's confidence is also below 90%, control is transferred to a human expert, who labels the traffic sample. M1 is then retrained using the new data and the expert-provided label. This approach enables continuous learning, expert interaction, and ongoing performance enhancement of both models, as illustrated in Figure 4.

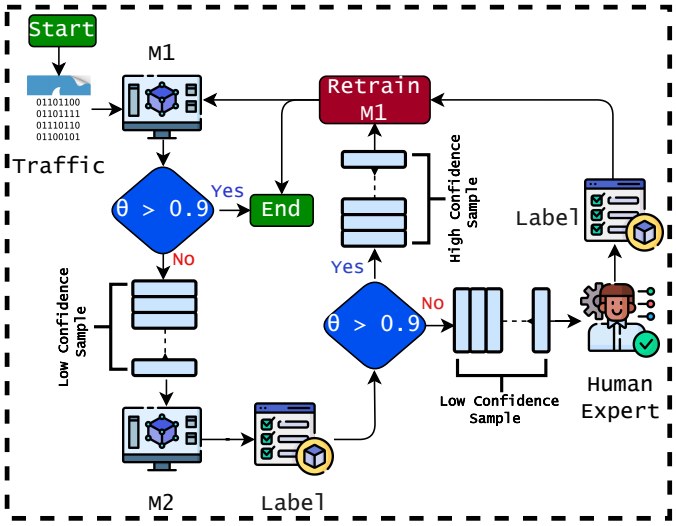


Figure 4: Overview of Multi-LF Flow Diagram

Algorithm 1 describes the continuous learning process. Let $\mathcal{T}$ denote the incoming traffic stream, and x represent a current traffic sample. The models used are M_1 (a lightweight model) and M_2 (a complex model), trained on datasets $\mathcal{D}_s$ (a small subset) and $\mathcal{D}$ (the full dataset), respectively. During prediction, M_1 and M_2 produce predicted labels $\hat{y}_1$ and $\hat{y}_2$, with associated confidence scores γ_1 and γ_2. If confidence is low, a human expert provides a label $\hat{y}_h$. The final output label is denoted by $\hat{y}$, and the confidence threshold for decision-making is set as $\theta = 0.9$.

Algorithm 1 Proposed MULTI-LF Algorithm

Input: Dataset $\mathcal{D}$, Confidence Threshold $\theta = 0.9$
Output: Predicted label $\hat{y}$ for incoming traffic stream $\mathcal{T}$
Initialize: Train model M_1 on subset $\mathcal{D}_s$ (400K samples using K-Means); train model M_2 on full dataset $\mathcal{D}$ (40.58M samples)
while $\mathcal{T}$ is not empty **do**
    $x \leftarrow$ next_sample($\mathcal{T}$)
    $\hat{y}_1, \gamma_1 \leftarrow M_1$.predict($x$)
    if $\gamma_1 \leftarrow \theta$ **then**
        $\hat{y} \leftarrow \hat{y}_1$
    else
        $\hat{y}_2, \gamma_2 \leftarrow M_2$.predict($x$)
        if $\gamma_2 > \theta$ **then**
            $\hat{y} \leftarrow \hat{y}_2$
            M_1.retrain($x, \hat{y}_2$)
        else
            $\hat{y}_h \leftarrow$ get_human_label(x)
            $\hat{y} \leftarrow \hat{y}_h$
            M_1.retrain($x, \hat{y}_h$) ▷ Periodic human updates
        end if
    end if
    Store or use $\hat{y}$
end while

4. Evaluation and Results Discussion

We conduct all training and evaluation on an Intel Core i7-11800H host equipped with 64 GB RAM and a 1 TB SSD. The experiments run inside an Ubuntu 24.04 virtual machine (VMware Workstation) with Python 3.12 and scikit-learn 1.6. To emulate realistic conditions, we replay mixed IoT and traditional traffic through DDoSHIELD-IoT and classify this traffic on a promiscuous Docker node that captures the entire packet stream. We report accuracy, precision, recall, F1 score, runtime, memory footprint, and estimated CO_2 emissions (via CodeCarbon [84]) for experiments.

4.1. Baseline Lightweight M1 Models Results

Table 5 shows results for the baseline M1 models, trained on a 400,000-instance sample using partial fit, with and without min-max scaling. Without scaling, the performance of the models is poor, except for BNB, which achieves an accuracy score of 1.000, and MNB, which reaches an accuracy score of 0.972. In contrast, both SGDC and PER have a very low accuracy scale. These results occur since SGDC and PER 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. However, the performance of all models with scaling is excellent in all evaluation metrics, with all models achieving scores of 1.00. Scaling ensures that all features contribute proportionately to the model, enhancing the training efficiency and effectiveness of SGDC and PER classifiers. Additionally, scaling normalizes the feature space, leading to more balanced weight updates and improved model performance. For instance, if packet sizes range from 0 to 10,000 and another feature, such as traffic rate, ranges from 0 to 1, the larger feature disproportionately influences the model's parameters, leading to suboptimal performance. Here, scaling mitigates this problem by normalizing the features. Similarly, the MNB and BNB models assume feature independence and use probabilities based on the occurrence of features. Without scaling, the probability estimates can become skewed, affecting the model's ability to correctly classify instances.

Table 5: Baseline M1 Models Results Using M-En Data

	With Scaling					Without Scaling				
Model	**Acc**	**Class**	**Pre**	**Rec**	**F1**	**Acc**	**Class**	**Pre**	**Rec**	**F1**
		IoT.B	1.00	1.00	1.00		IoT.B	0.25	1.00	0.40
		IoT.M	1.00	1.00	1.00		IoT.M	0.00	0.00	0.00
SGDC	1.000	Tr.B	1.00	1.00	1.00	0.250	Tr.B	0.00	0.00	0.00
		Tr.M	1.00	1.00	1.00		Tr.M	0.00	0.00	0.00
		Avg.	1.00	1.00	1.00		Avg.	0.06	0.25	0.10
		IoT.B	1.00	1.00	1.00		IoT.B	0.00	0.00	0.00
		IoT.M	1.00	1.00	1.00		IoT.M	0.00	0.00	0.00
PER	1.000	Tr.B	1.00	1.00	1.00	0.250	Tr.B	0.25	1.00	0.40
		Tr.M	1.00	1.00	1.00		Tr.M	0.00	0.00	0.00
		Avg.	1.00	1.00	1.00		Avg.	0.06	0.25	0.10
		IoT.B	1.00	1.00	1.00		IoT.B	0.99	1.00	1.00
		IoT.M	1.00	1.00	1.00		IoT.M	1.00	1.00	1.00
MNB	1.000	Tr.B	1.00	1.00	1.00	0.972	Tr.B	1.00	0.90	0.95
		Tr.M	1.00	1.00	1.00		Tr.M	0.91	0.99	0.95
		Avg.	1.00	1.00	1.00		Avg.	0.97	0.97	0.97
		IoT.B	1.00	1.00	1.00		IoT.B	1.00	1.00	1.00
		IoT.M	1.00	1.00	1.00		IoT.M	1.00	1.00	1.00
BNB	1.000	Tr.B	1.00	1.00	1.00	1.000	Tr.B	1.00	1.00	1.00
		Tr.M	1.00	1.00	1.00		Tr.M	1.00	1.00	1.00
		Avg.	1.00	1.00	1.00		Avg.	1.00	1.00	1.00

Figure 5 shows the 10-fold cross-validation performance of M1. Each model was evaluated with and without min-max scaling. Without scaling, SGDC and PER performed poorly (F1 = 0.10 ± 0.00) due to their sensitivity to feature magnitude, which hinders gradient descent convergence. After scaling, both achieved perfect F1 = 1.00 ± 0.00. Scaling normalizes feature ranges, such as packet size and traffic rate, ensuring balanced weight updates and improved performance. MNB and BNB also benefited from scaling: MNB improved from 0.97 ± 0.04 to 1.00 ± 0.00, while BNB maintained perfect performance in both cases, demonstrating its robustness. Consequently, BNB is selected as the M1 model in our approach.

