Designing an Adaptive Effective Intrusion Detection System for Smart Home IoT

A Device-Specific Approach with SDN Deployment

Hassen Sallay
College of Computing, Umm Al-Qura University, Makkah, KSA

***Abstract*—As the ubiquity of IoT devices in smart homes escalates, so does the vulnerability to cyber threats that exploit weaknesses in device security. Timely and accurate detection of attacks is critical to protect smart home networks. Intrusion Detection Systems (IDS) are a cornerstone in any layered security defense strategy. However, building such a system is challenging given smart home devices' resource constraints and behaviors' diversity. This paper presents an adaptative IDS based on a device-specific approach and SDN deployment. We categorize devices based on traffic profiles to enable specialized architectural design and dynamically assign the suitable detection model. We demonstrate the IDS efficiency, effectiveness, and adaptability by thoroughly benchmarking an ensemble of machine learning models, mainly tree ensemble models and extreme learning machine variants, on the up-to-date IoT CICIoT2023 security dataset. Our IDS multi-component device-aware architecture leverages software-defined networking and virtualized network functions for scalable deployment, with an edge computing design to meet strict latency requirements. The results reveal that our adaptive model selection ensures detection accuracy while maintaining low latency, aligning with the critical requirement of real-time accuracy and adaptability to smart home devices' traffic patterns.**

Keywords—Smart home; IoT; IDS; taxonomy, architecture; SDN; ELM

I. Introduction

The rapid growth of the smart home market broadly opens the doors to several threats to people's security and privacy. People are often unaware of security vulnerabilities, and manufacturers fail to prioritize security. This combination leads to a growing attack surface for hackers to exploit. Indeed, it is well known that many smart home devices, including IP cameras, smart locks, smart lighting systems, etc., contain vulnerabilities that attackers can exploit to intrude into home networks. Successful intrusions into IoT devices can allow hackers to not only steal sensitive user data but also take control of critical devices. Hence, there is a growing need for intelligent security systems to detect abnormal behaviors and attacks on smart home IoT devices in real time.

Since no one-size-fits-all security solution exists, a defense-in-depth approach and appropriate design and implementation should be context-aware to protect against threats and specific attack vectors. Among the complementary tools in the security layered defense comes network intrusion detection systems (NIDSs). They are security tools that continuously analyze traffic to identify intrusions and attacks. Traditional IDS employ signature-based detection, which matches known attack patterns. More advanced anomaly detection techniques spot statistical deviations from normal traffic to surface previously unseen attacks. However, building accurate intrusion detection models for IoT is challenging due to several factors. IoT devices have much more resource constraints than traditional computing systems and exhibit complex and dynamic behaviors. Moreover, the constantly evolving threats and vulnerabilities must be efficiently well-tracked for an adaptive security defense in the smart home context. Thus, knowing the ground truth for device and traffic features will be useful in tackling intrusion security challenges posed by smart home environments.

Several research efforts have been spent to tackle these challenges. The research in [1] provided a comprehensive review of intrusion detection systems using machine and deep learning in IoT, discussing challenges, solutions, and future directions. They emphasized the need for efficient and accurate detection methods but did not propose a specific architecture or implementation. The study in [2] surveys network intrusion detection for IoT security based on learning techniques, highlighting the importance of efficient learning algorithms for smart home security. It deeply and thoroughly explores recent works focusing on machine learning techniques. However, its scope does not include architectural design issues such as adaptability and real-time requirements. The study in [3] introduced a deep learning application for invasion detection in industrial IoT sensing systems. While this work is relevant for industrial applications, it may not directly translate to smart home environments due to different operational constraints and attack vectors. The research in [4] proposes an integrated multilayered framework for IoT intrusion decisions and instantiates it for the industrial IoT. Although the framework can be instantiated to the smart home context, the paper did not specify the architectural design and deployment. All these works raised the flag that most existing methods overlook key IoT constraints like low latency, dynamic device behaviors, and resource limitations that impact real-world-scale adoption.

