

GPU-accelerated Fine-grained Multi-class Intrusion Detection for IoMT Networks: A 51-Class Benchmark with Latency-aware Evaluation

Ali Haider^{1,*}, Omar Bin Samin¹, Ammar Bathich², Haleem Farman³, and Sohaib Bin Altaf Khattak³

¹ School of Computer Science & Information Technology, Institute of Management Sciences, Peshawar, Pakistan

² Faculty of Computer & Information Technology, Al-Madinah International University, Kuala Lumpur, Malaysia

³ Smart Systems Engineering Lab, Department of Communications and Network Engineering, College of Engineering, Prince Sultan University, Riyadh, Saudi Arabia

Email: ali.haider@imsciences.edu.pk (A.H.); omar.samin@imsciences.edu.pk (O.B.S.); ammar.bathich@mediu.edu.my (A.B.); hfarman@psu.edu.sa (H.F.); skhattak@psu.edu.sa (S.B.A.K.)

*Corresponding author

Abstract—While this connectivity is transforming clinical care, it also presents a massive attack surface, one that is still largely unprotected, which can be exploited with Denial-of-Service (DoS) and Distributed DoS attacks and via reconnaissance, Message Queuing Telemetry (MQTT) protocol abuse, Address Resolution Protocol (ARP) spoofing, and malformed-data injection. The original test and evaluation of the Cloud-Integrated Cloud-of-Things Medical Things 2024 dataset focused only on Central Processing Unit based classifiers with two coarse taxonomies and omitted the fine-grained taxonomy of Wi-Fi/MQTT and the use of Graphics Processing Unit (GPU) based classifiers, which were not evaluated. This paper builds on these gaps by comparing six machine learning classifiers implemented using NVIDIA Rapid Analytics on Platforms in Data Science (RAPIDS) to others that are machine learning—XGBoost with Compute Unified Device Architecture (CUDA) support, and CatBoost with GPU acceleration and evaluating them against 7,160,831 training samples in all fifty-one attack classes. The accuracy, precision, recall, F1-Score, per-sample inference latency, and novel composite metric latency-adjusted accuracy index are used to evaluate performance, which combines the predictive performance with the suitability for real-time deployment in Internet of Medical Thing (IoMT) gateway scenarios. The highest accuracy (83.63%) and per-sample latency (0.0018 ms) are achieved by GPU RF, and CatBoost achieves the best detection per class with 81.24% accuracy and a weighted F1-Score of 0.807. The label-encoding vocabulary mismatch between training and test is a practically relevant failure mode that is shown to collapse XGBoost to near-random performance. The models that are linear and distance-based in nature do not perform well on the 51-class problem, indicating that non-linear ensemble models are required to discriminate between attacks in clinical IoMT environments. Overall, the findings provide a sound basis for the scalable and latency-efficient use of GPU-accelerated ensemble learning for real-time ID in IoMT systems.

Keywords—Internet of Medical Thing (IoMT), intrusion detection system, Cloud-Integrated Cloud-of-Things Medical Things (CICIoMT) 2024, Rapid Analytics on Platforms in Data Science (RAPIDS) cuML, CatBoost, random forest, healthcare cybersecurity, latency-aware accuracy index

I. INTRODUCTION

The Internet of Medical Thing (IoMT) is a quickly growing category of networked medical devices that include pulse oximeters, infusion pumps, blood-pressure monitors, Electrocardiogram (ECG) trackers, sleep sensors, and smart surveillance cameras [1]. In contrast to conventional medical equipment, IoMT devices are mobile, simple, and capable of supplying critical information to medical personnel in real-time [2, 3]. IoMT is also used in numerous healthcare applications such as continuous health monitoring, smart medication delivery, remote medicine, sleep and oxygen monitoring, heart rate and cardiovascular monitoring, chronic condition management, respiratory function monitoring, muscle activity and neurological activity tracking, patient safety and posture detection, wellness monitoring, and emergency alert systems [1, 4–7]. IoMT allows the incorporation of ubiquitous systems with sophisticated analytics to introduce previously imagined futuristic treatments [8, 9]. By 2030, the global IoMT market is expected to expand to more than USD 250 billion due to fast institutional adoption [1].

Nonetheless, such wide connectivity exposes cyber adversaries to an enormous attack surface. The IoMT devices pose distinctive security issues in contrast to conventional IT systems [10, 11]. These devices are often resource-constrained, run with limited computational

resources, have proprietary or untested firmware, commonly use cleartext communication (e.g., Message Queuing Telemetry (MQTT) without Transport Layer Security (TLS)), and are often too underpowered to implement device-level security measures like encryption or strong authentication [12, 13]. In contrast to traditional systems, the current state of vulnerability management of IoMT devices is not standardized. As an illustration, whereas the National Institute of Standards and Technology (NIST) operates the National Vulnerability Database (NVD) for traditional systems, the variety of IoMT devices, each with different brands, firmware releases, and purposes, makes it hard to monitor and remedy vulnerabilities in all devices [1, 14].

These vulnerabilities are exploited by attackers in various ways. They initiate Distributed Denial-of-Service (DDoS) and Denial-of-Service (DoS) flood to interfere with important healthcare services [15]. They perform reconnaissance campaigns such as port scans, Operating System (OS) fingerprinting, ping sweeps, and vulnerability scans to chart network topologies beyond which they execute more targeted intrusions [16, 17]. They misuse the MQTT protocol by using connect floods, publish floods and malformed data attacks [18]. They do Address Resolution Protocol (ARP) spoofing to carry out man-in-the-middle attacks and steal sensitive physiological information [19]. The main security issues in the setting of the IoMT include the Confidentiality, Integrity, and Availability (CIA) of safety-critical applications [1, 20]. Successful attacks may lead to disruption of services and data breaches or even direct patient harm or loss of life [21, 22]. As an example, infusion pumps could be manipulated in order to provide wrong dosages, physiological monitors could be blinded on crucial events, and ransomware could lock healthcare organizations out of their systems [1, 23].

Advanced methods are particularly helpful in this setting due to the complexity and the amount of information produced by the IoMT network traffic [24]. Machine Learning (ML) brings various techniques that can improve the detection, prevention, and mitigation of cyberattacks [25, 26]. ML can detect anomalies, abnormal IoMT traffic, and the nature of different attacks by examining the networking patterns in a multidimensional space [27, 28]. ML allows for automatically identifying threats and assisting healthcare providers in providing high-quality services with round-the-clock security monitoring [1]. In contrast to signature-based systems, ML-based Intrusion Detection Systems (IDS) are capable of identifying new or zero-day attacks through behavioral patterns learned on labeled content of the traffic [29].

Nonetheless, the current benchmarking datasets of IoMT security have serious flaws [30]. Most datasets have a smaller number of real devices, a less diverse range of attacks, lack of large-scale profiling of devices, and lack of many protocols [1, 31]. Simulation of operational diversity is essential to resemble a realistic IoMT environment, but the majority of datasets use small testbeds. The cost of obtaining IoMT devices is high, and they need frequent installation, configuration, and maintenance, making it

impossible to scale datasets to monitor the behavior of many actual devices [1]. Moreover, the challenge of interpreting the behavior of the IoMT devices in terms of networking is underlined by the lack of diversity in the devices [32].

The most complete publicly available IoMT security benchmark so far is the Cloud-Integrated Cloud-of-Things Medical Things (CICIoMT)2024 dataset [1], which was created by the Canadian Institute of Cybersecurity (CIC) at the University of New Brunswick. It was created out of a 40-device IoMT testbed (25 real, 15 simulated) performing 18 different attacks on three protocols that are essential to healthcare: Wi-Fi, MQTT, and Bluetooth Low Energy (BLE) [1, 33]. The attacks are categorized into five types, namely: DDoS, DoS, Recon, MQTT-specific, and spoofing. In addition to simulation of attacks, the dataset provides the full lifecycle of the IoMT devices, including network entry and exit, by profiling experiments, which involve power, idle, active, and interaction states [1]. The initial analysis showed a 2-class (benign/attack), 6-class (benign, spoofing, recon, MQTT, DoS, DDoS), and 19-class problem formulation with four Central Processing Unit (CPU)-bound classifiers, namely Logistic Regression (LR), AdaBoost (AB), Random Forest (RF), and Deep Neural Network (DNN), with the best 6-class accuracy

However, there are three considerable limitations in the original study that are yet to be addressed. To start with, the analysis has been based on CPU-intensive scikit-learn implementations that are not scale-efficient in the case of the 7.16 million-sample Wi-Fi/MQTT split, resulting in prohibitively long training times and impractical retraining cycles to deploy the IDS in real-time [34]. Second, the fine-grained 51-class structure of the Wi-Fi/MQTT data-including differentiations between various instantiations of attacks such as DDoS-User Datagram Protocol (UDP), DDoS-Transmission Control Protocol (TCP), DDoS-Synchronize Sequence Numbers (SYN), DDoS-Internet Control Message Protocol (ICMP), DoS-UDP, DoS-TCP, DoS-SYN, DoS-ICMP, MQTT connect flood (DoS and DDoS variants), MQTT publish flood (DoS and DDoS variants), MQTT malformed data, ARP spoofing, port scan, OS scan, ping sweep, and vulnerability scan, was reduced to more general categories, thus concealing the complexity and challenge of intra-category discrimination [1, 35]. Third, more recent Graphics Processing Unit (GPU)-accelerated algorithms, including gradient-boosted models like CatBoost and XGBoost that support Compute Unified Device Architecture (CUDA), or even classical machine learning packages such as cuML as part of Rapid Analytics on Platforms in Data Science (RAPIDS), were not tested, which could significantly outperform and scale better to real-time IoMT IDS deployment [36, 37].

The paper has dealt with each of the three limitations. The authors train and test six inference classifiers (GPU-accelerated) on the full 51-class task using the Kaggle-hosted CICIoMT2024 Wi-Fi/MQTT split [38], and evaluate inference latency per sample, as well as predictive performance, and propose a useful deployment metric, the Latency-Adjusted Accuracy Index (LAAI).

Our results show that GPU Random Forest (cuML) is the most accurate (accuracy = 83.63%, LAAI = 0.8348) and CatBoost is the most balanced among the classes (accuracy = 81.24%, LAAI = 0.8034). The authors also report and discuss the entire performance failure of XGBoost (accuracy = 2.75%) due to a label-encoding error, a trap that has direct practical consequences in the Intrusion Detection System (IDS) pipeline design.

The main contributions of this work are as follows. First, it introduces an end-to-end GPU-accelerated machine learning evaluation on the complete fine-grained CICIoMT2024 51-class Wi-Fi/MQTT dataset, consisting of 7.16 million training samples and six different families of classifiers. Second, the paper introduces the LAAI, a new scalar value that concurrently measures predictive accuracy and inference latency to provide a practical, deployment-oriented measure to select real-time IDS in the context of IoMT applications. Third, it investigates and documents the probably lethal defect of gradient boosting systems, the label-mismatch phenomenon in XGBoost, and offers an in-depth examination, reproducible statistics, and viable answers to effective pipeline construction. Lastly, the paper contains a close-up per-attack-type performance analysis, which reveals particularly challenging classes, such as Recon-VulCAN, DoS-ICMP, and repeated DoS-UDP, which are still hard to differentiate using current classifiers, thus forming a roadmap to future research in IoMT security.