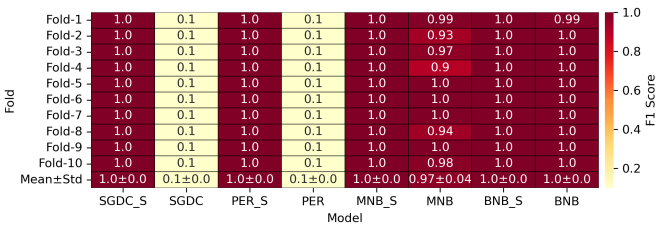


Figure 5: Baseline M1 Models F1 Score using K-Fold Cross Validation

To validate the effectiveness of our feature extraction configuration and threshold settings, we conducted experiments using three distinct parameter scenarios. Each scenario varied the

processing interval (PI), abnormal size threshold (AST), port frequency threshold (PFT), and short-lived threshold (SLT), all of which directly influence flow-based feature computation from PCAP files. As shown in Figure 6, all models achieved perfect accuracy under our balanced configuration, confirming its suitability. PER, SGDC, and MNB exhibited strong robustness across scenarios, while BNB showed greater sensitivity to threshold changes. These results underscore the importance of carefully tuning feature aggregation parameters and validate the reliability and generalizability of our selected configuration.

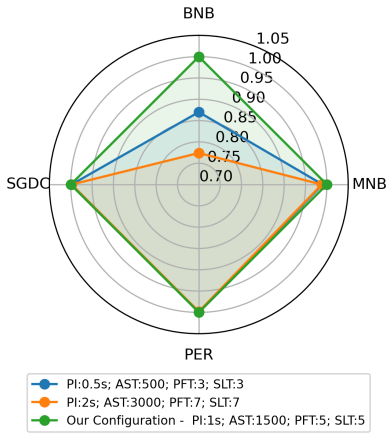


Figure 6: Accuracies Across Three Feature Extraction configurations

4.2. Baseline M2 Models Results

The performance of M2 models is shown in Table 6. 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 baseline approach for MULTI-LF 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 7. 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 PER 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 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 6: Results For Baseline M2 Models Using M-En Data

Model	Acc	Class	Pre	Rec	F1	Model	Acc	Class	Pre	Rec	F1
LR	0.999	IoT.B	1.00	1.00	1.00	BNB	0.831	IoT.B	0.82	1.00	0.90
		IoT.M	1.00	1.00	1.00			IoT.M	0.99	0.74	0.85
		Tr.B	1.00	1.00	1.00			Tr.B	0.80	1.00	0.89
		Tr.M	1.00	1.00	1.00			Tr.M	0.94	0.21	0.34
		Avg	1.00	1.00	1.00			Avg	0.89	0.73	0.74
SGDC	0.998	IoT.B	1.00	1.00	1.00	GNB	0.998	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	0.99	0.99			Tr.B	1.00	0.99	1.00
		Tr.M	1.00	1.00	1.00			Tr.M	1.00	1.00	1.00
		Avg	1.00	1.00	1.00			Avg	1.00	1.00	1.00
PER	0.999	IoT.B	1.00	1.00	1.00	RF	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
		Avg	1.00	1.00	1.00			Avg	1.00	1.00	1.00
MNB	0.994	IoT.B	1.00	1.00	1.00	ADA	0.998	IoT.B	1.00	1.00	1.00
		IoT.M	0.99	0.99	0.99			IoT.M	1.00	0.99	0.99
		Tr.B	0.99	0.99	0.99			Tr.B	0.99	0.99	0.99
		Tr.M	0.99	1.00	0.99			Tr.M	1.00	1.00	1.00
		Avg	0.99	0.99	0.99			Avg	1.00	1.00	1.00

Table 7: Baseline M2 Models Performance Metrics

Model	CP	WP	Error Rate
LR	10,140,676	5,395	0.0005
SGDC	10,128,182	17,889	0.0018
PER	10,140,625	5,446	0.0005
MNB	10,088,644	57,427	0.0057
BNB	8,437,084	1,708,987	0.1684
GNB	10,130,116	15955	0.0016
RF	10,146,052	19	0.0000
ADA	10,132,709	13362	0.0013

Table 8 shows the computational time required for the training and testing of baseline 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, PER, 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 8: Baseline M1 & M2 Models Computational Time

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

4.3. 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 Multi-LF. 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. (*i*) ***Scenario 1:*** M1 is used for attack detection without continuous learning capabilities and evaluated with benign and malicious traffic. (*ii*) ***Scenario 2:*** M1 is used for attack detection with continuous learning capabilities. (*iii*) ***Scenario 3:*** M2 is used independently for testing. (*iv*) ***Scenario 4:*** M1 and M2 are deployed together under Multi-LF without human involvement. (*v*) ***Scenario 5:*** M1 and M2 are deployed under Multi-LF with human involvement. The implementation of Scenarios 4 and 5 in Multi-LF is illustrated in Figure 7.

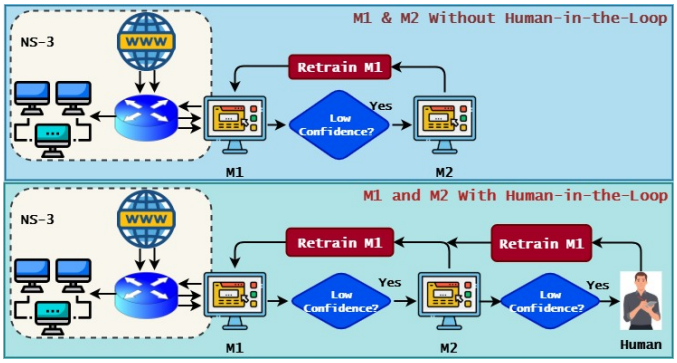


Figure 7: *Scenario 4 & 5* Visualization For Easy Understanding

Table 9 provides an evaluation of ***Scenario 1***, where the M1 model is used without continuous learning for attack detection. KNN demonstrates the highest accuracy of 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 exhibit a much smaller size of 2 KB, a lower MU of 122.3 MB, and a minimal CPU impact of 45.61% with a faster prediction time of 0.0007 seconds, with reduced accuracy of 0.536. These results highlight the trade-off between model complexity and computational efficiency, making KNN suitable for accuracy-critical scenarios, while lighter models, such as LR, are better suited for resource-constrained environments.