We also cite some works that gave us insights to develop our proposed solution. The study in [5] proposed an intrusion detection system using an Online Sequence Extreme Learning Machine in the advanced metering infrastructure of smart grids. Their model focused on sequential data processing, which is pertinent and can be adapted to the continuous monitoring required in smart homes. The research in [6] discussed intrusion detection in fog computing and Mobile

Edge Computing. This work is particularly relevant as it considers the edge computing paradigm, which is increasingly adopted in smart home IoT. However, it does not discuss the use of machine learning in intrusion detection. The study in [7] presented a self-configurable cyber-physical intrusion detection system for smart homes using reinforcement learning. This system's adaptability to changing conditions in smart homes is a significant step towards dynamic and responsive security systems. The research in [8] explored a hybrid approach using an artificial immune system for intrusion detection in smart home networks. This work highlighted the potential of ELM for fast learning and generalization but did not focus on the real-time aspect of intrusion detection. The study in [9] integrates the software-defined networking, machine learning, and manufacturer usage descriptions standard with an intrusion detection and prevention system to assess its influence on network security. While including standards-based ingredients is interesting, their work is limited to manufacturing and does not consider the architectural effectiveness and network traffic characteristics.

For the traffic characterization, [10] recognizes the importance of understanding IoT data characteristics for modeling the data bursts typical of IoT use cases, and it introduces an advanced ON/OFF traffic modeling approach tailored for the varied applications within a smart city context. While the work is pivotal for statistical modeling of the IoT traffic, it does not consider their solutions' architectural design and deployment. The study in [11] provides insights into IoT traffic characteristics in the specific context of smart home and campus environments. They found that IoT devices exhibit periodic behavior with significant idle time. The devices generate a small amount of traffic, and most communicate with a small number of remote servers, often located in the same country as the device. The study also found several security and privacy issues, including devices communicating over unencrypted channels and devices communicating with servers in countries known for privacy concerns. This paper aims to design and build a flexible, scalable IDS that efficiently ensures security defense in real smart home environments without losing generality and adaptability. The contributions are:

- We introduce a device-aware approach that categorizes IoT devices based on their traffic profiles and behaviors, leading to a more tailored and efficient detection process that can adapt to the heterogeneous nature of smart home devices.
- Our solution employs an ensemble of optimized machine learning models, including extreme learning machine variants chosen based on the device category, balancing the tradeoffs between speed, accuracy, and resource usage.
- We provide an experimental evaluation using an up-to-date security dataset, demonstrating the effectiveness of our approach in a realistic smart home context.
- We design a multi-component IDS architecture using network traffic profiles for real-time intrusion detection in smart home IoT environments. Our architecture leverages software-defined networking (SDN) and virtualized network functions (VNFs), allowing for a flexible and scalable deployment that can be adapted for both cloud and edge computing scenarios. We mainly opted for an edge computing-based deployment on an SDN testbed to meet strict latency requirements.

The remainder of this paper is organized as follows: Section II presents our methodology steps. Section III characterizes the smart home devices' traffic and presents a simple traffic-based taxonomy. Section IV shows the architecture design and deployment. Section V presents the benchmarking of the machine learning models. Section VI shows the benchmarking results. Section VII concludes the paper and gives some future works.

II. Methodology

We propose a two-stage methodology within four steps to develop our intrusion detection system:

A. *Devices' traffic characterization*

1) Explore and categorize the commonly used devices in the smart home environment.

2) Characterize the traffic devices and build a traffic-centric devices taxonomy.

B. *Architectural Design and Model Selection*

1) Design and deployment of a smart home IDS-tailored architecture.

2) Benchmark the ML models on a recent dataset and select the appropriate model based on the previous steps.

More specifically, we start by enumerating devices and device/data categories to understand the ecosystem, and then we characterize traffic patterns and classify devices into a useful taxonomy. We are leveraging this knowledge to design and deploy suitable IDS architecture. Then, select appropriate machine learning models that detect intrusions optimized for the specifics of the smart home domain. The result is an IDS purpose-built to the unique smart home environment versus more generic systems. The key rationale is that threats exploit specific device vulnerabilities and traffic flows in the smart home, so an IDS must be aware of these devices and patterns to identify attacks. The proposed methodology builds this intrinsic knowledge by examining the ecosystem to customize the IDS. This context-aware solution can better distinguish attacks from normal traffic and has utility detecting intrusions that more generic learning-based systems may overlook in the IoT setting.