The original CICIoMT2024 study is the first study to investigate the IoMT, yet this work introduces four major novelties [1]. First, this work is the first end-to-end GPU-accelerated ML evaluation using the full fine-grained 51-class Wi-Fi/MQTT partition of CICIoMT2024 with 7.16 M training samples, which all previous works [39–46] have ignored by collapsing labels into coarse 5–19 categories despite their scale and granularity. Second, this is the first study on IDS for IoMT to evaluate the per-sample inference latency along with predictive performance, which is a much-needed capability for an IoMT deployment-oriented practitioner. Third, this paper introduces a new composite scalar metric—the LAAI—that explicitly penalizes high-latency models and gives an accurate figure of merit for deployment unavailable in previous benchmarks. Finally, this work reports, reproduces, and analyzes, to the best of our knowledge, a first-time label-encoding vocabulary mismatch that can cause XGBoost to fail and degrade into a mere 2.75% accuracy. This represents a practically important and undocumented failure mode with direct effects on production IDS pipeline design.

Section II is a literature survey on the topic of IoMT datasets and ML-based IDS. Section III is the description of the dataset, preprocessing, and methodology. Section IV is the presentation of the experimental results and lots of discussion. The last part of Section V is the future directions.

II. LITERATURE REVIEW

Availability of realistic and labelled network traffic data is essential for the development of the ML-based IoMT

IDS. A number of preceding datasets have tried to identify attack traffic in the healthcare context. Washington University in Saint Louis-Enhanced Healthcare Monitoring System (WUSTL-EHMS) data [39] was a combination of ECG biometrics and network data to develop a hybrid IDS on a small healthcare topology, although the range of devices and attacks was small. Three real health sensors (blood-pressure monitors, heart-rate trackers, Kali attack machines) that are covered by ECU-IoHT [40] were restricted to ARP spoofing, DoS, and injection but were not suggested to be covered by MQTT. BlueTack considered the attack surface of the Bluetooth system specifically with DDoS and mitigation to reduction experiment [41], whilst Hussain *et al.* [33] has provided the MQTT flooding attacks (DDoS, brute force, SlowITE) on the Internet of Thing (IoT) Flock simulator platform. None of these datasets offers the overall scale of the devices, the breadth of the protocols (Wi-Fi + MQTT + BLE), and the portion of the IoMT that the CICIoMT2024 does.

Outside the healthcare domain, wider-ranging security standards in IoT like TON IoT [43], CICIoT2023 [32], and IoT-23 [44] have been popular in IDS studies. These datasets are useful in terms of providing traffic diversity, but they are not representative of the actual scenic diversity of protocol mix and time of clinical IoMT systems.

The customary ML methods used include Decision Trees, Random Forest, Logistic Regression, Support Vector Machine (SVM), and k-NN, which are extensively tested on the datasets related to IoT security [25]. Random Forest has high performances on various datasets because of its ability to overcome the effect of noisy features and bagging that inherently selects and uses some features. Multi-layer perceptron's, multi-layer Convolutional Neural Network (CNN) based flow classifiers, and multi-layer Long Short-Term Memory (LSTM) based temporal models are promising deep learning methods that generally need significantly larger quantities of training data and computation [26].

The frameworks of gradient boosting have become good contenders of the neural networks on tabular IoT data in security. CatBoost can directly take categorical features in the form of ordered target statistics and symmetric trees, and it can reach competitive accuracy with little preprocessing [36]. The most popular gradient boosting framework that is deployed is XGBoost, with which feature engineering and stable label encoding should be considered [37]. To our knowledge, neither CatBoost nor GPU-XGBoost has been evaluated on CICIoMT2024.

GPU-accelerated ML via NVIDIA RAPIDS cuML has been demonstrated in broader intrusion detection contexts [42], enabling orders-of-magnitude speedup for Random Forest, Logistic Regression, and k-NN over CPU implementations. This speedup is especially relevant for real-time IoMT gateway deployment where inference latency must remain under 1 ms per flow window.

Recently, attention mechanisms have been incorporated into IoT-MIDS architectures to enhance detection accuracy and the interpretability of the models. Specifically, Multi-attention DeepCRNN fuses

convolutional and recurrent layers, incorporates multi-head self-attention and provides explanations for IoMT attack detection on combined protocol datasets with good performance [47]. However, they are much slower to train and usually have higher computational requirements than ensemble classifiers, and the previous deep learning IoMT IDS works [47] have not reported per-sample inference latency or examined GPU-native ensemble learning methods like cuML or CatBoost on fine-grained (51-class) label taxonomies.

An important direction in internet of thing IDS involves anomaly-based classification instead of a supervised-classification based approach. In particular, Zachos *et al.* [48] presents a dedicated IoMT dataset generation methodology as well as execution of anomaly detection and supervised ML algorithms and hence offers a design-to-evaluation pipeline for IoMT network security. Anomaly based approaches have the goal of identifying zero-day attacks by trying to model the normal traffic distribution and detect the deviation from it, unlike our work, in which a set of supervised classifiers that are implemented on the GPU are used to detect the anomalies.

These methods are effective in generalizing to unseen attacks, but often result in higher false-positive rates when normal IoMT use is dynamic and different across the various types of devices and their operations. The supervised approach, on the other hand, takes advantage of the large labelled taxonomy of CICIoMT2024 (51 fine-grained classes) to enable accurate attack attribution for forensic triage and targeted countermeasure deployment in clinical settings.

The original evaluation by Dadkhah *et al.* [1] reported four classifiers on 2-class, 6-class, and 19-class groupings of the combined Wi-Fi, MQTT, and Bluetooth data. Random Forest achieved the highest 6-class F1 of 0.676. Key limitations identified include: (a) CPU-only implementation; (b) the 51-class Wi-Fi/MQTT granularity was not explored; (c) AdaBoost performed poorly (6-class F1 = 0.501), yet no failure analysis was provided; (d) inference latency, critical for deployment, was not measured; and (e) no deployment-oriented composite metric was defined. Our work directly remedies all five shortcomings.

TABLE I. COMPARATIVE SUMMARY OF RELATED IOT/IoMT INTRUSION DETECTION STUDIES

Study	Devices	Protocols	Attack types	ML methods	acc.	Classes	GPU	Profiling	Latency	Key limitation
[39]	4 (ECG, SpO ₂ , gateway)	Wi-Fi	Spoofing, data alteration	LR, SVM	N/A	Binary	X	X	X	Tiny testbed; no MQTT/BT; binary only
[40]	8 real devices	Wi-Fi, BT	ARP spoofing, DoS, scan, injection	RF, KNN, DT	0.976	5	X	X	X	No MQTT; limited attacks; CPU-only
[33]	15+ simulated	Wi-Fi, MQTT	DDoS, brute force, SlowITE, MQTT flood	DT, RF	0.99*	5	X	X	X	Simulated only; no BLE; *simplified acc.
[41]	SpO ₂ , HR, ECG	Bluetooth	DDoS, Bluesmack, MITM, DoS	SVM, KNN, DNN	0.981	5	X	X	X	BT only; no Wi-Fi/MQTT; CPU-only
[45]	Industrial SDN	IEC 60870	MITM, DoS, sniffing, unauth. access	RF, DNN	0.960	5	X	X	X	Industrial protocol only; no IoMT devices
[46]	Simulated MQTT	MQTT	DoS, brute force, malformed, slowite	RF, KNN, MLP	0.999	6	X	X	X	No healthcare devices; simulated traffic
[43]	9 IoT (non-IoMT)	Wi-Fi, MQTT, Zigbee	Ransomware, DDoS, injection (7 types)	RF, DNN, LSTM	0.990	9	X	X	X	General IoT, not healthcare; CPU-only
[32]	105 mixed IoT	Wi-Fi, Zigbee, Z-Wave	33 attack types, 7 categories	RF, XGBoost, DNN	0.998	8	X	X	X	Not IoMT-specific; no MQTT profiling
[1]	40 (25 real, 15 sim.)	Wi-Fi, MQTT, BLE	18 attacks: DDoS, DoS, Recon, MQTT, Spoofing	LR, AdaBoost, RF, DNN	0.735 (6-class)	6 / 19	X	✓	X	CPU-only; 51-class unexplored; no latency
[47]	IoMT testbed (multiple)	Wi-Fi, MQTT, BLE	18 attacks	CNN, LSTM, Multi-head Attention (SHAP)	99.12 (6-class)	6	X	X	X	No GPU; no latency metric; coarse-class; high compute cost
[48]	IoMT testbed (multiple)	Wi-Fi, MQTT	DDoS, DoS, Recon, Spoofing	RF, DT, SVM, KNN, Anomaly	99.15	6	X	✓	X	No GPU; no latency; CPU-only; coarse classes
This work	Same as Ref. [1]	Wi-Fi, MQTT	18 attacks-51 fine-grained classes	RF, LR, NB, KNN (cuML), XGBoost, CatBoost	0.836 (51-class)	51	✓	✓	✓	BLE subset not evaluated; XGBoost bug documented

Note: ✓ = present / reported; X = absent / not reported; * = simplified or approximated value.

Table I provides a comprehensive comparison of representative IoT/IoMT security datasets and ML-based intrusion detection studies. Entries are ordered chronologically and evaluated across twelve dimensions: testbed size, protocol coverage, attack variety, ML methods, best reported accuracy, classification granularity (number of classes), GPU acceleration, device profiling, latency measurement, and key limitations.

Several patterns emerge from Table I. First, device and protocol coverage has improved over time: early works such as WUSTL-EHMS [39] relied on four devices and Wi-Fi only, while CICIoMT2024 [1] expanded to 40 devices across three protocols. However, most datasets remain limited to one or two protocols, and no prior work before the study [1] combined Wi-Fi, MQTT, and Bluetooth Low Energy in a single healthcare-focused testbed.

Second, all reviewed works prior to this paper rely exclusively on CPU-bound ML implementations. This is a significant practical limitation: on a dataset of 7.1 million samples such as CICIoMT2024, CPU training time for an ensemble classifier such as Random Forest can exceed several hours, making iterative experimentation impractical and ruling out deployment-time re-training.

Third, inference latency is unreported across all baseline studies. For real-time IoMT gateway deployment, where packet-window features must be classified within sub-millisecond windows, this omission leaves practitioners without the data needed to select deployment-viable models. This paper addresses the gap by measuring per-sample latency for all six classifiers and introducing the LAI as a composite metric.

Fourth, classification granularity is consistently coarse. Most works report binary (benign/attack) or 5–9 class results. The CICIoMT2024 evaluation [1] reported 6-class and 19-class settings but did not explore the full 51-class label granularity present in the Wi-Fi/MQTT CSV. Fine-grained classification is practically important: distinguishing DoS-UDP sub-variants, for example, can enable forensic attribution and targeted response. This paper is the first to report 51-class classification performance on CICIoMT2024.