Table 9: **Scenario 1:** Baseline 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

Scenario 2 evaluates the M1 model with continuous learning capabilities, showcasing improvements in both efficiency and performance as shown in Table 10. Among the models tested, PER achieves the highest accuracy of 0.704 while maintaining a minimal MS of 3 KB and a low prediction time of 0.0007 seconds. These results demonstrate the model's ability to quickly adapt to new data without a significant increase in computational costs. Other models, such as MNB and SGDC, also display competitive accuracies of 0.594 and 0.581, respectively, while maintaining a low memory footprint of 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 achieves an accuracy of 0.704 compared to ***Scenario 1***, which has an accuracy of 0.688.

Table 10: **Scenario 2:** Baseline M1 With Continuous Learning

Model	BNB	MNB	PER	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 machine learning models, as shown in Table 11. The results show notable variation in accuracy, prediction time, CPU utilization, and memory usage. MLP achieves the highest accuracy of 0.885 but requires high CPU utilization of 57.14% and memory usage of 140.12 MB. LR offers balanced performance with an accuracy of 0.818 and low computational cost, making it suitable for resource-constrained environments. RF attains moderate accuracy of 0.738 but consumes the most resources, with an MS of 33,009 KB and MU of 267.68 MB, highlighting a trade-off between accuracy and efficiency. BNB and GNB achieve accuracies of 0.725 and 0.721 respectively, maintaining low prediction times and memory use, though their high CPU utilization limits real-time applicability. Compared to ***Scenario 2***, where continuous learning was employed, ***Scenario 3*** shows that M2 can achieve higher accuracy with some models, such as MLP, but at the cost of increased computational overhead. This emphasizes the importance of continuous learning for optimizing resource usage and ensuring real-time performance. Consequently, M1 with continuous learning is applied at the first level, and the best M2 model is deployed at the second level in the Multi-LF.

Table 11: **Scenario 3:** Baseline M2 Performance Comparison

Model	MS (KB)	Acc	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
PER	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 12 shows the performance comparison of four models, BNB, MNB, PER, 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 M2 because of its significant performance. The PER 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 computational intensity. In contrast, BNB maintains high accuracy at 0.975 with a much lower CPU usage of 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 of 0.983, achieved by the PER model. However, the computational overhead is significantly reduced, as evidenced by lower CPU utilization, 3.221%, and MU 0.956 MB. These results suggest 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 performance, efficiency trade-offs between using MLP and RF as M2 models in MULTI-LF. The PER 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 12: **Scenario 4:** Baseline M1 and M2 Under MULTI-LF Without Human-in-the-Loop

MLP → M2 & Without Human-in-the-Loop				
Matrix	BNB	MNB	PER	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 → M2 & Without Human-in-the-Loop				
Matrix	BNB	MNB	PER	SGDC
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. We used BNB, MNB, PER, and SGDC as M1 with MLP and RF as the M2, same as ***Scenario 4***. This scenario demonstrated the impact of incorporating human oversight on model performance. When MLP was used as the M2, the PER achieved the highest accuracy of 0.999, but it required significant CPU utilization of 17.84% and MU of 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 of 56.53 MB remained notably high compared to other models. When RF was used as the M2, PER again demonstrated high accuracy of 0.983, but with a reduced CPU utilization of 10.05% and a lower MU of 3.632 MB. Prediction time was also reduced to 0.434 seconds, indicating that RF as the M2 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 of 72.69 MB. Comparing ***Scenario 5*** to ***Scenario 4*** under 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 M2, highlighting the advantage of human intervention in refining the model's performance.

Table 13: **Scenario 5:** Baseline M1 and M2 Under MULTI-LF With Human-in-the-Loop

MLP → M2 & Human-in-the-Loop				
Matrix	BNB	MNB	PER	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 → M2 & Human-in-the-Loop				
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

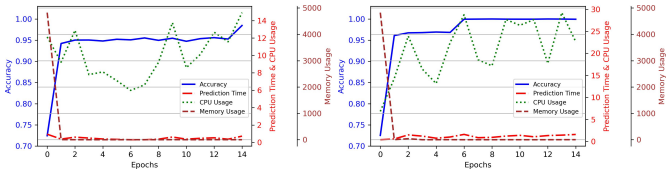
Human Intervention Feasibility: In our experimental setup, approximately 111,644 samples are processed across 15 batches. Human intervention is required for only 109 samples, resulting in a human effort percentage of just 0.0976%. This behavior shows that the approach is highly feasible for real-time deployment, as the burden on human input remains minimal, as shown in Table 14.

Table 14: Human Workload Under MULTI-LF Using Baseline M1 & M2

Metric	**M1 → M2 → Human**
M1 TRP	111,644
M1 CPP	111,153
M2 TRP	491
M2 CPP	382
HIR	109
HE	0.0976%

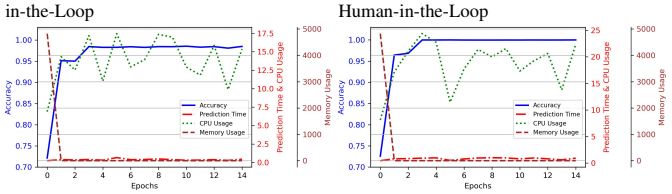
TRP– Total Received Packets; **CPP**– Correctly Predicted Packets; **HIR**– Human Intervention Required; **HE**– Human Efforts

Figure 8 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.



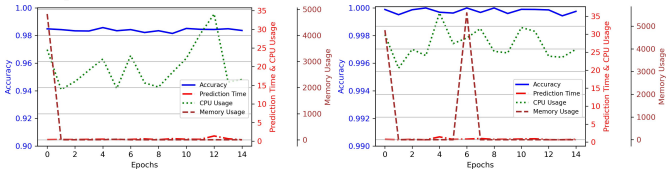
(a) M1 (BNB) & M2 (RF) without Human-in-the-Loop

(b) M1 (BNB) & M2 (MLP) without Human-in-the-Loop



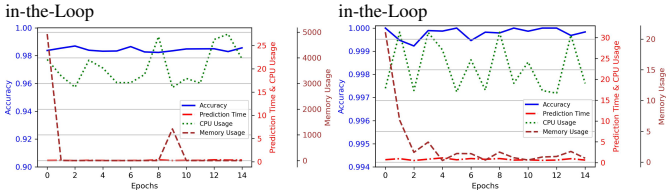
(c) M1 (BNB) & M2 (RF) with Human-in-the-Loop

(d) M1 (BNB) & M2 (MLP) with Human-in-the-Loop



(e) M1 (PER) & M2 (RF) without Human-in-the-Loop

(f) M1 (PER) & M2 (MLP) without Human-in-the-Loop



(g) M1 (PER) & M2 (RF) with Human-in-the-Loop

(h) M1 (PER) & M2 (MLP) with Human-in-the-Loop

Figure 8: Per-Batch Results of Best Models For *Scenarios 4* and *5*

4.4. Proposed DL Models Results Under Multi-LF *Framework*

In addition to traditional machine learning models, we implemented several deep learning architectures in our proposed Multi-LF to enhance malicious traffic detection performance. These include T1, T2, L1, and L2, where T1 and L1 represent lightweight models (used as M1) and T2 and L2 represent more complex variants (used as M2). Deep learning models showed significant performance improvements compared to traditional baselines, largely due to their ability to capture long-term dependencies and complex patterns in the data. The lightweight variants were specifically designed to maximize throughput while maintaining high accuracy. Table 15 shows DL models' performance achieved nearly identical score values, exceeding 0.999, indicating robust and reliable performance. The lightweight models (T1 and L1) show higher throughput, especially T1, which achieved a throughput of 66,130 packets/sec, significantly higher than L1 (6,157 packets/sec) and the complex models T2 and L2. Notably, T1 had a slightly higher error rate of 0.0651, compared to L1, which is 0.0612, but the difference is minimal and offset by its efficiency advantage.