III. Devices' Traffic Characterization

A typical smart home would include various IoT devices commonly used, such as smart thermostats, smart lighting systems, smart security cameras, smart locks, smart appliances (e.g., refrigerators, washing machines), smart speakers or home assistants, and smart TVs. Table I characterizes common consumer IoT devices along three dimensions: subcategories based on features, key devices' behavior patterns, and data reflecting network traffic patterns in size and time. Grouping into broader categories like smart speakers while still enumerating specific device types enables roll-up summarization and device-specific analysis. Smart TVs, for instance, have subcategories of basic models focused on video streaming with basic controls versus smart TVs, which have an integrated app platform enabling third-party applications like Netflix and YouTube. The use cases cover on-demand video and accessing these apps for entertainment and information. This expanded functionality versus standalone streaming leads to more diverse, multimodal traffic encompassing control commands, actual video data, and app communications.

Referring to Table I and the data nature, we see a mix of primarily unimodal control traffic for simpler devices like lightbulbs, thermostats, and locks with relatively straightforward command and monitoring use cases. Meanwhile, more advanced devices like cameras, speakers, and fridges demonstrate bimodal or multimodal traffic indicative of more mixed media, including audio, video, and firmware downloads, resulting in variable network utilization.

The complexity arising from supporting multiple integrated apps paired with streaming video in one device results in a complex traffic profile that may require advanced analytic approaches beyond simple machine learning algorithms to adequately characterize if simple models prove to have insufficient descriptive capability and predictive accuracy. However, for unimodal traffic, simpler models should suffice without overcomplicating analysis.

Thus, we built a simple taxonomy of smart home IoT devices based on their network traffic characteristics:

- Streaming Devices [Smart speaker, IP camera, Voice assistant robot] (Traffic patterns are: (1) Bimodal packet size distribution (small control + large streaming packets), (2) Bursty packet timing during streaming, and (3) Higher and variable traffic volume.)
- Intermittent Control Devices [Smart lightbulb, Smart thermostat, Smart fridge, Smart doorbell, Smart blinds, Irrigation controller] (Traffic patterns are: (1) Uniformly small packet sizes, (2) Periodic keepalives + event-driven commands, and (3) Low traffic volume with occasional spikes)
- Monitoring Devices [Smoke detector, Motion sensor, Door/window sensor] (Traffic patterns are: (1) Small packets for status updates, (2) Sporadic or periodic timing, and (3) Very low traffic volume)
- Actuators [Smart lock, Garage door opener, Smart plug] (Traffic patterns are: (1) Small command packets, (2) Event-driven timing (3) Extremely low traffic volume).

Based on this taxonomy, considering device behaviors and traffic profiles, we categorize home IoT devices into two broad classes for architectural design:

- High-throughput devices include video cameras, media hubs, etc., generating high volumes of multimedia traffic. The patterns are more complex and variable.
- Low-throughput devices consist of simpler sensors and controllers for lighting, smoker detectors, etc., with minimal traffic. The patterns tend to be regular and predictable.

Accordingly, in the next section, we propose a multi-component IDS architecture to secure the smart home.

TABLE I. DEVICES TRAFFIC CHARACTERISTICS

Device Category	**Device**		**Data**	
	Subcategory	***Behavior***	***Size***	***Timing***
Smart TV	Basic, Smart	Intermittent streaming	Bimodal (control + audio packets)	Bursty during use and periodic otherwise
Smart Speaker	Audio streaming, Voice assistant, Smart display	Mostly control commands	Uniformly small	Event-driven/periodic
Smart lightbulb	Tunable white, RGB, Motion sensor	Continuous video	Bimodal (small + large video packets)	Periodic real-time streaming
IP camera	Video doorbell, Baby monitor, Security camera	Infrequent controls	Uniformly small	Periodic polling + event-driven
Smart thermostat	Self-contained, HVAC integrated	Intermittent traffic	Bimodal (control + firmware updates)	Periodic sensors updates
Smart fridge	Display model, Bottom-freezer model	Sparse controls	Uniformly small	Infrequent periodic keepalives
Smart lock	Bluetooth, WiFi, Z-Wave	Activated when used	Small control packets	Event-driven only
Garage door opener	WiFi/Bluetooth connected, Remote controlled	Intermittent controls	Small control packets	Periodic + event-driven
Smart blinds	Motorized, App/voice controlled	Sparse status report	Small power toggling packets	Periodic status updates
Smart plug	Controllable, Monitored	Event-driven alerts	Small alert packets	Sporadic alarms, periodic heartbeats
Smoke detector	Integrated, Smart alarm	Regularly scheduled operation	Small control/status packets	Periodic polling + daily schedules
Irrigation controller	App connected, Weather adjusted	Intermittent streaming	Packet Size Distribution	Packet Timing Distribution

IV. Architectural Design and Model Selection

A. Architecture Design

The core of our proposed intrusion detection system comprises a multi-component architecture tailored to secure diverse IoT devices in smart home environments. Our design meets the following key requirements: real-time detection capability, adaptability to evolving behaviors, detection accuracy for known and zero-day attacks, and computational efficiency to operate given smart home resource constraints.