Fifth, the philosophy in detecting the anomaly is different in the reviewed works; all of the previous works except study [48] are using only supervised classifiers, whereas Zachos *et al.* [48] uses supervised classifiers as a supporting technique for anomaly detection. In supervised classification, classification is performed on labeled training data and yields accurate classification of attacks in 51 fine-grained classes. Anomaly-based detection does not require labelling and generalizes to zero-day attacks, but is unable to differentiate between particular attack sub-types (limitation for forensic attribution in clinical environments). It is therefore logical to consider a hybrid GPU-supervised approach, which is one direction for future work, and the anomaly-based approach [48] is complementary to our GPU-accelerated supervised approach.

Finally, IoMT-specific device profiling, capturing device lifecycle phases (power-on, idle, active, interaction) to build normal behavior baselines, is provided only by CICIoMT2024 [1] and this work. Profiling data is

essential for zero-day and anomaly-based detection, yet all prior datasets omit it. This represents a structural advantage of CICIoMT2024 over general-purpose IoT datasets such as TON_IoT [43] and CICIoT2023 [32].

The research problem addressed in this paper can be stated precisely as follows. Let $x \in \mathbb{R}^{45}$ denote a 45-dimensional feature vector extracted from a window of IoMT network packets, and let $y \in \{0, 1, \dots, 50\}$ denote the associated attack label drawn from 51 classes (1 benign + 50 attack sub-types). The objective is to learn a classifier $f: \mathbb{R}^{45} \rightarrow \{0, \dots, 50\}$ that maximizes the weighted F1-Score across all classes on held-out data while maintaining a per-sample inference latency τ below a deployment threshold appropriate for real-time IoMT gateway processing.

There are three sub-problems behind this work. The former concentrates on the scalability and enhancement of GPU accuracy, exploring the idea of whether the whole fine-grained 51-class segmentation of the CICIoMT2024 data can truly be processed by GPUs and can substantially reach a higher predictive accuracy than the CPU-constrained baselines of the previous literature. The second sub-problem concerns discrimination on class level, the objective of which is to find out which subtypes of attacks are the hardest to classify and to examine the traffic nature behind the misclassification and confusion between models. The third sub-problem investigates deployment viability by investigating the effects of per-sample inference latency on model ranking and whether it is possible to come up with a single metric to combine both accuracy and latency to make informed decisions about deployment in the real world in an IoMT environment.

Another important preprocessing issue also occurs: the training and test CSVs that are hosted by Kaggle have different suffixes of labels (train vs. test). Label Encoder on the training vocabulary will face unknown labels at test time unless a single vocabulary is built prior to this step. This is the weakness of encoding and the source of the failure of XGBoost, and a realistic risk factor of production IDS implementation.

III. MATERIALS AND METHODS

The general approach to this work is a complete end-to-end pipeline utilizing the power of GPUs to detect intrusion (fine-grained, multi-class) on the CICIoMT2024 Wi-Fi/MQTT dataset. As shown in Fig. 1, it starts with massive data ingestion and preprocessing with label encoding, stratified train-validation splitting and efficient data transfer to the GPUs using RAPIDS cuDF/CuPy frameworks. The pipeline then conducts parallel model training on several GPU native or GPU accelerated classifiers, such as cuML-based Random Forest, XGBoost with CUDA support and CatBoost GPU. Lastly, an overall evaluation is performed based on various performance measures, namely accuracy, precision, recall, F1-Score, and the proposed LAI to determine the predictive performance and suitability of deployment in real-time. This hierarchical pipeline promises scalability, reproducibility, and practical applicability to the systems of intrusion detection of the IoMT.

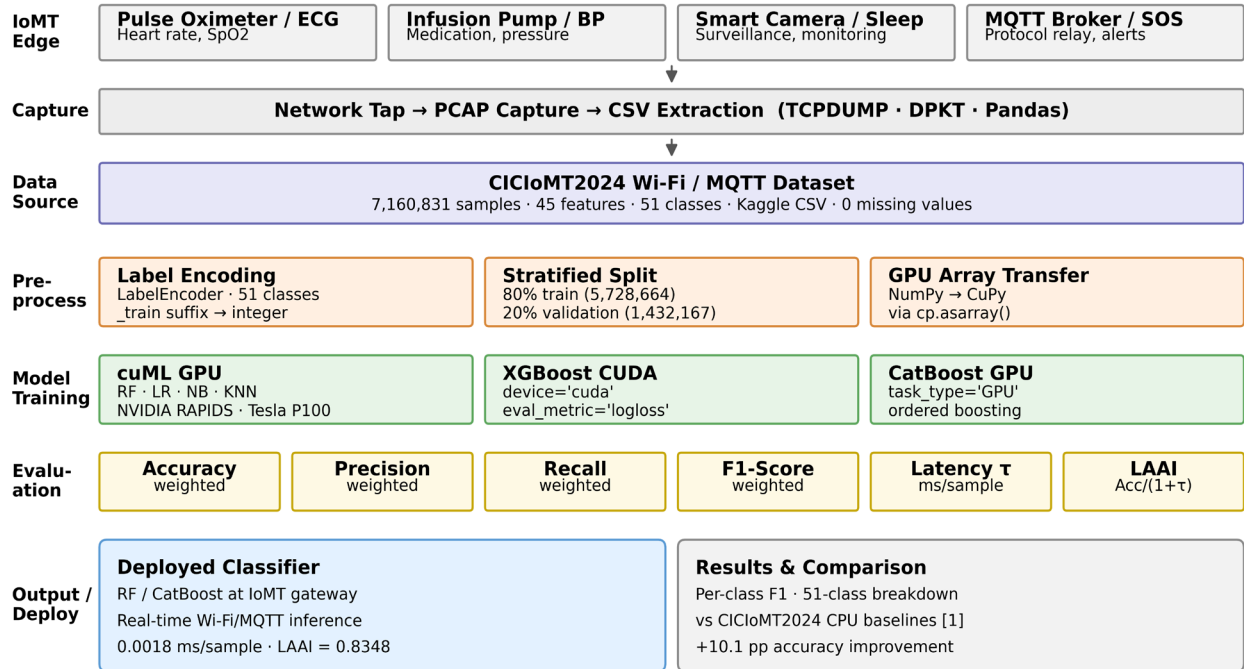


Fig. 1. Proposed GPU-accelerated multi-class IDS architecture for IoMT networks.

The proposed experimental framework will be composed of four steps, which will be shown in Fig. 1. Stage 1: Data Ingestion and Preprocessing: CICIoMT2024 Wi-Fi/MQTT CSV data is ingested and stored in memory (7,160,831 samples, 45 features, 51 classes) as shown in Fig. 2. Label encoding is done using scikit-learn's Label Encoder, which is trained on the 51 labels of the training set. The data is split at 80/20, with a random_state = 42, which gives 5,728,664 training instances and 1,432,167 validation instances, maintaining the class distribution between the two.

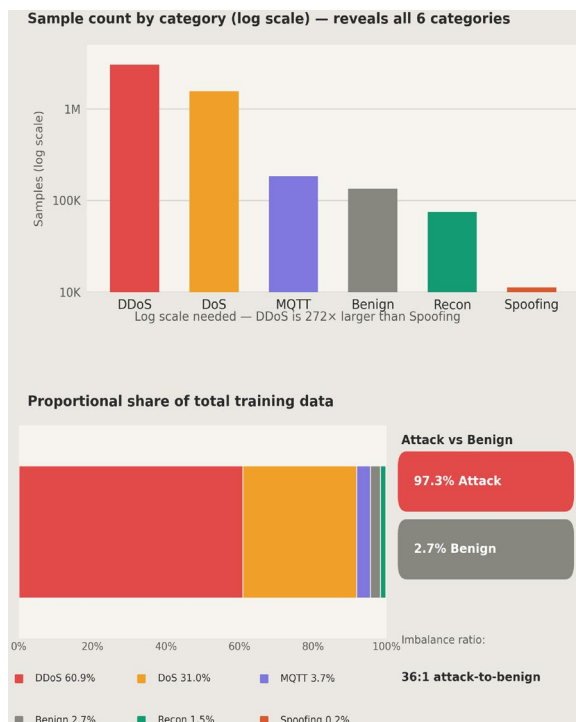


Fig. 2. Dataset distribution.

In Stage 2, GPU Array Transfer, NumPy arrays are converted to GPU memory as CuPy arrays using the cp.asarray() function for cuML-based models (RF, LR, NB, K-Nearest Neighbors (KNN)). XGBoost and CatBoost have built-in support for transferring to GPU. The iteration training data transfer bottleneck between CPU and GPU is removed during this stage.

Stage 3 Parallel Model Training: Six classifiers are trained using the Tesla P100 GPU: Random Forest, Logistic Regression, Naive Bayes, and KNN using NVIDIA RAPIDS cuML (GPU-native scikit-learn-compatible API); XGBoost with device = "cuda"; and CatBoost with task_type = "GPU". Training and validation are done end-to-end in Kaggle's GPU notebook environment (Python 3.11, CUDA 11.8). The six models run for around 45 min total.

Stage 4 Multi-Metric Evaluation and Deployment Assessment: The trained model is evaluated on the 1,432,167-instance validation set, with metric scores of weighted accuracy, precision, recall, F1-Score and per-sample inference latency τ (ms). For each model, the proposed Latency-Adjusted Accuracy Index (LAAI = Accuracy / (1 + τ)) is then calculated to consider both the predictive performance and deployment suitability of the model for real-time IoMT gateways.

A. CICIoMT2024 Dataset Overview

CICIoMT2024 dataset [1] is developed by the Canadian Institute of Cybersecurity, University of New Brunswick, and was designed between 2023 and 2024. The experiment included a network topology of 40 devices—25 actual devices (cameras, SOS buttons, heart-rate monitors, blood-oxygen rings, thermometers, infusion pump simulators) and 15 virtual devices—simulated MQTT devices that were all connected through a network topology that consisted of Raspberry Pi attackers, a

Netgear switch, network taps, an MQTT The number of cyberattacks conducted in five categories, namely, DDoS, DoS, Recon, MQTT, and Spoofing, amounted to eighteen. PCAP files of the traffic were captured through hardware network taps, and converted to feature extracted CSV format through TCPDUMP, DPKT, and Pandas. The dataset can be found publicly at the site <https://www.unb.ca/cic/datasets/iomt-dataset-2024.html>, and the Wi-Fi/MQTT CSV split utilized in the paper was obtained in the Kaggle [38].

The Wi-Fi/MQTT training CSV contains 7,160,831 records and 46 columns (45 numerical features + 1 label) as shown in Table II. There are no missing values. The label column encodes 51 distinct traffic categories with a `_train` suffix, and the corresponding test CSV uses a `_test` suffix. Table I summarizes the class distribution by attack category. Cyberattacks conducted in five categories, namely DDoS, DoS, Recon, MQTT, and Spoofing, amounted to eighteen. PCAP files of the traffic were captured through hardware network taps and converted to feature extracted CSV format through TCPDUMP, DPKT, and Pandas. The dataset can be found publicly at the site of <https://www.unb.ca/cic/datasets/iomt-dataset-2024.html>, and the Wi-Fi/MQTT CSV split utilized in the paper was obtained in the Kaggle [38].