Table 16 further summarizes the performance of our Multi-LF under human-in-the-loop settings. Both the T1 → T2 and L1 → L2 configurations correctly handled nearly all incoming samples at the M1 stage (115,789 and 116,669 correctly predicted packets, respectively). Only a few packets required M2-level processing (6 for T1, 4 for L1), and even fewer required human intervention (3 for T1, none for L1). The *human effort* (HE) remained minimal in both cases, just 0.0026% for the Transformer-based path and 0.0000% for the LSTM-based path. Due to the higher throughput and lower error rate, we selected the Transformer-based architecture for deployment over the LSTM-based one. The minimal involvement required from human experts proves that our approach is both feasible and effective for real-time scenarios.

Table 15: Results Using Proposed DL-based Models

Model	Acc	Pre	Rec	F1	Error Rate	Throughput
T1	0.9993	0.9994	0.9993	0.9993	0.0651	66130
T2	0.9994	0.9994	0.9994	0.9994	0.0612	65595
L1	0.9994	0.9994	0.9994	0.9994	0.0612	6157
L2	0.9994	0.9995	0.9994	0.9994	0.0611	4263

Table 16: Human Workload Under Multi-LF Using DL-based M1 & M2

Metric	T1 → T2 → Human	L1 → L2 → Human
M1 TRP	115,795	116,673
M1 CPP	115,789	116,669
M2 TRP	6	4
M2 CPP	3	4
HIR	3	0
HE	0.0026 %	0.0000 %

TRP– Total Received Packets; **CPP**– Correctly Predicted Packets; **HIR**— Human Intervention Required; **HE**— Human Efforts

Further, to demonstrate the significance and sustainability of our proposed approach under the Multi-LF framework, we measured the environmental impact and computational cost using the `CodeCarbon`[4] library. Table 17 presents the carbon emissions and energy consumption metrics for each deep learning model variant. The results show that the lightweight models (T1 and L1) are notably more energy-efficient and environmentally friendly than their complex counterparts (T2 and L2). For instance, the T1 model consumed only 0.0242 kWh of energy and emitted just 0.0070 kg of CO_2, while the L2 model, being the most complex, used 0.1074 kWh and emitted 0.0312 kg of CO_2.

Table 17: Environmental Impact Statistics of DL-based Models

Model	CO_2	CPU_E	RAM_E	Total_E	CPU_P
T1	0.0070	0.0165	0.0078	0.0242	42.5
T2	0.0134	0.0314	0.0148	0.0462	42.5
L1	0.0158	0.0369	0.0174	0.0543	42.5
L2	0.0312	0.0730	0.0343	0.1074	42.5

CO_2– CO_2 Emission (KG); **CPU_E**– CPU Energy (kWh); **RAM_E**– RAM Energy (kWh); **Total_E**–Total Energy (kWh); **CPU_P**– CPU Power (W)

4.5. Results using Other Benchmark Datasets

This section presents the performance of the proposed Multi-LF framework using two benchmark datasets, M-En Dataset-1 [15] and M-En Dataset-2 [12], which are synthetically created by combining well-known intrusion detection datasets.

[4]https://codecarbon.io/

Dataset-1 is a combination of IoTID-20, UNSW-NB15, and InSDN, while Dataset-2 consists of IoTID-20 and UNSW-NB15. The results show that the M1 correctly classifies the majority of packets, while the M2 effectively handles low-confidence cases passed from M1, as shown in Table 18. Only a small fraction of packets require human intervention, with human effort reduced to as low as 0.29% in Dataset-2. These significant results show the effectiveness of the proposed Multi-LF in maintaining high accuracy while substantially reducing human workload across different benchmark datasets.

Table 18: Proposed Multi-LF Using Benchmark Datasets

Metric	Dataset-1 [15]	Dataset-2 [12]
M1 TRP	20,865	22,680
M1 CPP	19,428	22,481
M2 TRP	1249	85
M2 CPP	240	18
HIR	951	67
HE	4.5485 %	0.2954%

TRP– Total Received Packets; **CPP**– Correctly Predicted Packets; **HIR**— Human Intervention Required; **HE**— Human Efforts;

4.6. Comparison With Benchmark Studies

Many studies validate their malicious traffic detection 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. [52] deployed their approach on an M-En dataset using the 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 [15, 12] proposed optimization-based approaches to handle diverse traffic patterns in M-En networks, employing PSO and MFO, respectively. These studies introduced PSO-D-SEM and a *Fully Automated Malicious Traffic Detection System* (FAMTDS). Another study [56] 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. [54] 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. Furthermore, we deployed modern GNN-based and transformer-based architectures to compare our approach with advanced approaches [23, 24]. In contrast, our study conducted both online and offline testing and was built on a real-time M-En dataset.

In Table 19, we compare our approach, Multi-LF, to relevant M-En benchmark studies. Most existing approaches [52, 15, 12, 56, 54] 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 retraining 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 malicious traffic detection in M-En environments.

Table 19: Comparison With Existing Studies

Ref.	Year	Approach	Testing		Results
			offline	online	
[52]	2023	ETC, S-DATE	✓	×	0.986
[15]	2024	PSO-D-SEM	✓	×	0.978
[12]	2024	FAMTDS	✓	×	0.991
[56]	2024	DDT-LightGBM	✓	×	0.980
[54]	2024	Meta-Fed IDS	✓	×	0.94
[23]	2025	GTAE-IDS	✓	×	0.968
[24]	2025	GSAGE+RF	✓	×	0.99
Our	2025	Multi-LF	✓	✓	1.00, 0.999

5. Discussion

In this study, we collect a dataset in the M-En and propose Multi-LF for malicious traffic detection, testing it in both offline and real-time scenarios. We deploy Multi-LF to reduce computational costs and enhance accuracy over time. We reduce the computational cost because the initial traffic is evaluated by the lightweight M1 model, which has a faster prediction time due to its lightweight architecture. 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, we improve the approach's performance by incorporating an extra layer of security using M2 and human involvement.

We achieve continuous improvement in model performance over time through continuous training with new data, enabling the model to efficiently adapt to evolving traffic patterns, as well as through our feature engineering strategy. We utilize general and statistical features, as they both significantly contributed to the model's effectiveness, as illustrated in Figure 9. Statistical features, such as *ConnectionErrors*, *DstPortEntropy*, *MostFreqPayloadSize*, *SourceEntropy*, *MostFreqPacketSizeFreq*, *FlowRate*, *PacketCount*, *PacketSizeVar*, and *AvgPayloadSize*, played a critical role in detecting malicious traffic according to the importance score. For instance, a sudden spike in packet count from a specific IP within a short time window may signal an attack. Additionally, some general features, such as *SYN* and *ACK* flags, are also among the top contributors for malicious traffic detection. These statistical and

general features together provided highly correlated input representations, as shown in Figure 9, thereby enhancing the overall performance of the framework.