The modular architecture allows customizing specific components to address deployment-specific needs. The IoT devices would commonly be connected to a local network, including a gateway or router, to manage network traffic and connect to the Internet. The network commonly includes a mix of wired and wireless connections, depending on the specific devices used, and HTTP(S), MQTT, CoAP, and Zigbee are the common IoT-used protocols. Furthermore, smart home often has a network firewall and other basic security measures for each device the manufacturer provides. The security threats landscape includes denial or distributed denial of service (DoS/DDoS) attacks, malware or ransomware attacks, unauthorized access or intrusion attempts, and data breaches or exfiltration attempts. We add a layer for intrusion detection that stays behind the firewall. Mainly the NIDS system includes the following components:

- Traffic Inspector capturing and pre-processing all device traffic flows. It mainly (1) captures raw network traffic using port mirroring, (2) extracts flow-based features like source/destination IP, ports, packet sizes, etc., (3) tags flow with device identities from logs, and (4) forwards processed flows to Device Profiler.
- Device Profiler identifies and assigns device type to a high/low throughput category. It mainly (1) maintains an inventory of identified IoT devices, (2) classifies devices into high or low throughput groups, and (3) pushes device type and group to ML Model Selector.
- ML Model Selector chooses the optimal intrusion detection model for that device type. It mainly (1) houses a catalog of optimized ML models for each device group, (2) models tailored for the complexity and behaviors of that group, (3) queries device group for a flow from Device Profiler and (4) dynamically it selects the matching model for anomaly detection.
- Model Repository contains specialized ML models tailored for each device class. It mainly (1) stores specialized ML models, (2) contains different algorithms that suit traffic complexities, and (3) includes models pre-trained on normal and attack device data.
- Intrusion Detector to analyze traffic for intrusion using a selected model. It mainly (1) receives network traffic flow features, (2) feeds to Model Selector chosen model, (3) the model analyzes the sequence for intrusions, and (4) the classifier flags intrusion if found.
- Alert Manager raising intrusion alerts as needed with attack details. It mainly (1) collects intrusion alerts from Intrusion Detector, (2) provides details like affected device attack type, and (3) raises notifications to admin and response systems.

We first utilize a Traffic Inspector module that captures raw network traffic using port mirroring techniques. It then extracts flow-based features like source and destination IPs, ports, packet sizes, and tag flows to specific device identities obtained from logs. The processed traffic flows are forwarded to a Device Profiler component, which maintains an inventory of devices identified on the network. Leveraging both domain knowledge, the Device Profiler categorizes devices into either high throughput or low throughput groups. High throughput devices like cameras and media hubs generate higher volumes of multimedia network traffic with more complex and variable patterns. In contrast, simpler sensors and controllers constitute the low throughput group with minimal and regular traffic.

The device type and group information are passed into an ML Model Selector module that maintains a catalog of specialized models tailored for each device group. When the Model Selector receives a query with the device group for a particular traffic flow, it dynamically selects the matching specialized model to analyze that flow for intrusions. This model repository containing diverse algorithms suited for varying traffic complexities is pre-trained on normal and attack data generated from devices in the corresponding category.

An Intrusion Detector module takes the network traffic flow features and feeds them into the model instance chosen by the Model Selector for that flow. Based on previous learning, the selected model analyzes the sequence to detect intrusions, finally flagging likely security intrusions. Any intrusion alerts are collected by an Alert Manager, who provides details like the affected device and attack type to administrators and incident response systems.

B. Architecture Deployment

Our proposed intrusion detection system's components leverage software-defined networking (SDN) capabilities for efficient and flexible system deployment [12,13]. The SDN controller provides a central orchestration point for the various IDS modules [14]. Network switches are configured using SDN policies to mirror IoT traffic flows that need to be inspected, tapping them to feed into the IDS Traffic Inspector module. The centralized network view within the SDN control plane also enables mapping these flows to specific IoT devices on the network.