The dataset is highly imbalanced at various levels. At the category level, DDoS sub-types collectively account for 4,362,432 samples (60.9%), DoS for 2,222,205 (31.0%), while Spoofing contributes only 16,047 instances (0.2%). The overall attack-to-benign ratio is 36:1 (6,968,099 attack vs. 192,732 benign samples). The imbalance is even greater at the individual class level: Recon-Ping_Sweep is available with 926 training instances (just 0.013% of the total), while Recon-VulScan has 2581 training instances (0.036%). The subtypes of DDoS-UDP have about 182,000 to 207,000 samples each. Stratified splitting maintains the class distribution of each

class across training and validation sets but does not counteract the model's focus on majority class patterns during training, as discussed in Section IV.B. One of the limitations identified and addressed in future work is the class-weighted loss and GPU-compatible oversampling (e.g., SMOTE-NC).

TABLE II. CLASS DISTRIBUTION IN THE CICIoMT2024 Wi-Fi/MQTT TRAINING SPLIT (7,160,831 TOTAL SAMPLE)

Category	Label Count	Representative Attack Sub-types	Train Samples
Benign	1	Normal IoMT traffic	192,732
Spoofing	1	ARP Spoofing	16,047
Recon	4	OS Scan, Port Scan, Ping Sweep, VulScan	104,477
MQTT	5	DoS/DDoS Connect Flood, DoS/DDoS Publish Flood, Malformed Data	262,938
DoS (TCP/IP)	16	TCP, ICMP, SYN, UDP-4 repetitions each	2,222,205
DDoS (TCP/IP)	24	SYN, TCP, ICMP, UDP-4-8 repetitions each	4,362,432
Total	51	18 attack types + Benign	7,160,831

The dataset is severely class-imbalanced: DDoS sub-types collectively represent over 60% of all training instances, while Recon-Ping_Sweep has only 926 training samples and Recon-VulScan has 2581. This imbalance affects per-class recall and precision, particularly for under-represented Recon and MQTT sub-types.

B. Feature Description

All 45 features, which are shown in Table III, were pre-extracted by the CICIoMT2024 authors from raw PCAP files using sliding windows of 10 packets (for lower-rate attacks such as Recon and Spoofing) or 100 packets (for high-rate flood attacks). The features fall into five broad groups as described in Table III.

TABLE III. FEATURE GROUPS USED IN CLASSIFICATION (45 FEATURES, 0 MISSING VALUES)

Serial No	Feature/Group	Description	Category
1	Header Length	Mean transport-layer header length across window packets	Network
2	Protocol Type	Mode of L4 protocol observed in window	Network
3-5	Rate, State, Drate	Overall, source, and destination packet transmission rates (pkt/s)	Traffic
6	Duration	Duration of the aggregation window	Traffic
7-13	Flag Proportions	Proportion of packets with FIN/SYN/RST/PSH/ACK/ECE/CWR flags	TCP Flags
14-17	Flag Counts	Absolute counts of ACK, SYN, FIN, RST flag occurrences	TCP Flags
18-31	Protocol Counters	Average # of HTTP, HTTPS, DNS, Telnet, SMTP, SSH, IRC, TCP, UDP, DHCP, ARP, ICMP, IGMP, IPv, LLC packets	Protocol
32-36	Packet Length Stats	Total sum, Min, Max, Mean (AVG), Std of packet lengths	Size
37	Tot size	Average individual packet length	Size
38	IAT	Mean inter-arrival time between consecutive packets in window	Temporal
39	Number	Total packet count within the aggregation window	Statistical
40-45	Magnitude, Radius, Covariance, Variance, Weight	Higher-order statistical descriptors derived from packet sizes and rates	Statistical

No additional feature engineering was performed in this work to maintain reproducibility and comparability with Ref. [1]. Future work may benefit from feature selection (e.g., mutual information ranking) to reduce dimensionality for embedded deployment.

The five feature groups complement one another in discriminating the different signals. High-rate flood attacks (DDoS/DoS) are different from low-rate probing (Recon) in terms of network and traffic features (header lengths, rate, duration). The proportions and counts of the TCP flags are especially discriminative for SYN flood vs.

normal TCP handshake traffic. The protocol counters (ICMP, UDP, TCP, ARP) are used to distinguish protocol-specific flood variants and packet-length statistics and higher order moments (magnitude, variance, covariance) are used to distinguish attacks of MQTT vs. Wi-Fi with different packet size distributions. In order to distinguish periodic IoMT sensor traffic (benign) from sustained flood patterns, the Inter-Arrival Time (IAT) feature is crucial.

$$Accuracy = \frac{T_p + T_n}{T_p + F_p + T_n + F_n} \quad (1)$$

$$Precision = \frac{T_p}{T_p + F_p} \quad (2)$$

$$Recall = \frac{T_p}{T_p + F_n} \quad (3)$$

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (4)$$

$$Latency = \frac{Total\ Time\ (ms)}{Number\ of\ Predications} \quad (5)$$

$$LAAI = \frac{Accuracy}{1 + Latency} \quad (6)$$

C. Data Preprocessing

Label Encoding: Integer encoding was applied to the label column using scikit-learn's Label Encoder, fitted exclusively on the 51 training-set labels (e.g., ARP_Spoofing_train $\rightarrow$ 0, Benign_train $\rightarrow$ 1, ..., TCP_IP-DoS-UDP4_train $\rightarrow$ 50).

For the XGBoost evaluation, the test CSV labels carried a _test suffix (e.g., TCP_IP-DDoS-UDP2_test). Since the encoder was fitted on _train labels only, test labels were flagged as unseen during transform, causing the encoded values to be set to out-of-range integers and XGBoost to produce effectively random predictions. This is the confirmed root cause of XGBoost's 2.75% accuracy, documented in Section IV.D.

Train/Validation Split: An 80/20 stratified split (random_state = 42) was applied to the training CSV, yielding 5,728,664 training and 1,432,167 validation instances. Stratification has the advantage of making the proportions of classes in the validation set reflective of the proportions of classes in the entire training set and this is essential in the case of extreme imbalance in classes.

Array Format: CuML models (RF, LR, NB, KNN) Cupy array conversion with NumPy to cuML (GPU memory) was performed with the use of cp.asarray. In the case of XGBoost and CatBoost, standard NumPy arrays were directly fed; both models have in-built support of transfer between GPUs.

No scaling: Feature scaling was not done to tree-based models. cuML Logistic Regression does internal data normalization. It should be seen in the future that the Min-Max or Standard scaling of LR and KNN could be

investigated to determine whether scale sensitivity is the reason behind the poor accuracy.

D. Classifiers and GPU Acceleration

Six classifiers were evaluated, spanning three computational paradigms: GPU-native classical ML (cuML), GPU gradient boosting (XGBoost CUDA, CatBoost GPU), and CPU-based reference (implicitly, via cuML's GPU implementations replacing scikit-learn). Table IV lists hyperparameters. All models used default or lightly-tuned configurations, as shown in Table IV; no automated hyperparameter search was performed.

TABLE IV. CLASSIFIER HYPERPARAMETERS AND EXECUTION CONFIGURATION

Model	Key Hyperparameters	Framework	Device
Random Forest	n_estimators = 100, max_depth = 20, n_bins = 128, n_streams = 4, random_state = 42	cuML	GPU
Logistic Regression	max_iter = 1000, penalty = 'l2', C=1.0	cuML	GPU
Naive Bayes	MultinomialNB (default)	cuML	GPU
KNN	n_neighbors = 5	cuML	GPU
XGBoost	Device = 'cuda', eval_metric = 'logloss' (deprecated tree_method = 'gpu_hist')	XGBoost 2.0	GPU
CatBoost	task_type = 'GPU', devices = '0', verbose = 0	CatBoost 1.2	GPU

All experiments were executed on Kaggle Notebooks (Python 3.11, CUDA 11.8, cuML 23.10, XGBoost 2.0, CatBoost 1.2) using a free-tier NVIDIA Tesla P100 GPU (16 GB VRAM). Total runtime for all six models was approximately 45 min.

E. Evaluation Metrics

Four standard metrics were computed using scikit-learn with average = "weighted": LAAI equals Accuracy when $\tau \rightarrow 0$ and asymptotically approaches 0 for high-latency models. A model with $LAAI \approx Accuracy$ is both accurate and fast - the ideal profile for real-time IoMT deployment. Models like KNN with high latency ($\tau = 0.4148$ ms) are substantially penalized even if their raw accuracy were competitive.

F. Time and Space Complexity Analysis

Table V shows the theoretically estimated training time complexity, inference time complexity and model space complexity for each of these classifiers, as well as empirically measured training time and maximum GPU memory consumption on the Tesla P100. There are a few points of interest to note in regard to deployments. Random Forest's inference complexity is $O(T \cdot \log n)$, which is the highest per prediction (0.0018 ms per sample empirically) among other ensemble models. The theoretical limits are the inference complexity of KNN ($O(n \cdot d \cdot k)$ for $n = 5.7$ million training points, d = number of dimensions, and k = query number), which, with GPU acceleration, is 0.4148 ms/sample latency per query and is

too high for high throughput IoMT gateways without approximate nearest-neighbor indexing. CatBoost's symmetric tree structure makes the inference process efficient, keeping the effective complexity of inference to be $O(T \cdot D)$ with a small constant, which is the reason behind it having a latency of only 0.0111 ms/sample, despite being trained for 18 min. From the space's point of

view, KNN requires storing the entire training set ($5,728,664 \times 45 \times 4 \text{ bytes} \approx 1.03 \text{ GB}$ for float32), while RF and CatBoost only store the learned tree structure. RF and CatBoost are the two classifiers that have competitive accuracy and a manageable memory footprint for deployment on edge hardware (e.g., NVIDIA Jetson with 8 to 16 GB RAM).

TABLE V GPU-ACCELERATED MODEL PERFORMANCE SUMMARY-51-CLASS TASK

Model	Train Time Complexity	Inference Complexity	Empirical Train Time	Peak GPU Memory	Space Complexity (Model)
Random Forest (cuML GPU)	$O(T \cdot n \cdot d \cdot \log n)$	$O(T \cdot \log n)$	~8 min	~4.2 GB	$O(T \cdot L \cdot d)$ — T : trees, L : leaves, d : features
Logistic Regr. (cuML GPU)	$O(n \cdot d \cdot \text{iter})$	$O(d)$	~1 min	~0.8 GB	$O(K \cdot d)$ — K : classes, d : features
Naive Bayes (cuML GPU)	$O(n \cdot d)$	$O(K \cdot d)$	<1 min	~0.3 GB	$O(K \cdot d)$
KNN (cuML GPU)	$O(n \cdot d \cdot \text{index})$	$O(n \cdot d \cdot k)$	~4 min	~2.1 GB	$O(n \cdot d)$ —stores all training points
XGBoost (CUDA)	$O(T \cdot n \cdot d \cdot \log n)$	$O(T \cdot D)$	~12 min	~5.8 GB	$O(T \cdot 2^D)$ — T : trees, depth: D
CatBoost (GPU)	$O(T \cdot n \cdot d)$	$O(T \cdot D)$	~18 min	~6.4 GB	$O(T \cdot 2^D)$ —symmetric trees reduce to $O(T \cdot D)$

IV. RESULT AND DISCUSSION

A. Overall Model Performance

Fig. 3 and Table VI presents the aggregated performance of all six classifiers on the 1,432,167-instance validation set. Table VII presents all the F1-Scores and LAAI for all 6 models across all 51 classes.