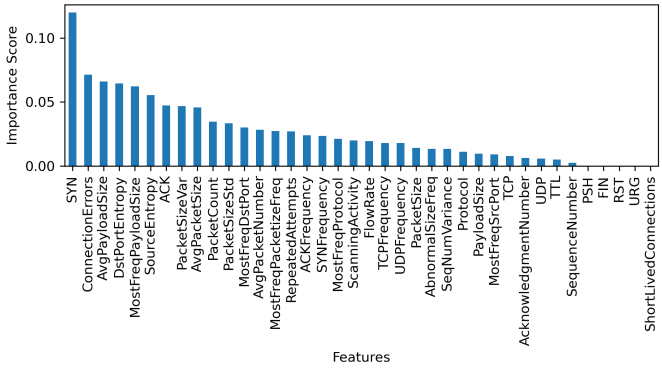


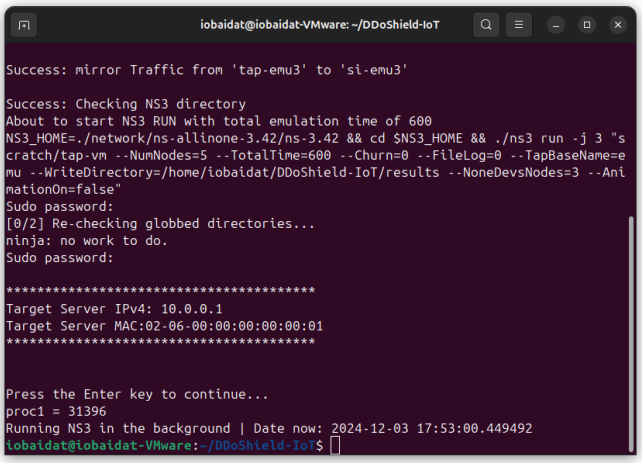
Figure 9: Feature Importance Score

To justify the need for a robust AI solution for M-En networks and the limitations of existing datasets, we conducted cross-domain experiments. Table 20 shows performance variations across scenarios, underscoring the value of a unified M-En-based approach. 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 20: Results Using Different Training & Testing Datasets

Training	Testing	Acc	Class	Pre	Rec	F1
IoT	Trad.	0.006	0	0.01	0.03	0.01
			1	0.00	0.00	0.00
Trad.	IoT	0.014	0	0.13	0.02	0.03
			1	0.00	0.00	0.00
M-En	IoT & Trad.	0.999	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.



(a) Server Connection Establishment

(b) Live Performance of Models and Continuous Improvement

Figure 10: Real-time Testing Snapshots

⇒ **Significance:** This study has several significances in network security for malicious attack detection and shows a strong contribution in the given domain: ***(i)*** 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. ***(ii)*** 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. ***(iii)*** 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. ***(iv)*** Multi-LF ability to leverage a lightweight model (M1) for initial detection significantly reduces computational overhead, which is crucial for real-time applications. This workflow 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: **(*i*)** The present work focuses on IoT and traditional networks within the M-En framework. This controlled scope ensures methodological clarity and reproducibility. However, the framework is inherently extensible and can seamlessly integrate additional domains, such as SDN and *consumer electronics* (CE) networks, during real-time deployment, enabling broader applicability in larger and more heterogeneous infrastructures. **(*ii*)** Multi-LFincorporates human oversight to validate critical or uncertain detections. While this may introduce latency under high-alert conditions, it enhances interpretability and operational reliability, which are key for security in sensitive environments. Notably, studies and industry reports indicate that a human analyst typically manages 100–200 alerts per day[5], underscoring the value of incorporating guided automation to reduce analyst workload. Future work will focus on integrating active learning and feedback–driven tuning to streamline this process further. **(*iii*)** The continual learning strategy retrains the model using previously misclassified samples, enhancing adaptability to evolving threats. Although this iterative update process increases computational demand over time, it ensures the model remains robust against concept drift. Future optimization, such as lightweight retraining or selective memory replay, can maintain adaptability while minimizing resource overhead.

6. Conclusion & Future Direction

In this paper, we first constructed a packet-level M-En dataset (from live DDoShield-IoT traffic mixed with public PCAP traces) to provide the research community with a realistic corpus and a reproducible pipeline for heterogeneous M-En attack research. Building on this foundation, we presented Multi-LF, a continuous-learning framework that unifies lightweight and deep architecture models and incorporates selective human feedback to secure M-En networks. We deployed Multi-LF online in DDoShield-IoT using a Transformer-based M1 & M2, maintaining 0.999 accuracy while requiring human intervention to label only 0.0026% for 115,795 live packets. In contrast, the baseline approach requires significantly more human intervention, 109 expert checks out of 111,644 packets, which represents 0.0976% of the workload. This online evaluation confirms a high detection rate under real traffic, while the chosen transformer models provide high throughput and consume low power, resulting in 0.007 kg CO_2 emission per experiment. We concluded that traditional models suffer a drop in accuracy during online testing, whereas our Multi-LFeffectively handles live traffic through layered verification and minimal human intervention, achieving near error-free performance. Additionally, our statistical feature extraction configuration, using a 1-second processing interval, proved effective. Features like *average payload size* and *destination port entropy* significantly improved malicious traffic detection. Further, our findings show that continuous learning with weight interpolation enables the M1 model to adapt to new traffic and attacks while preserving prior knowledge, minimizing human effort.

In future work, we aim to expand the dataset with encrypted and zero-trust traffic and explore unsupervised drift detection to enable adaptive retraining. We also plan to enrich DDoShield-IoT by injecting additional binaries and diverse attack types, along with augmenting it using public PCAP repositories via our modular pipeline. To evaluate realism, we will compute entropy and protocol-distribution metrics and benchmark them against backbone traces (e.g., MAWI [85]). Finally, we intend to explore the integration of *Large Language Models* (LLMs) to support or automate human-in-the-loop decisions for improved speed and reliability.

[5] https://countershadow.com/blog/ai-vs-cyber-threats-accelerating-response-in-a-24-7-landscape/

Data Availability Statement

The code and datasets used for reproducibility of the results in this study are publicly available on GitHub at `https://github.com/furqanrustam/MultiLF`.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

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

Acknowledgment

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

References

[1] Ziming Zhao, Zhuotao Liu, Huan Chen, Fan Zhang, Zhuoxue Song, and Zhaoxuan Li. Effective ddos mitigation via ml-driven in-network traffic shaping. *IEEE Transactions on Dependable and Secure Computing*, pages 1–18, 2024.

[2] Ahsan Nazir, Jingsha He, Nafei Zhu, Ahsan Wajahat, Xiangjun Ma, Faheem Ullah, Sirajuddin Qureshi, and Muhammad Salman 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*, 35(10):101820, 2023.

[3] Furqan Rustam, Pasika S Ranaweera, and Anca Delia Jurcut. Ai on the defensive and offensive: Securing multi-environment networks from ai agents. In *Proceedings of the IEEE International Conference on Communications (ICC)*, page To be assigned, USA, March 2024.

[4] Furqan Rustam and Anca Delia Jurcut. Adaptive security framework for multi-environment networks using ensemble data drift detection and incremental deep learning. *Journal of Information Security and Applications*, 94:104219, 2025.

[5] Salih Ismail, Hani Ragab Hassen, Mike Just, and Hind Zantout. A review of amplification-based distributed denial of service attacks and their mitigation. *Computers & Security*, 109:102380, 2021.