A software-defined implementation offers significant advantages in flexibility, programmability, and scalability. The centralized control plane greatly simplifies tapping into a high volume of IoT flows in dynamic environments while automating complex policy configurations needed for mirroring. Device profiles and policies can be updated easily as new IoT devices get added over time. SDN also enables large-scale deployments with intelligent traffic engineering and usage optimization across available IDS resources. Therefore, an SDN-based deployment for the intrusion detection infrastructure makes our IDS more agile and adaptive, mainly as smart home IoT adoption grows exponentially.

Practically, the core detection modules of the IDS, including the Device Profiler, ML model Selector, and Intrusion Detector, are implemented as virtualized network functions (VNFs). By leveraging VNFs and placing them flexibly on commodity servers, we scale out these modules on demand to meet the throughput needs of real-time detection across many IoT devices. As device diversity expands or new models are added to the model repository, more VNF instances can be spun up accordingly. The global view allows the SDN controller to intelligently load balance traffic flows across the VNF resources for optimal efficiency.

The VNF-based deployment can leverage both cloud and edge computing approaches: (1) Cloud-based Deployment where the VNFs for the IDS components like Traffic Inspector, Device Profiler, Model Repository, and Intrusion Detector can be hosted on virtual machines or containers in a private or public cloud. This allows leveraging cloud platforms' flexibility, scalability and managed services. The globally distributed nature of major cloud providers also allows VNFs to be placed closer to IoT deployments for lower latency. However, wide-area network traffic and cloud usage costs may be concerns. (2) Edge Computing Deployment, where we deploy the VNFs on edge servers directly located in smart homes. Edge computing overcomes cloud-based analysis's latency and bandwidth challenges by processing data locally. It provides better responsiveness for real-time intrusion detection [15, 16]. Edge servers can also interface with hardware accelerators for efficient ML model inference. While cloud and edge are viable deployment options, edge computing is better aligned to meet the low latency requirements for real-time intrusion detection across smart home installations. Indeed, the proximity of edge servers to IoT environments makes the IDS more adaptive.

The deployment experimentation can be performed by an SDN testbed where we integrate the edge computing-based deployment with the Mininet/Ryu [17]. Mainly, we set up edge computing nodes in the Mininet topology to host the VNFs (Device Profiler, Model repository, Intrusion Detector). These would consist of lightweight Docker containers. Then, we configure the Ryu controller to steer copies of IoT traffic flows to the nearest edge node for intrusion detection analysis. This mimics real-world edge deployment. The VNF containers process the mirrored device traffic, generate alerts if needed, and export IDS telemetry data. We expose the VNFs via REST APIs for integration with the Ryu controller and monitoring software and evaluate overall latency from the IoT devices to the edge-based IDS VNFs during attack scenarios in Mininet. We can then analyze the responsiveness, overhead, and accuracy relative to an Edge-based deployment. This deployment allows prototyping and demonstrating the benefits of edge computing for IoT environments, leveraging Ryu's programmability and Mininet's flexibility. Automated traffic steering to nearby edge nodes also validates the low latency premise.

C. *Architecture Suitability*

Following, we discuss how the proposed modular multi-components IDS architecture design and its SDN-based deployment, along with the Edge-computing technology, help to meet key requirements of real-time detection, adaptability, and accuracy:

1) *Real-time detection capability*: The lean and specialized machine learning models ensure low latency between packet capture by the Traffic Inspector and intrusion alert generation by the Intrusion Detector. In the next section, we will show that the selected models are optimized for efficiency without sacrificing detection accuracy. The virtualized deployment also allows dynamic scaling of detection modules to match incoming traffic volumes. Together, these allow the IDS to provide real-time, sub-second analysis of IoT traffic flows to meet real-time detection needs.

2) *Adaptability to evolving behaviors*: The feedback loop from the Device Profiler to the ML Model Selector allows the system to adapt to changes in device behaviors over time. As traffic patterns change, updated device profiles trigger selection of different models tailored to new behaviors. The models themselves, through re-training, will also adapt during operational use as they observe more data. This tight integration between device knowledge and flexible model selection allows the IDS to adjust to evolving IoT environments.