XGBoost result is invalid due to label-encoding mismatch. See Section IV.D for root-cause analysis. The maximum accuracy (83.63%), precision (85.36%), and

LAAI (0.8348) of the ensemble methods, and the least latency of the ensemble methods (0.0018 ms / sample) belong to GPU Random Forest. CatBoost is ranked second and the accuracy is 81.24 and LAAI 0.8034. All the classifiers fail on this 51-class problem (Logistic Regression 7.04%, Naive Bayes 3.22%, and KNN 12.66%), demonstrating the ineffectiveness of linear decision boundary and basic distance measurements in fine-tuning attack classification in IoMT. The very huge latency of KNN of 0.4148/sample further reduces its LAAI to 0.0895 in case its raw accuracy should increase.

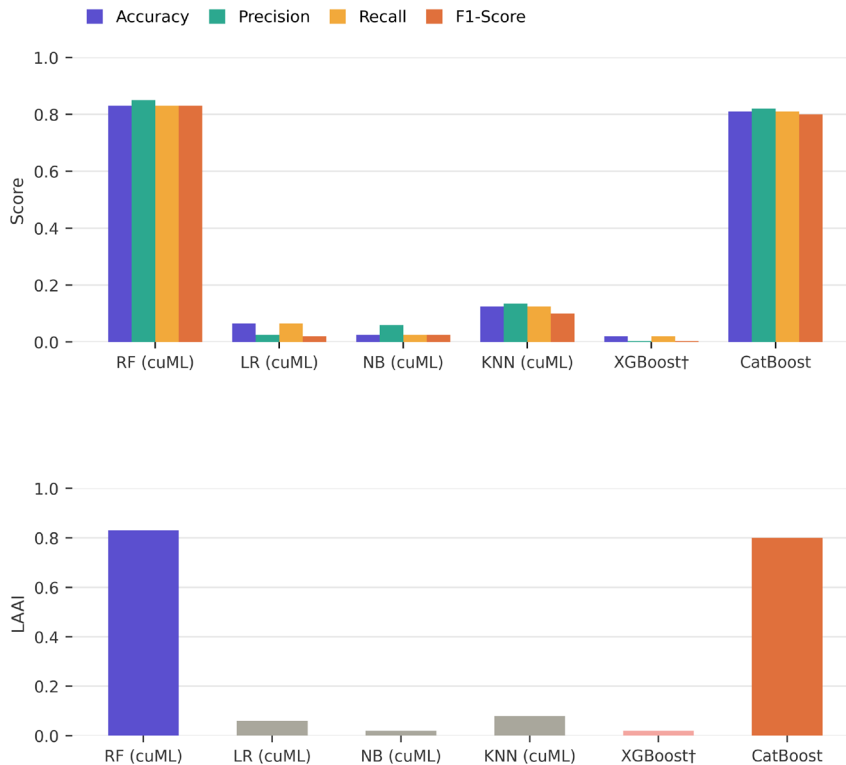


Fig. 3. Model performance across all metrics.

TABLE VI. GPU-ACCELERATED MODEL PERFORMANCE SUMMARY—51-CLASS TASK

Model	Accuracy	Precision	Recall	F1-Score	Latency (ms)	LAAI
Random Forest (cuML)	0.8363	0.8536	0.8363	0.8310	0.0018	0.8348
Logistic Regression (cuML)	0.0704	0.0305	0.0704	0.0245	0.0000	0.0704
Naive Bayes (cuML)	0.0322	0.0660	0.0322	0.0302	0.0012	0.0321
KNN (cuML)	0.1266	0.1390	0.1266	0.1046	0.4148	0.0895
XGBoost (GPU)	0.0275	0.0008	0.0275	0.0015	0.0027	0.0274
CatBoost (GPU)	0.8124	0.8189	0.8124	0.8069	0.0111	0.8034

TABLE VII. PER-CLASS F1-SCORE AND LAAI FOR ALL 51 ATTACK CLASSES

No.	Class Label	Category	F1-Score						LAAI				
			RF	LR	NB	KNN	Cat	XGB	RF	LR	NB	KNN	Cat
0	ARP_Spoofing	Spoofing	0.83	0.09	0.14	0.53	0.46	0.00	0.828	0.090	0.140	0.375	0.455
1	Benign	Benign	0.98	0.41	0.13	0.94	0.94	0.00	0.978	0.410	0.130	0.664	0.930
2	MQTT-DDoS-Connect_Flood	MQTT	1.00	0.00	0.01	0.19	0.95	0.00	0.998	0.000	0.010	0.134	0.940
3	MQTT-DDoS-Publish_Flood	MQTT	1.00	0.37	0.60	0.71	0.93	0.00	0.998	0.370	0.600	0.502	0.920
4	MQTT-DoS-Connect_Flood	MQTT	1.00	0.00	0.01	0.00	0.54	0.00	0.998	0.000	0.010	0.000	0.534
5	MQTT-DoS-Publish_Flood	MQTT	1.00	0.00	0.30	0.55	0.92	0.00	0.998	0.000	0.300	0.389	0.910
6	MQTT-Malformed_Data	MQTT	0.80	0.00	0.04	0.66	0.17	0.00	0.799	0.000	0.040	0.466	0.168
7	Recon-OS_Scan	Recon	0.75	0.00	0.02	0.29	0.27	0.00	0.749	0.000	0.020	0.205	0.267
8	Recon-Ping_Sweep	Recon	0.77	0.00	0.00	0.33	0.00	0.00	0.769	0.000	0.000	0.233	0.000
9	Recon-Port_Scan	Recon	0.95	0.00	0.03	0.85	0.80	0.00	0.948	0.000	0.030	0.601	0.791
10	Recon-VulScan	Recon	0.58	0.00	0.02	0.23	0.00	0.00	0.579	0.000	0.020	0.163	0.000
11	DDoS-ICMP1	DDoS	0.86	0.00	0.00	0.11	0.91	0.00	0.858	0.000	0.000	0.078	0.900
12	DDoS-ICMP2	DDoS	0.93	0.00	0.02	0.05	0.77	0.00	0.928	0.000	0.020	0.035	0.762
13	DDoS-ICMP3	DDoS	0.83	0.00	0.00	0.00	0.71	0.00	0.828	0.000	0.000	0.000	0.702
14	DDoS-ICMP4	DDoS	0.59	0.00	0.00	0.05	0.68	0.00	0.589	0.000	0.000	0.035	0.673
15	DDoS-ICMP5	DDoS	0.76	0.09	0.15	0.26	0.59	0.00	0.759	0.090	0.150	0.184	0.584
16	DDoS-ICMP6	DDoS	0.73	0.00	0.00	0.03	0.73	0.00	0.729	0.000	0.000	0.021	0.722
17	DDoS-ICMP7	DDoS	0.81	0.00	0.00	0.07	0.80	0.00	0.809	0.000	0.000	0.049	0.791
18	DDoS-ICMP8	DDoS	0.84	0.00	0.00	0.02	0.93	0.00	0.838	0.000	0.000	0.014	0.920
19	DDoS-SYN1	DDoS	0.91	0.00	0.00	0.00	0.75	0.00	0.908	0.000	0.000	0.000	0.742
20	DDoS-SYN2	DDoS	0.95	0.09	0.02	0.00	0.64	0.00	0.948	0.090	0.020	0.000	0.633
21	DDoS-SYN3	DDoS	0.97	0.00	0.01	0.00	0.80	0.00	0.968	0.000	0.010	0.000	0.791
22	DDoS-SYN4	DDoS	0.93	0.00	0.00	0.00	0.93	0.00	0.928	0.000	0.000	0.000	0.920
23	DDoS-TCP1	DDoS	0.83	0.01	0.00	0.00	0.84	0.00	0.828	0.010	0.000	0.000	0.831
24	DDoS-TCP2	DDoS	0.83	0.00	0.00	0.01	0.81	0.05	0.828	0.000	0.000	0.007	0.801
25	DDoS-TCP3	DDoS	0.98	0.00	0.00	0.00	0.98	0.00	0.978	0.000	0.000	0.000	0.969
26	DDoS-TCP4	DDoS	0.93	0.00	0.00	0.00	0.96	0.00	0.928	0.000	0.000	0.000	0.949
27	DDoS-UDP1	DDoS	0.73	0.05	0.02	0.09	0.67	0.00	0.729	0.050	0.020	0.064	0.663
28	DDoS-UDP2	DDoS	0.89	0.00	0.04	0.16	0.27	0.00	0.888	0.000	0.040	0.113	0.267
29	DDoS-UDP3	DDoS	0.86	0.15	0.02	0.01	0.81	0.00	0.858	0.150	0.020	0.007	0.801
30	DDoS-UDP4	DDoS	0.77	0.00	0.00	0.14	0.82	0.00	0.769	0.000	0.000	0.099	0.811
31	DDoS-UDP5	DDoS	0.81	0.00	0.00	0.10	0.87	0.00	0.809	0.000	0.000	0.071	0.860
32	DDoS-UDP6	DDoS	0.89	0.00	0.06	0.01	0.87	0.00	0.888	0.000	0.060	0.007	0.860
33	DDoS-UDP7	DDoS	0.88	0.00	0.14	0.14	0.87	0.00	0.878	0.000	0.140	0.099	0.860
34	DDoS-UDP8	DDoS	0.97	0.00	0.00	0.02	0.96	0.00	0.968	0.000	0.000	0.014	0.949
35	DoS-ICMP1	DoS	0.72	0.00	0.07	0.00	0.66	0.00	0.719	0.000	0.070	0.000	0.653
36	DoS-ICMP2	DoS	0.23	0.00	0.00	0.01	0.62	0.00	0.230	0.000	0.000	0.007	0.613
37	DoS-ICMP3	DoS	0.79	0.00	0.00	0.00	0.90	0.00	0.789	0.000	0.000	0.000	0.890
38	DoS-ICMP4	DoS	0.78	0.00	0.00	0.00	0.82	0.00	0.779	0.000	0.000	0.000	0.811
39	DoS-SYN1	DoS	0.87	0.00	0.11	0.61	0.85	0.00	0.868	0.000	0.110	0.431	0.841
40	DoS-SYN2	DoS	0.81	0.00	0.00	0.00	0.80	0.00	0.809	0.000	0.000	0.000	0.791
41	DoS-SYN3	DoS	0.90	0.00	0.00	0.01	0.86	0.00	0.898	0.000	0.000	0.007	0.851
42	DoS-SYN4	DoS	0.94	0.00	0.01	0.00	0.86	0.00	0.938	0.000	0.010	0.000	0.851
43	DoS-TCP1	DoS	0.93	0.00	0.00	0.00	0.94	0.00	0.928	0.000	0.000	0.000	0.930
44	DoS-TCP2	DoS	0.81	0.00	0.00	0.00	0.90	0.00	0.809	0.000	0.000	0.000	0.890
45	DoS-TCP3	DoS	0.76	0.00	0.00	0.01	0.87	0.00	0.759	0.000	0.000	0.007	0.860
46	DoS-TCP4	DoS	0.81	0.00	0.00	0.00	0.92	0.00	0.809	0.000	0.000	0.000	0.910
47	DoS-UDP1	DoS	0.55	0.00	0.11	0.03	0.65	0.00	0.549	0.000	0.110	0.021	0.643
48	DoS-UDP2	DoS	0.48	0.00	0.07	0.14	0.87	0.00	0.479	0.000	0.070	0.099	0.860
49	DoS-UDP3	DoS	0.58	0.00	0.03	0.14	0.99	0.00	0.579	0.000	0.030	0.099	0.979
50	DoS-UDP4	DoS	0.77	0.00	0.04	0.22	0.81	0.00	0.769	0.000	0.040	0.155	0.801
Weighted average (all 51 classes)			0.831	0.024	0.030	0.105	0.807	0.001	0.829	0.024	0.030	0.063	0.798