[6] An Wang, Wentao Chang, Songqing Chen, and Aziz Mohaisen. A data-driven study of ddos attacks and their dynamics. *IEEE Transactions on Dependable and Secure Computing*, 17(3):648–661, 2020.

[7] Arash Mahboubi, Khanh Luong, Hamed Aboutorab, Hang Thanh Bui, Geoff Jarrad, Mohammed Bahutair, Seyit Camtepe, Ganna Pogrebna, Ejaz Ahmed, Bazara Barry, and Hannah Gately. Evolving techniques in cyber threat hunting: A systematic review. *Journal of Network and Computer Applications*, 232:104004, 2024.

[8] Simona De Vivo, Islam Obaidat, Dong Dai, and Pietro 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)*, pages 1–8, 2024.

[9] Yongyi Cao, Hao Jiang, Yuchuan Deng, Jing Wu, Pan Zhou, and Wei Luo. Detecting and mitigating ddos attacks in sdn using spatial-temporal graph convolutional network. *IEEE Transactions on Dependable and Secure Computing*, 19(6):3855–3872, 2022.

[10] Jalal Bhayo, Syed Attique Shah, Sufian Hameed, Awais Ahmed, Jamal Nasir, and Dirk Draheim. Towards a machine learning-based framework for ddos attack detection in software-defined iot (sd-iot) networks. *Engineering Applications of Artificial Intelligence*, 123:106432, 2023.

[11] Mohammad Najafimehr, Sajjad Zarifzadeh, and Seyedakbar Mostafavi. Ddos attacks and machine-learning-based detection methods: A survey and taxonomy. *Engineering Reports*, 5(12):e12697, 2023.

[12] Furqan Rustam, Wajdi Aljedaani, Mahmoud Said Elsayed, and Anca Delia Jurcut. Famtds: A novel mfo-based fully automated malicious traffic detection system for multi-environment networks. *Computer Networks*, 251:110603, 2024.

[13] Imtiaz Ullah and Qusay H. Mahmoud. A scheme for generating a dataset for anomalous activity detection in iot networks. In Cyril Goutte and Xiaodan Zhu, editors, *Advances in Artificial Intelligence*, pages 508–520, Cham, 2020. Springer International Publishing.

[14] Nour Moustafa and Jill 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)*, pages 1–6, 2015.

[15] Furqan Rustam and Anca Delia Jurcut. Malicious traffic detection in multi-environment networks using novel s-date and pso-d-sem approaches. *Computers & Security*, 136:103564, 2024.

[16] George F Riley and Thomas R Henderson. The ns-3 network simulator. In *Modeling and tools for network simulation*, pages 15–34. Springer, 2010.

[17] Dirk Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, 2014(239):2, 2014.

[18] Mulualem Bitew Anley, Angelo Genovese, Davide Agostinello, and Vincenzo Piuri. Robust ddos attack detection with adaptive transfer learning. *Computers & Security*, page 103962, 2024.

[19] Iman Sharafaldin, Arash Habibi Lashkari, Saqib Hakak, and Ali A. Ghorbani. Developing realistic distributed denial of service (ddos) attack dataset and taxonomy. In *2019 International Carnahan Conference on Security Technology (ICCST)*, pages 1–8, 2019.

[20] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *International Conference on Information Systems Security and Privacy*, 2018.

[21] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*, pages 1–6. Ieee, 2009.

[22] Amal M. Al-Eryani, Eman Hossny, and Fatma A. Omara. Efficient machine learning algorithms for ddos attack detection. In *2024 6th International Conference on Computing and Informatics (ICCI)*, pages 174–181, 2024.

[23] Jalal Ghadermazi, Soumyadeep Hore, Ankit Shah, and Nathaniel D. Bastian. Gtae-ids: Graph transformer-based autoencoder framework for real-time network intrusion detection. *IEEE Transactions on Information Forensics and Security*, 20:4026–4041, 2025.

[24] Guan-Yan Yang, Farn Wang, and Kuo-Hui Yeh. Gnn-enhanced traffic anomaly detection for next-generation sdn-enabled consumer electronics. *IEEE Transactions on Consumer Electronics*, pages 1–1, 2025.

[25] Avtar Singh, Harpreet Kaur, and Navjot Kaur. A novel ddos detection and mitigation technique using hybrid machine learning model and redirect illegitimate traffic in sdn network. *Cluster Computing*, 27(3):3537–3557, 2024.

[26] Abdul Waleed, Abdul Fareed Jamali, and Ammar Masood. Which open-source ids? snort, suricata or zeek. *Computer Networks*, 213:109116, 2022.

[27] Md Alamin Talukder, Md Manowarul Islam, Md Ashraf Uddin, Khondokar Fida Hasan, Selina Sharmin, Salem A Alyami, and Mohammad Ali Moni. Machine learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction. *Journal of Big Data*, 11(1):33, 2024.

[28] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. A detailed analysis of the cicids2017 data set. In *Information Systems Security and Privacy: 4th International Conference, ICISSP 2018, Funchal-Madeira, Portugal, January 22-24, 2018, Revised Selected Papers 4*, pages 172–188. Springer, 2019.

[29] Mohammad Hafiz Mohd Yusof, Akram A. Almohammedi, Vladimir Shepelev, and Osman Ahmed. Visualizing realistic benchmarked ids dataset: Cira-cic-dohbrw-2020. *IEEE Access*, 10:94624–94642, 2022.

[30] V Sri Vigna Hema, S Devadharshini, and P Gowsalya. Malicious traffic flow detection in iot using ml based algorithms. *International Research Journal on Advanced Science*, 3(5):68–76, 2023.

[31] Zequn Niu, Jingfeng Xue, Yong Wang, Tianwei Lei, Weijie Han, and Xianwei Gao. Qarf: A novel malicious traffic detection approach via online active learning for evolving traffic streams. *Chinese Journal of Electronics*, 33(3):645–656, 2024.

[32] Abiodun Esther Omolara, Abdullah Alabdulatif, Oludare Isaac Abiodun, Moatsum Alawida, Abdulatif Alabdulatif, Humaira Arshad, et al. The internet of things security: A survey encompassing unexplored areas and new insights. *Computers & Security*, 112:102494, 2022.

[33] Moemedi Lefoane, Ibrahim Ghafir, Sohag Kabir, and Irfan-Ullah Awan. Internet of things botnets: A survey on artificial intelligence based detection techniques. *Journal of Network and Computer Applications*, 236:104110, 2025.

[34] Bilal Babayigit and Mohammed 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*, 128:107515, 2024.

[35] Mohamed Amine Ferrag, Othmane Friha, Djallel Hamouda, Leandros Maglaras, and Helge Janicke. Edge-iiotset: A new comprehensive realistic cyber security dataset of iot and iiot applications for centralized and federated learning. *IEEE Access*, 10:40281–40306, 2022.

[36] M. Zolanvari, M. A. Teixeira, L. Gupta, K. M. Khan, and R. Jain. WUSTL-IIOT-2021 Dataset for IIoT Cybersecurity Research, October 2021.

[37] Muna Al-Hawawreh, Elena Sitnikova, and Neda Aboutorab. X-iiotid: A connectivity-agnostic and device-agnostic intrusion data set for industrial internet of things. *IEEE Internet of Things Journal*, 9(5):3962–3977, 2022.