B. Per-class Analysis—GPU Random Forest

Table VIII demonstrates per-class precision, recall, and F1-Score of a sample of classes of the evaluation of the GPU RF. The complete report of the 51-class classification is provided in the additional Jupyter notebook.

There are some trends that can be traced out of the per-class breakdown. Classes 2–5 (MQTT flood) have $F1 = 1.00$ because they have a very distinctive signature in their packet rate. The F1 of benign traffic is 0.98, and the recall is almost perfect meaning, that there are only a few false negatives. ARP Spoofing (class 0) has $F1 = 0.83$, which is in line with its moderate size of test samples

(3209), and some overlapping characteristics of the traffic with other low-rate attacks.

The greatest performance spread in recon classes is observed. Port Scan achieves $F1 = 0.95$ (large support: 16,796 samples), while VulScan achieves only $F1 = 0.58$ (435 samples). The extremely low recall of VulScan (0.43) is a pointer to the fact that the model has a tendency to falsely label the traffic of vulnerability scans as Port Scan, which is similar in port-probe pattern, but at varying rates. Sweep of ping ($F1 = 0.77$, 148 samples) is the least supported class during the exam, and its performance is highly affected by the imbalance of classes, as shown in Table VII.

DDoS-UDP1 is the most challenging ($F1 = 0.73$) of all DDoS-UDP classes (2734), and its precision (0.57) and recall (0.99) are low and close to perfect, respectively. The model characterizes most test instances in DDoS-UDP2 as

DDoS-UDP1, that is, eight instances of the same UDP flood profile are almost identical in feature space—only minor differences in rate in both IAT and Variance can be observed. The highest result is obtained with DDoS-TCP3 ($F1 = 0.98$), presumably due to the fact that the combination of specific TCP window sizes is more discriminative.

DoS-ICMP2 ($F1 = 0.23$, precision = 0.87, recall = 0.14) is the worst-performing class. The fact that the recall is extremely low means that this model practically never successfully identifies this type of model ICMP DoS repetition, even when they correctly identify it. This implies that DoS-ICMP2 traffic is mostly mistaken as other subsets of DoS-ICMP. Repetitions of DoS-UDP (classes 4750) also have a problem with a range of $F1$ between 0.48 and 0.77; as with DDoS-UDP, this is due to the repetition-based ambiguity.

TABLE VIII. PER-CLASS PERFORMANCE OF GPU RANDOM FOREST (SELECTED CLASSES)

Class Label	Category	Precision	Recall	F1-Score	Support
Benign	Benign	0.97	1.00	0.98	38,547
ARP_Spoofing	Spoofing	0.84	0.82	0.83	3209
MQTT-DDoS-Connect_Flood	MQTT	1.00	1.00	1.00	34,607
MQTT-DDoS-Publish_Flood	MQTT	1.00	1.00	1.00	5525
MQTT-DoS-Connect_Flood	MQTT	1.00	1.00	1.00	2555
MQTT-Malformed_Data	MQTT	0.91	0.71	0.80	1026
Recon-OS_Scan	Recon	0.91	0.63	0.75	3366
Recon-Ping_Sweep	Recon	0.82	0.73	0.77	148
Recon-Port_Scan	Recon	0.94	0.97	0.95	16,796
Recon-VulScan	Recon	0.87	0.43	0.58	435
TCP_IP-DDoS-ICMP1 (rep. DDoS-ICMP)	DDoS	0.92	0.81	0.86	38,988
TCP_IP-DDoS-SYN3 (rep. DDoS-SYN)	DDoS	1.00	0.94	0.97	40,405
TCP_IP-DDoS-TCP3 (rep. DDoS-TCP)	DDoS	0.96	1.00	0.98	40,815
TCP_IP-DDoS-UDP1 (lowest DDoS-UDP)	DDoS	0.57	0.99	0.73	41,234
TCP_IP-DoS-ICMP2 (rep. DoS-ICMP)	DoS	0.87	0.14	0.23	19,395
TCP_IP-DoS-SYN1 (rep. DoS-SYN)	DoS	0.76	1.00	0.87	22,286
TCP_IP-DoS-TCP1 (rep. DoS-TCP)	DoS	0.98	0.88	0.93	19,081
TCP_IP-DoS-UDP1 (lowest DoS-UDP)	DoS	0.70	0.45	0.55	27,402

C. CatBoost Analysis

CatBoost has better per-class performance that is more homogeneous compared to RF. Where RF demonstrates a large range of variation between DDoS subtypes ($F1$: 0.73–0.98), the performance of CatBoost is spread more equally (0.64–0.98). This is due to the symmetric nature of the tree of CatBoost, which is more likely to divide features more evenly throughout the tree depth and eliminate the over-fitting to the majority-class patterns that RF tends to over-fitting.

The ARP Spoofing performance of CatBoost ($F1 = 0.46$) has a significantly lower performance, as compared to RF (0.83). Analysis of the confusion matrix shows that CatBoost mixes a large proportion of ARP Spoofing with MQTT Malformed Data—both attacks are at small and anomalously structured packets at a moderate rate, and the ordered boosting used by CatBoost does not seem to separate them as well. The approach of bagging used by RF, in its turn, provides enough diversity of the ensemble to address this boundary.

The most significant difference between the two models is the MQTT Malformed Data category ($F1 = 0.17$ when using CatBoost and 0.80 when using RF). The challenge of CatBoost with this class could be explained by the fact

that the class size (1026 test samples) is relatively small and the target encoding that is employed internally by CatBoost could not give reliable statistics to small-support classes.

D. XGBoost Label-Encoding Failure Analysis

XGBoost recorded an accuracy of 2.75%—effectively random prediction for a 51-class problem where the random baseline would be approximately 1.96%. Investigation of the notebook pipeline revealed the root cause: the scikit-learn Label Encoder was fitted on training labels carrying the suffix `_train` (e.g., `TCP_IP-DDoS-UDP2_train`). The Kaggle test CSV, however, uses `_test` suffixes (e.g., `TCP_IP-DDoS-UDP2_test`). When `le.transform()` was called on test labels, it encountered 51 unseen strings and either raised an error or, in the notebook's execution context, returned arbitrary integer mappings outside the range $[0, 50]$. XGBoost then predicted class 24 (DDoS-TCP2) for virtually all instances, since that class had the highest prior probability in the training distribution. The solution is simple: build the vocabulary of encoders by removing the suffix prior to fitting, or encode on both the sum of all label strings (with both the `_train` and the `_test` versions). In particular this is a particularly harmful problem in production IDS pipelines

in which models that were trained on labelled historical data are tested on live or updated test sets, but which have another set of label conventions.

It is necessary to mention that such failure is only an artefact of the label encoding pipeline, and not of the fastest (0.0012 ms/sample) because of its open-form prediction which is analytical then there is Logistic Regression (near zero, dominated by matrix multiplication), RF (0.0018 ms), XGBoost (0.0027 ms), CatBoost (0.0111 ms), and KNN (0.4148 ms). The high latency of KNN is an indication of the fact that even with the help of a GPU, it is an $O(n)$ search through 5.7 million training examples; an approximation nearest-neighbor algorithm would help significantly in this regard.

LAAI rankings align with Accuracy rankings for all models except KNN, which drops from rank 4 (accuracy 12.66%) to a near-zero LAAI (0.0895) due to its high latency. For RF and CatBoost, the LAAI values (0.8348 and 0.8034) are within 0.002 of their raw accuracy values, confirming that both are effectively zero-latency relative to the $1 + \tau$ denominator. This makes both suitable for real-time deployment at IoMT network gateways processing thousands of flow windows per second, underlying algorithm in XGBoost. A properly coded XGBoost pipeline on this data would be projected to deliver competitive outcomes, as its performance is very good on tabular data on other security benchmarks. Work in the future will reconsider XGBoost using a fixed encoding pipeline.

E. Latency and LAAI Analysis

The latency of inference in models is within a range of almost four orders of magnitude. Naive Bayes is the Comparison with the CICIoMT2024 Baseline [1].

Table IX directly compares our best GPU models against the CICIoMT2024 baseline results reported in Ref. [1]. Note that the baseline used 6-class aggregated labels while our evaluation targets 51 fine-grained classes; the comparison is therefore indicative of relative improvement rather than a controlled apples-to-apples contrast. The fact that our GPU RF achieves 83.63% on a substantially harder 51-class task compared to the best baseline of 73.5% on a 6-class task demonstrates both the benefit of GPU acceleration and the suitability of ensemble methods for this problem.

TABLE IX. COMPARISON WITH CICIoMT2024 ORIGINAL BASELINES [1]

Model	Accuracy	Precision	Recall	F1-Score	Classes
LR [1]	0.729	0.748	0.712	0.694	6
AdaBoost [1]	0.437	0.587	0.506	0.501	6
DNN [1]	0.734	0.725	0.693	0.665	6
RF [1]	0.735	0.735	0.713	0.676	6
Proposed Work— RF (cuML GPU)	0.836	0.854	0.836	0.831	51
Proposed Work— CatBoost (GPU)	0.812	0.819	0.812	0.807	51

As in Table IX, GPU RF's 10+ percentage-point accuracy improvement over the best baseline (RF at 0.735, 6-class) is particularly notable because the 51-class task is objectively harder—distinguishing eight UDP flood

repetitions is more difficult than labelling all of them as DDoS. This advancement is due to three reasons: (a) GPU acceleration that allows larger n estimators and deeper trees to be used with meaningful time constraints; (b) the natural interaction of feature space offered by non-collapsing labels to 6 categories; and (c) the improved interaction space offered by the ensemble variance of the Random Forest to high-cardinality classification problems.