[38] Shizhou Zhu, Xiaolong Xu, Juan Zhao, and Fu Xiao. Lkd-stnn: A lightweight malicious traffic detection method for internet of things based on knowledge distillation. *IEEE Internet of Things Journal*, 11(4):6438–6453, 2024.

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

[40] Sebastian Garcia, Agustin Parmisano, and Maria Jose Erquiaga. Iot-23: A labeled dataset with malicious and benign iot network traffic. *Stratosphere Lab., Praha, Czech Republic, Tech. Rep*, 2020.

[41] Yuehua Huo, Wei Liang, Junhan Chen, Shangyuan Zhuang, and Jiyan Sun. Lightguard: A lightweight malicious traffic detection method for internet of things. *IEEE Internet of Things Journal*, pages 1–1, 2024.

[42] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 122–138, Cham, 2018. Springer International Publishing.

[43] Wang Wei and Lu David. USTC-TFC2016: A Dataset for Traffic Classification, 2016. Accessed: 2023-10-25.

[44] University of New Brunswick Canadian Institute for Cybersecurity. CIC IoT Dataset Collection, 2023. Accessed: 2023-10-25.

[45] Xin Fan, Chenlu Li, and Xiaoju Dong. A real-time network security visualization system based on incremental learning (chinavis 2018). *Journal of Visualization*, 22:215–229, 2019.

[46] Xiaohu Xu, Xixi Zhang, Qianyun Zhang, Yu Wang, Bamidele Adebisi, Tomoaki Ohtsuki, Hikmet Sari, and Guan Gui. Advancing malware detection in network traffic with self-paced class incremental learning. *IEEE Internet of Things Journal*, 11(12):21816–21826, 2024.

[47] Souad Ajjaj, Souad El Houssaini, Mustapha Hain, and Mohammed-Alamine El Houssaini. Incremental online machine learning for detecting malicious nodes in vehicular communications using real-time monitoring. In *Telecom*, volume 4, pages 629–648. MDPI, 2023.

[48] Daniel Krajzewicz. Traffic simulation with sumo–simulation of urban mobility. *Fundamentals of traffic simulation*, pages 269–293, 2010.

[49] Ruonan Wang, Jinlong Fei, Rongkai Zhang, Maohua Guo, Zan Qi, and Xue Li. Drnet: Dynamic retraining for malicious traffic small-sample incremental learning. *Electronics*, 12(12):2668, 2023.

[50] Ziming Zhao, Zhaoxuan Li, Zhuoxue Song, Wenhao Li, and Fan Zhang. Trident: A universal framework for fine-grained and class-incremental unknown traffic detection. In *Proceedings of the ACM on Web Conference 2024*, pages 1608–1619, 2024.

[51] Mahmoud Said Elsayed, Nhien-An Le-Khac, and Anca D. Jurcut. Insdn: A novel sdn intrusion dataset. *IEEE Access*, 8:165263–165284, 2020.

[52] Furqan Rustam, Anca Delia Jurcut, Wajdi Aljedaani, and Imran 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)*, pages 1–10. IEEE, 2023.

[53] Pubudu L Indrasiri, Ernesto Lee, Vaibhav Rupapara, Furqan Rustam, and Imran Ashraf. Malicious traffic detection in iot and local networks using stacked ensemble classifier. *Computers, Materials and Continua*, 71(1):489–515, 2022.

[54] Umer Zukaib, Xiaohui Cui, Chengliang Zheng, Dong Liang, and Salah Ud 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*, page 104934, 2024.

[55] Lionel F Gonzalez Casanova, Po-Chiang Lin, et al. Malicious network traffic detection for dns over https using machine learning algorithms. *APSIPA Transactions on Signal and Information Processing*, 12(2), 2023.

[56] Furqan Rustam, Rahman Shafique, Sarath Kumar Posa, and Anca Delia 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)*, pages 598–603. IEEE, 2024.

[57] Khalil El-Khatib. Impact of feature reduction on the efficiency of wireless intrusion detection systems. *IEEE TRANSACTIONS on parallel and distributed systems*, 21(8):1143–1149, 2009.

[58] Abhinav Mohanty, Islam Obaidat, Fadi Yilmaz, and Meera Sridhar. Control-hijacking vulnerabilities in IoT firmware: A brief survey. In *Proceedings of the 1st International Workshop on Security and Privacy for the Internet-of-Things (IoTSec)*, 2018.

[59] Islam Obaidat, Bennett Kahn, Fatemeh Tavakoli, and Meera Sridhar. Creating a large-scale memory error iot botnet using ns3dockeremulator. In *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 470–479. IEEE, 2023.

[60] Benoît Claise. Cisco Systems NetFlow Services Export Version 9. RFC 3954, October 2004.

[61] Ramin Fadaei Fouladi, Tina Seifpoor, and Emin Anarim. Frequency characteristics of dos and ddos attacks. In *2013 21st Signal Processing and Communications Applications Conference (SIU)*, pages 1–4. IEEE, 2013.

[62] Matt Mathis and John Heffner. Packetization Layer Path MTU Discovery. RFC 4821, March 2007.

[63] Syam Madanapalli, Gabriel Montenegro, Soohong Daniel Park, and Samita Chakrabarti. Transmission of IPv4 Packets over the IP Convergence Sublayer of IEEE 802.16. RFC 5948, August 2010.

[64] Igor V Kotenko, Igor Saenko, and Alexey Kushnerevich. Parallel big data processing system for security monitoring in internet of things networks. *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.*, 8(4):60–74, 2017.

[65] Ancy Sherin Jose, Latha R Nair, and Varghese Paul. Towards detecting flooding ddos attacks over software defined networks using machine learning techniques. *Revista Geintec-Gestao Inovacao E Tecnologias*, 11(4):3837–3865, 2021.

[66] Krista Rizman Žalik. An efficient k'-means clustering algorithm. *Pattern Recognition Letters*, 29(9):1385–1391, 2008.

[67] Mihir Gada, Zenil Haria, Arnav Mankad, Kaustubh Damania, and Smita Sankhe. Automated feature engineering and hyperparameter optimization for machine learning. In *2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS)*, volume 1, pages 981–986. IEEE, 2021.

[68] Shereen Ismail and Hassan Reza. Evaluation of naïve bayesian algorithms for cyber-attacks detection in wireless sensor networks. In *2022 IEEE world AI IoT congress (AIIoT)*, pages 283–289. IEEE, 2022.

[69] Angelos Angelopoulos, Anastasios E Giannopoulos, Nikolaos C Kapsalis, Sotirios T Spantideas, Lambros Sarakis, Stamatis Voliotis, and Panagiotis Trakadas. Impact of classifiers to drift detection method: a comparison. In *International conference on engineering applications of neural networks*, pages 399–410. Springer, 2021.

[70] Belal Sudqi Khater, Ainuddin Wahid Bin Abdul Wahab, Mohd Yamani Idna Bin Idris, Mohammed Abdulla Hussain, and Ashraf Ahmed Ibrahim. A lightweight perceptron-based intrusion detection system for fog computing. *applied sciences*, 9(1):178, 2019.

[71] Mrutyunjaya Panda, Ajith Abraham, and Manas Ranjan Patra. Discriminative multinomial naive bayes for network intrusion detection. In *2010 Sixth International Conference on Information Assurance and Security*, pages 5–10. IEEE, 2010.

[72] Alimov Abdulboriy and Ji Sun Shin. An incremental majority voting approach for intrusion detection system based on machine learning. *IEEE Access*, 2024.

[73] Akhil Jabbar Meerja, A Ashu, and Aluvalu 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*, pages 150–156. Springer, 2021.

[74] Mohamed Ali Setitra, Mingyu Fan, Bless Lord Y. Agbley, and Zine El Abidine Bensalem. Optimized mlp-cnn model to enhance detecting ddos attacks in sdn environment. *Network*, 3(4):538–562, 2023.

[75] Silpa Chalichalamala, Niranjana Govindan, and Ramani Kasarapu. Logistic regression ensemble classifier for intrusion detection system in internet of things. *Sensors*, 23(23):9583, 2023.

[76] Jahongir Azimjonov and Taehong Kim. Stochastic gradient descent classifier-based lightweight intrusion detection systems using the efficient feature subsets of datasets. *Expert Systems with Applications*, 237:121493, 2024.

[77] BS Sharmila and Rohini Nagapadma. Intrusion detection system using naive bayes algorithm. In *2019 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)*, pages 1–4. IEEE, 2019.

[78] Arianto Nugroho, Raditia Wahyuwidayat, Santi Tiodora Sianturi, Muhamad Fauzi, M Febrianto Ramadhan, Baskoro Adi Pratomo, Ary Mazharuddin Shiddiqi, et al. Ensemble methods classifier comparison for anomaly based intrusion detection system on cidds-002 dataset. In *2021 13th International Conference on Information & Communication Technology and System (ICTS)*, pages 62–67. IEEE, 2021.

[79] Md Al Mehedi Hasan, Mohammed Nasser, Shamim Ahmad, and Khademul Islam Molla. Feature selection for intrusion detection using random forest. *Journal of information security*, 7(3):129–140, 2016.

[80] Weiming Hu, Wei Hu, and Steve Maybank. Adaboost-based algorithm for network intrusion detection. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(2):577–583, 2008.

[81] Latifur Khan, Mamoun Awad, and Bhavani Thuraisingham. A new intrusion detection system using support vector machines and hierarchical clustering. *The VLDB journal*, 16:507–521, 2007.

[82] M Govindarajan and RM Chandrasekaran. Intrusion detection using k-nearest neighbor. In *2009 First International Conference on Advanced Computing*, pages 13–20. IEEE, 2009.

[83] Yasser A Ali, Emad Mahrous Awwad, Muna Al-Razgan, and Ali Maarouf. Hyperparameter search for machine learning algorithms for optimizing the computational complexity. *Processes*, 11(2):349, 2023.

[84] Benoit Courty, Victor Schmidt, Kamal Goyal, Marion Coutarel, Boris Feld, Jérémy Lecourt, Liam Connell, Amine Sab, Inimaz, Supatomic, Mathilde Léval, Luis Blanche, Alexis Cruveiller, Sara Oumina, Franklin Zhao, Aditya Joshi, Alexis Bogroff, Amine Saboni, Hugues de Lavoreille, Niko Laskaris, Edoardo Abati, Douglas Blank, Ziyao Wang, Armin Catovic, Alencon, Michał Stęchły, Christian Bauer, Lucas Otávio, JPW, and MinervaBooks. CodeCarbon (version 2.4.1). `https://doi.org/10.5281/zenodo.11171501`, May 2024. Zenodo. DOI: 10.5281/zenodo.11171501.

[85] Kenjiro Cho, Koushirou Mitsuya, and Akira Kato. Traffic data repository at the wide project. In *2000 USENIX Annual Technical Conference (USENIX ATC 00)*, 2000.

Appendix A. Statistical Feature Definitions and Calculation Details

Statistical Feature

1. *Packet Count*: The total number of packets observed in each time window, represented as $\text{PacketCount}_t = \sum_{i=1}^{n} \text{Packet}_i$.
2. *Destination Port Entropy*: Measures the entropy of destination ports to detect scanning activities, defined as $\text{Entropy} = -\sum_{i=1}^{n} p_i \log(p_i)$.
3. *Most Frequent Source Port*: Identifies the most common source port in a window, $\text{SourcePort}_{\max}$.
4. *Most Frequent Destination Port*: Identifies the most common destination port in a window, $\text{DestinationPort}_{\max}$.
5. *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.
6. *Repeated Connection Attempts*: Measures repeated connection attempts, $\text{RepeatedAttempts} = \sum_{i=1}^{n} \alpha_i$, where α_i indicates a repeated attempt.
7. *Network Scanning Activity*: Counts instances of SYN flags without ACK flags, $\text{Scanning} = \sum_{i=1}^{n} (\text{SYN} - \text{ACK})$.
8. *Flow Rate*: Calculated as packets per second, $\text{FlowRate} = \frac{\text{TotalPackets}}{\text{TimeInterval}}$.
9. *Source Entropy*: Entropy of source addresses, $\text{SourceEntropy} = -\sum_{i=1}^{n} q_i \log(q_i)$, where q_i is the probability distribution of source addresses.
10. *Connection Errors (RST flag)*: Counts instances of RST flags, $\text{RSTCount} = \sum_{i=1}^{n} \text{RST}_i$.
11. *Most Frequent Packet Size Frequency*: Identifies the most common packet size, $\text{PacketSize}_{\max}$.
12. *Abnormal Size Frequency*: Counts packets exceeding a size threshold, $\text{AbnormalSizeFrequency} = \sum_{i=1}^{n} \text{Packet}_i \text{ if } \text{size}_i > \text{Threshold}$.
13. *Sequence Number Variance*: Variance in sequence numbers, Var(SequenceNumber).
14. *Average Packet Number*: Average packets per interval, $\text{AvgPackets} = \frac{\text{TotalPackets}}{\text{TimeIntervals}}$.
15. *SYN Frequency*: Frequency of SYN flags, $\text{SYNFrequency} = \frac{\text{TotalSYN}}{\text{TimeInterval}}$.
16. *ACK Frequency*: Frequency of ACK flags, $\text{ACKFrequency} = \frac{\text{TotalACK}}{\text{TimeInterval}}$.
17. *TCP Frequency*: Proportion of TCP packets, $\text{TCPFrequency} = \frac{\text{TotalTCP}}{\text{TotalPackets}}$.
18. *UDP Frequency*: Proportion of UDP packets, $\text{UDPFrequency} = \frac{\text{TotalUDP}}{\text{TotalPackets}}$.
19. *Most Frequent Protocol*: Most used protocol, $\text{Protocol}_{\max}$.
20. *Packet Size Variability*: Variance in packet sizes, Var(PacketSize).
21. *Most Frequent Payload Size*: Most common payload size, $\text{PayloadSize}_{\max}$.
22. *Average Payload Size*: Mean payload size, $\text{AvgPayloadSize} = \frac{\text{TotalPayload}}{\text{TotalPackets}}$.
23. *Packet Size Standard Deviation*: Standard deviation of packet sizes, StdDev(PacketSize).
24. *Average Packet Size*: Mean size of packets within a window, $\text{AvgPacketSize} = \frac{\text{TotalPacketSize}}{\text{TotalPackets}}$.