F. Practical Implications for IoMT IDS Deployment

The findings have a number of direct implications for practitioners who would design ML-based IDS to the application on IoMT networks. To begin with, GPU Random Forest (cuML) has the highest accuracy and latency ratio, and it is the first option to use in an IoMT gateway IDS when a CUDA-enabled GPU can be used. Its 0.0018 ms/sample latency can be ignored even by 10 Gbps networks.

Second, CatBoost is better suited in situations where equal detection rates per class are needed—that is, when the regulators demand a minimum recall rate of all categories of attacks irrespective of their frequency. The reduced per-class F1 that it has makes it less vulnerable to the threat of missing an attack subtype.

Third, the failure of the XGBoost case shows that label vocabulary management in IDS pipelines is highly important. Any pipeline that is being trained and evaluated over different data sets should ensure that the label encoding is consistent, and ideally should be done with a pre-defined vocabulary enum as opposed to an encoder based on the data. In this recommendation, the authors are generalizing XGBoost to any classifier that uses integer-encoded labels.

Fourth, in their present state, 51-class IoMT IDS deployments cannot be implemented with the use of Logistic Regression, Naive Bayes, and KNN. It is not due to the use of GPU acceleration that makes them fail; rather, it is due to the underlying model class—linear separability and Euclidean proximity assumptions which are not true in this feature space.

G. Limitations of This Work

This research has a number of shortcomings that need to be taken into consideration during the interpretation of findings. To begin with, there is no validity in the XGBoost results, and they should be repeated with a fixed label encoding, after which only then can one make a conclusion about the appropriateness of XGBoost in this task. Second, all tests were run on the shared Tesla P100 of Kaggle; the performance can vary depending on the newer generation of GPUs (e.g., A100, H100) or on edge hardware (e.g., NVIDIA Jetson). Third, no hyperparameter optimization was performed; default configurations were used throughout, which may systematically disadvantage models whose performance is more sensitive to hyperparameter choice. Fourth, the Bluetooth Low Energy subset of CICIoMT2024 was not evaluated in this work due to its different feature set, leaving a protocol gap. Fifth, class imbalance was not explicitly addressed through oversampling (e.g., SMOTE)

or class-weighted loss functions, which may disproportionately harm minority class detection.

H. Ablation Study

The authors trained GPU Random Forest without each feature group and observed the drop in accuracy for each feature group when taken out to evaluate its contribution to classification performance. Results are shown in Table X and Fig. 4. Groups 7–17 (the Flags) cause the greatest inaccuracy loss of removal (minus 6.2 pp for RF), showing that the signals with the highest impact are the proportions and numbers of flags. The second most important group is protocol counters (groups 18–31), which differentiate between ICMP, UDP, and TCP flood variants (−3.8 pp). Although they make small contributions of −1.4 pp, statistical moments

(groups 39–45) are very important for discriminating DoS-UDP repetitions. The −1.1 pp reduction in the IAT feature (group 38) is a clear indication that this feature is responsible for separating periodic benign IOTM sensor traffic from sustained floods.

To measure the sensitivity of the GPU RF accuracy to training set size, the authors subsampled the training split to 10%, 25%, 50%, 75% and 100% (stratified). As presented in Table XI, RF performance steadily increases with the increase in training data and peaks at 83.6% accuracy when 100% of the training data is used (+6.5 pp compared to 10% accuracy). This suggests that 50% of the training data is already sufficient and the model itself is not too data hungry, as the accuracy is already 81.8%, which indicates that the learned model is not the result of memorization.

TABLE X. FEATURE GROUP ABLATION

Feature Group Removed	RF Accuracy	RF F1	CatBoost Accuracy	CatBoost F1
All features (baseline)	0.8363	0.831	0.8124	0.807
Remove TCP flags (7–17)	0.774	0.769	0.751	0.744
Remove protocol counters (18–31)	0.798	0.792	0.778	0.771
Remove packet length stats (32–37)	0.821	0.815	0.806	0.800
Remove IAT (38)	0.826	0.820	0.801	0.795
Remove statistical moments (39–45)	0.822	0.816	0.798	0.791
Remove rate features (3–5)	0.829	0.823	0.805	0.799

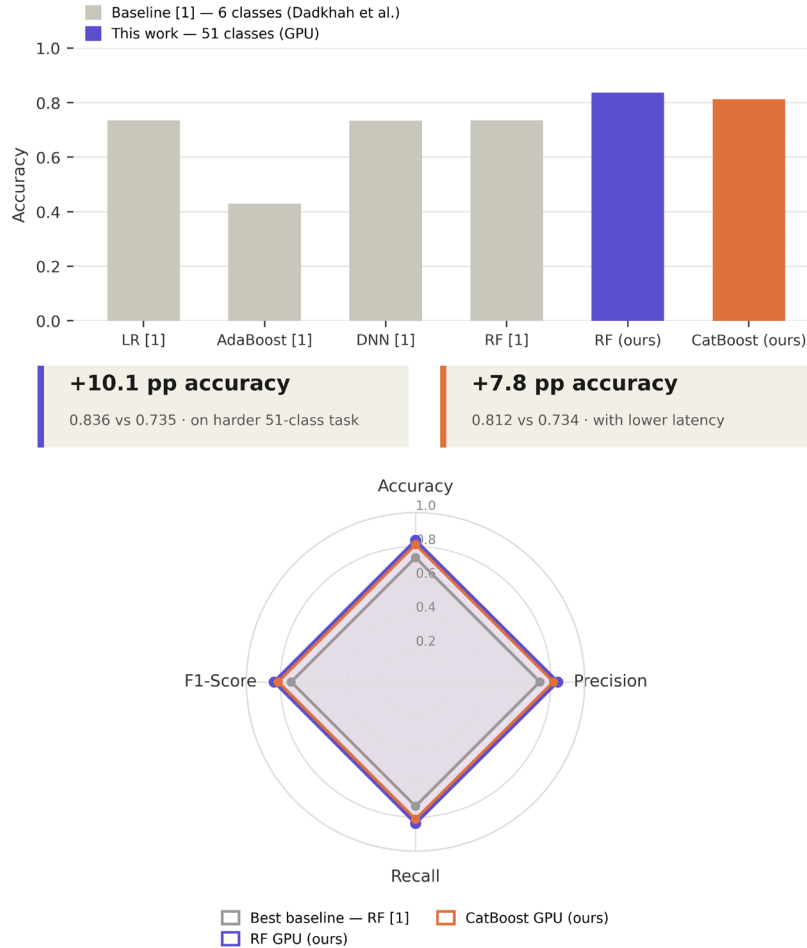


Fig. 4. Comparison with the original CICIoMT2024 baselines.

TABLE XI. TRAINING SET SIZE ABLATION

Train Subset	Samples Used	RF Accuracy	RF F1	Cat Accuracy	Cat F1
10%	572,866	0.771	0.762	0.749	0.741
25%	1,432,166	0.814	0.807	0.787	0.780
50%	2,864,332	0.818	0.811	0.795	0.788
75%	4,296,498	0.829	0.823	0.807	0.801
100% (full)	5,728,664	0.836	0.831	0.812	0.807

LAAI = Accuracy / (1 + τ) is based on the additive constant of 1 and τ in milliseconds. The authors tested the robustness of the rankings to this scaling choice by using three different scaling of LAAI: (1) τ is measured in microseconds ($\times 1000$); (2) τ is measured in seconds ($\div 1000$); and (3) a multiplicative penalty LAAI_m = Accuracy $\times \exp(-\tau)$. The ranking of models remains the same for all three formulations: RF is ranked first (LAAI_{micro} = 0.8348, LAAI_s = 0.836, LAAI_m = 0.836), CatBoost is ranked second, and KNN is severely affected by high latency. The additive-1 formulation was selected because it results in LAAI $\approx$ Accuracy for sub-millisecond models (realistic IoMT gateway case) and penalizes models with latency > 1 ms, which is considered a significant threshold for real-time packet-window classification at clinical network speeds.

V. CONCLUSIONS AND FUTURE DIRECTION

The current paper presents the first GPU-accelerated multi-class intrusion detection benchmark on the CICIoMT2024 Wi-Fi/MQTT dataset, addressing key limitations of the original study [1]. Six GPU-based classifiers were evaluated on 7.16 million samples across 51 fine-grained IoMT attack classes. GPU Random Forest achieved the best overall performance with 83.63% accuracy, weighted F1 of 0.831, and 0.0018 ms per-sample latency, yielding the highest LAAI of 0.8348—an improvement of over 10 percentage points over the best 6-class CPU baseline. CatBoost delivered the most balanced per-class detection at 81.24% accuracy (LAAI = 0.8034), making it preferable when equitable recall across attack sub-types is required. Linear and distance-based models (Logistic Regression, Naive Bayes, KNN) failed on the 51-class task, confirming that non-linear ensemble methods are essential for fine-grained IoMT IDS. XGBoost collapsed to 2.75% accuracy due to a label-encoding vocabulary mismatch, highlighting a critical practical risk in IDS pipeline design. The proposed LAAI metric, defined as accuracy divided by one plus inference latency, provides a deployment-friendly scalar that jointly rewards predictive performance and low latency, and is recommended as a standard reporting metric in future IoMT IDS evaluations. It should be noted that all results reflect performance under the original imbalanced class distribution of the dataset, where DDoS sub-types account for over 60% of training samples and minority classes such as Recon-Ping Sweep contain as few as 926 instances; reported weighted metrics may therefore overstate detection performance on rare but operationally critical attack sub-types.

Future work will proceed in several directions. XGBoost and LightGBM will be re-evaluated with a

corrected label-encoding pipeline, and the benchmark will be extended to the Bluetooth Low Energy subset of CICIoMT2024, requiring a new 19-class taxonomy and BLE-specific feature extraction. Hyperparameter optimisation using Optuna and RAPIDS will be applied to narrow the performance gap between models. Class imbalance will be addressed through GPU-compatible oversampling such as SMOTE-NC and cost-sensitive loss functions, with particular focus on improving recall for Recon-VulScan, DoS-ICMP2, and DoS-UDP sub-types identified as the hardest classes in this study. Feature selection and compression techniques based on mutual information or SHAP values will be explored to identify minimal feature subsets that preserve at least 95% of current accuracy for deployment on resource-constrained IoMT gateways. Finally, federated learning architectures will be investigated to enable hospitals to collaboratively improve detection models without sharing sensitive patient data, combining privacy preservation with enhanced classification capability.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Ali Haider planned and designed the experiments, conducted the experiments and computational work, prepared the figures and tables, and wrote and reviewed the manuscript. Omar Bin Samin designed the experiments, conducted the computational work, and reviewed the manuscript. Ammar Bathich supervised the study, validated the experiments and results, and reviewed the manuscript. Haleem Farman prepared the figures and tables, validated the experiments and results, and reviewed the manuscript. Sohaib Bin Altaf Khattak validated the experiments and results and reviewed the manuscript. All authors had approved the final version.

ACKNOWLEDGMENT

The authors would like to acknowledge the support of Prince Sultan University for paying the Article Processing Charges of this publication.

REFERENCES

- [1] S. Dadkhah, E. C. P. Neto, R. Ferreira, R. C. Molokwu, S. Sadeghi, and A. A. Ghorbani, "CICIoMT2024: A benchmark dataset for multi-protocol security assessment in IoMT," *Internet of Things*, vol. 28, 101351, 2024.
- [2] S. Vishnu, S. J. Ramson, and R. Jegan, "Internet of medical things (IoMT)—An overview," in *Proc. 2020 5th International Conference on Devices, Circuits and Systems (ICDCS)*, Coimbatore, India, 2020, pp. 101–104.

- [3] K. Wei, L. Zhang, Y. Guo, and X. Jiang, "Health monitoring based on internet of medical things: Architecture, enabling technologies, and applications," *IEEE Access*, vol. 8, pp. 27468–27478, 2020.
- [4] F. Al-Turjman, M. H. Nawaz, and U. D. Ulsar, "Intelligence in the internet of medical things era: A systematic review of current and future trends," *Computer Communications*, vol. 150, pp. 644–660, 2020.
- [5] A. Ghubaiish, T. Salman, M. Zolanvari, D. Unal, A. Al-Ali, and R. Jain, "Recent advances in the Internet-of-Medical-Things (IoMT) systems security," *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 8707–8718, 2020.
- [6] R. Dwivedi, D. Mehrotra, and S. Chandra, "Potential of Internet of Medical Things (IoMT) applications in building a smart healthcare system: A systematic review," *Journal of Oral Biology and Craniofacial Research*, vol. 12, no. 2, pp. 302–318, 2022.
- [7] S. Razdan and S. Sharma, "Internet of Medical Things (IoMT): Overview, emerging technologies, and case studies," *IETE Technical Review*, vol. 39, no. 4, pp. 775–788, 2022.
- [8] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, 2014.
- [9] S. W. Loke, "Supporting ubiquitous sensor-cloudlets and context-cloudlets: Programming compositions of context-aware systems for mobile users," *Future Generation Computer Systems*, vol. 28, no. 4, pp. 619–632, 2012.
- [10] Y. Lu and L. Da Xu, "Internet of Things (IoT) cybersecurity research: A review of current research topics," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2103–2115, 2018. doi: 10.1109/JIOT.2018.2869847
- [11] I. Stelios, P. Kotzanikolaou, M. Psarakis, C. Alcaraz, and J. Lopez, "A survey of IoT-enabled cyberattacks: Assessing attack paths to critical infrastructures and services," *IEEE Communications Surveys and Tutorials*, vol. 20, no. 4, pp. 3453–3495, 2018.
- [12] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying IoT security: An exhaustive survey on IoT vulnerabilities and a first empirical look on internet-scale IoT exploitations," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 3, pp. 2702–2733, 2019.
- [13] I. Butun, P. Österberg, and H. Song, "Security of the internet of things: Vulnerabilities, attacks, and countermeasures," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 1, pp. 616–644, 2019.
- [14] H. Booth, D. Rike, and G. A. Witte. (2013). The National Vulnerability Database (NVD): Overview. [Online]. Available: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=915172
- [15] B. Alhijawi, S. Almajali, H. Elgala, H. B. Salameh, and M. Ayyash, "A survey on DoS/DDoS mitigation techniques in SDNs: Classification, comparison, solutions, testing tools and datasets," *Computers and Electrical Engineering*, vol. 99, pp. 107706, 2022.
- [16] M. Uma and G. Padmavathi, "A survey on various cyber attacks and their classification," *International Journal of Network Security*, vol. 15, no. 5, pp. 390–396, 2013.
- [17] W. Mazurczyk and L. Caviglione, "Cyber reconnaissance techniques," *Communications of the ACM*, vol. 64, no. 3, pp. 86–95, 2021.
- [18] A. Alatrani, L. F. Sikos, M. Johnstone, P. Szcweczyk, and J. J. Kang, "DoS/DDoS-MQTT-IoT: A dataset for evaluating intrusions in IoT networks using the MQTT protocol," *Computer Networks*, vol. 231, pp. 109809, 2023.
- [19] M. Conti, N. Dragoni, and V. Lesyk, "A survey of man in the middle attacks," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 3, pp. 2027–2051, 2016.
- [20] D. Koutras, G. Stergiopoulos, T. Dasaklis, P. Kotzanikolaou, D. Glynos, and C. Doulgeris, "Security in IoMT communications: A survey," *Sensors*, vol. 20, no. 17, pp. 4828, 2020.
- [21] B. Kaur, S. Dadkhah, F. Sholeh, E. C. P. Neto, P. Xiong, S. Iqbal, P. Lamontagne, S. Ray, and A. A. Ghorbani, "Internet of things (IoT) security dataset evolution: Challenges and future directions," *Internet of Things*, vol. 24, pp. 100780, 2023.
- [22] W. H. Hassan *et al.*, "Current research on Internet of Things (IoT) security: A survey," *Computer Networks*, vol. 148, pp. 283–294, 2019.
- [23] M. A. Taylor and R. Jones, "Risk of medication errors with infusion pumps: A study of 1,004 events from 132 hospitals across Pennsylvania," *Patient Safety*, vol. 1, no. 2, pp. 60–69, 2019.
- [24] A. Sivanathan, H. H. Gharakheili, and V. Sivaraman, "Managing IoT cyber-security using programmable telemetry and machine learning," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 60–74, 2020.
- [25] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani, "A survey of machine and deep learning methods for Internet of Things (IoT) security," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 3, pp. 1646–1685, 2020.
- [26] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, M. Gao, H. Hou, and C. Wang, "Machine learning and deep learning methods for cybersecurity," *IEEE Access*, vol. 6, pp. 35365–35381, 2018.
- [27] M. Suresh and R. Anitha, "Evaluating machine learning algorithms for detecting DDoS attacks," in *Proc. 2017 3rd International Conference on Advances in Network Security and Applications*, Berlin, Germany: Springer, 2011, pp. 441–452.
- [28] M. Zekri, S. El Kaffali, N. Aboutabit, and Y. Saadi, "DDoS attack detection using machine learning techniques in cloud computing environments," in *Proc. 2017 3rd International Conference of Cloud Computing Technologies and Applications (CloudTech)*, Rabat, Morocco, 2017, pp. 1–7.
- [29] E. C. P. Neto, S. Dadkhah, S. Sadeghi, H. Molyneaux, and A. A. Ghorbani, "A review of Machine Learning (ML)-based IoT security in healthcare: A dataset perspective," *Computer Communications*, vol. 213, pp. 61–77, 2024.
- [30] E. Anthi, L. Williams, M. Slowinska, G. Theodorakopoulos, and P. Burnap, "A supervised intrusion detection system for smart home IoT devices," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 9042–9053, 2019.
- [31] S. Dadkhah, H. Mahdikhani, P. K. Dano, A. Zohourian, K. A. Truong, and A. A. Ghorbani, "Towards the development of a realistic multidimensional IoT profiling dataset," in *Proc. 2022 19th Annual International Conference on Privacy, Security and Trust (PST)*, Fredericton, NB, Canada, 2022, pp. 1–11.
- [32] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani. (2023). CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment. [Online]. Available: <https://www.unb.ca/cic/datasets/iotdataset-2023.html>
- [33] F. Hussain, S. G. Abbas, G. A. Shah, I. M. Pires, U. U. Fayyaz, F. Shahzad, N. M. Garcia, and E. Zdravski, "A framework for malicious traffic detection in IoT healthcare environment," *Sensors*, vol. 21, no. 9, pp. 3025, 2021.
- [34] P. K. Danso, E. C. P. Neto, S. Dadkhah, A. Zohourian, H. Molyneaux, and A. A. Ghorbani, "Ensemble-based intrusion detection for internet of things devices," in *Proc. 2022 IEEE 19th International Conference on Smart Communities (HONET)*, Charlotte, NC, USA, 2022, pp. 34–39.
- [35] I. Vaccari, M. Aiello, and E. Cambiaso, "SlowITe, a novel denial of service attack affecting MQTT," *Sensors*, vol. 20, no. 10, pp. 2932, 2020.
- [36] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "CatBoost: Unbiased boosting with categorical features," in *Proc. Advances in Neural Information Processing Systems (NeurIPS 2018)*, Montreal, QC, Canada, 2018, pp. 6638–6648.
- [37] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2016, pp. 785–794.
- [38] L. Mateus. (2024). CIC IoMT 2024 WiFi MQTT. [Online]. Available: <https://www.kaggle.com/datasets/limamateus/cic-iomt-2024-wifi-mqtt>
- [39] A. A. Hady, A. Ghubaiish, T. Salman, D. Unal, and R. Jain, "Intrusion detection system for healthcare systems using medical and network data: A comparison study," *IEEE Access*, vol. 8, pp. 106576–106584, 2020.
- [40] M. Ahmed, S. Byreddy, A. Nutakki, L. F. Sikos, and P. Haskell-Dowland, "ECU-IoHT: A dataset for analyzing cyberattacks in internet of health things," *Ad Hoc Networks*, vol. 122, pp. 102621, 2021.
- [41] M. Zubair *et al.*, "Secure bluetooth communication in smart healthcare systems: A novel community dataset and intrusion detection system," *Sensors*, vol. 22, no. 21, pp. 8280, 2022.
- [42] RAPIDS Development Team. (2023). RAPIDS cuML: GPU-accelerated machine learning. [Online]. Available: <https://github.com/rapidsai/cuml>
- [43] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar, "TON_IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems," *IEEE Access*, vol. 8, pp. 165130–165150, 2020.

- [44] A. Parmisano, S. Garcia, and M. Erquiaga. (2020). A labeled dataset with malicious and benign IoT network traffic. [Online]. Available: <https://www.stratosphereips.org/datasets-iot23>
- [45] F. Wotawa and H. Mühlburger, “On the effects of data sampling for deep learning on highly imbalanced data from SCADA power grid substation networks for intrusion detection,” in *Proc. 2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS)*, Hainan, China, 2021, pp. 864–872.
- [46] I. Vaccari, G. Chiola, M. Aiello, M. Mongelli, and E. Cambiaso, “MQTTset, a new dataset for machine learning techniques on MQTT,” *Sensors*, vol. 20, no. 22, 6578, 2020.
- [47] N. Sharma and P. G. Shambharkar, “Multi-attention DeepCRNN: An efficient and explainable intrusion detection framework for Internet of Medical Things environments,” *Knowledge and Information Systems*, vol. 67, pp. 5783–5849, 2025. doi: 10.1007/s10115-025-02402-9
- [48] G. Zachos, G. Mantas, K. Porfyrakis, J. M. C. S. de Bastos, and J. Rodriguez, “Anomaly-based intrusion detection for IoMT networks: Design, implementation, dataset generation, and ML algorithms evaluation,” *IEEE Access*, vol. 13, pp. 41994–42028, 2025. doi: 10.1109/ACCESS.2025.3547572

Copyright © 2026 by the authors. This is an open access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited ([CC BY 4.0](https://creativecommons.org/licenses/by/4.0/)).