3) *Detection accuracy*: The model repository for the device category allows highly accurate intrusion detection based on specific device profiles. Tailoring models to capture different IoT devices' normal/attack behavior patterns results in a solution that outperforms one-size-fits-all approaches.

V. Intrusion Detection Models Benchmarking

A. *Methodology and Dataset*

As per our proposed methodology, we leverage the CICIoT2023 dataset in [18] to categorize smart home IoT devices based on network traffic profiles and select suitable ML models for intrusion detection accordingly. The CICIoT2023 dataset has been created to accelerate research into security analytics and intrusion detection systems tailored for smart home IoT environments. It contains network traffic captures from an extensive smart home IoT testbed comprising over 100 heterogeneous devices. The key value of CICIoT2023 lies in the 33 contemporary IoT-focused attacks spanning seven categories executed on the devices. Many of these attack types are unavailable in other IoT IDS datasets. The attacks leverage compromised IoT devices to penetrate the network, enabling the evaluation of multi-vector IoT threats. Therefore, CICIoT2023 is an invaluable, up-to-date resource for further research into robust intrusion detection tailored for smart home IoT environments facing escalating threats. Moreover, the CICIoT2023 experiments provide data-driven guidance for model selection in our multi-component IDS architecture that analyzes runtime traffic profiles to pick the optimal intrusion detection model tailored to IoT device behaviors and capacities.

The CICIoT2023 is an unbalanced dataset since it is attack intensive. We deploy the SMOT technique to generate a balanced dataset sample. Then, we subsample each dataset into low and high-traffic flow groups that align with our taxonomy of simple and complex IoT devices, respectively. The CICIoT2023 dataset contains 46 features describing the traffic flows. The flow rate in packets/second feature is the most discriminative feature for this clustering across all the available dataset attributes. Then, we experiment with various models on both traffic datasets (balanced and unbalanced), measuring evaluation and model efficiency metrics. Comparing these metrics reveals how different algorithms fare in detecting IoT intrusions for simple and complex device categories, respectively. Additionally, we benchmark the models on the full dataset to validate the improvements gained from our approach of tailored model selection per device traffic profiles. By correlating the model evaluation and efficiency metrics, we can determine the detection accuracy and precision vs. latency tradeoff and evaluate our meeting degree to our key IDS objectives around real-time alerts, adaptability to varying traffic volumes and attack types, and detection precision for IoT environments within typical resource constraints.

B. Models Selection

1) Intuitive analysis of models' suitability: We intuitively discuss here the suitability of the ML models for the IDS requirements in our context. Tree-based models construct multiple shallow decision trees. They capture nonlinear interactions and complex patterns like network attacks through branching decisions; the tree ensembles balance bias and variance. The tree architectures also suit evolving data through continuous model updates and handheld high-dimensional network data. Support vector machine (SVM) is known for its effectiveness with high dimensional multimodal data, but its complexity impacts real-time performance. While Deep learning models, based on many neural networks, identify complex patterns and capture sequential dependencies, helping detect multi-stage attacks, they require significant data and computing resources that may not suit resource-constrained IoT context. One interesting approach to reducing this complexity is the Extreme Learning Machine (ELM) [11], a fast, single-layer feedforward neural network for classification and regression. They randomly initialize input layer weights and analytically determine output weights. This allows very fast model training suited for real-time usage.

Intuitively, the tree ensemble and ELM class of models seem optimal for balancing efficiency, accuracy, and adaptability within typical IoT constraints. The modular architecture enables deploying complex models like support vector machine or deep learning techniques selectively for capable devices while using tree ensemble and ELM for most real-time detection. The Model Selector dynamically handles this model assignment per device profile. Based on this intuitive analysis, we selected the following set of ML models for benchmarking. Following is a brief description of each model:

2) Benchmarked models: We select 12 models to perform our benchmark. Six of them are variants of the ELM, which intuitively brings a promising adequacy for our design requirements. We experiment with its variants [19-23]:

- Kernel ELM is an extension of basic ELM that applies kernel functions like sigmoid, radial basis function (RBF), hyperbolic tangent (tanh), etc., to non-linearly map the input data to new feature spaces before output weight computation. This adds nonlinearity to improve model learning capability for complex patterns.
- Regularized ELM imposes additional constraints on optimizing the output weights matrix calculation. Regularization parameters control model complexity to prevent overfitting, enabling more robust intrusion detection.
- Weighted ELM introduces random scaling factors or weights when multiplying the input layer feature values during forward propagation. This acts as a regularizer like dropout techniques in neural networks, reducing inter-dependencies and improving generalizability.
- OS-ELM (Online Sequential ELM) is the online sequential version of ELM that processes streaming data instance-by-instance for model updates rather than batch learning. This fast incremental learning allows continuous real-time model adaptation, useful for evolving traffic in our context.
- Voting ELM is an ensemble method that trains multiple OS-ELM models on bootstrap samples of the original data. During prediction, the OS-ELM outputs are aggregated through voting to output the overall class.
- Bagging ELM is another ensemble technique using bootstrap sampling to train multiple OS-ELM models. The predictions are aggregated by weighted averaging rather than voting.
- The other six used ML models in the benchmarking are [24-27]:
- Logistic regression is a linear classification model that assigns probabilities to data points belonging to classes using the logistic/sigmoid function. It is fast to train but assumes linear decision boundaries.
- Decision Tree is a simple hierarchical model with branching decisions based on feature thresholds. It is interpretable but prone to overfitting with noisy IoT data.
- Random Forest is an ensemble method combining predictions from many uncorrelated decision trees. It averages out bias and variance for robust performance despite some complexity.
- AdaBoost is another ensemble technique that iteratively focuses on misclassified instances. It can reduce bias and variance errors despite the complexity cost.

- XGBoost is a scalable tree-boosting system for both classification and regression. It uses regularized model formalization for controllable complexity and prevents overfitting.
- LightGBM is a gradient-boosting framework specifically engineered for efficiency, performance, and lower memory usage.

C. *Performance Metrics*

- We choose the following key evaluation metrics for assessing our benchmark models' performance:
- Training Time (s): selecting efficient models is crucial for real-time model updates as new devices get added and data change in volume and nature. Therefore, lower training time is better.
- Prediction Time (s) measures classifying flow instances. It impacts the real-time intrusion alert generation capability. Therefore, a lower prediction time is better.
- Latency Time (s) is the sum of training and prediction times reflecting the end-to-end delay from data ingestion to producing an alert. This metric is directly relevant for real-time adaptative detection needs.
- Memory Usage (MB) measures the RAM consumed during model training and inference. It is important due to memory constraints in embedded IoT devices.
- Accuracy is the fraction of correctly classified instances. It provides an overall performance measure but can be misleading for imbalanced data.
- Precision is the fraction calculated by dividing True Positives over the sum of True Positives and False Positives. It is important to minimize false alarms which disrupt users.
- Recall is the fraction calculated by dividing True Positives by the sum of True Positives and False Negatives. It is crucial to maximize the detection of actual attacks and intrusions.
- F1 score is a harmonic mean of precision and recall balancing both metrics. It provides a good measure of detection capability.

Analyzing the latency time tradeoff with accuracy and other metrics allows us to determine the models most suited to real-time, adaptive intrusion detection requirements in smart home IoT environments.

VI. Results and Discussion

The results presented for the various machine learning models indicate a diverse range of performance outcomes, with particular interest in the balance between latency and detection metrics such as accuracy and precision. Since we have two datasets (unbalanced and balanced), the figures show the precision metric related to latency times for the unbalanced dataset since the accuracy can be misleading for unbalanced datasets. This is in contrast to the balanced dataset, where accuracy is appropriate. We generated 8*3*2 plots for the 12 models since we measured 8 metrics (training time, prediction time, latency time, memory usage, accuracy, precision, recall, and F1 score) for 3 datasets (low rate, high rate, and low + high rate) sampled from 2 datasets (unbalanced and balanced). Hereafter, we show a subset of the plots most related to the IDS architecture design requirement: real-time, accuracy/precision, and adaptability.

A. *Devices with Low-rate Traffic*

For the unbalanced CICIOT2023 dataset, Fig. 1 and Fig. 2 show that the Extreme Learning Machine algorithms demonstrate fast training and prediction latency times, meeting real-time intrusion detection needs. Specifically, the Regularized ELM provides the best balance with strong precision and total latency time. Decision Tree has low latency, making it a good candidate for low-rate devices, as intuitively predicted. In contrast, ensemble methods can enhance precision but have high training times.

For the balanced dataset, the Regularized ELM has the shortest training time at 1.0281 seconds, contributing to a low overall latency time of 1.3915 seconds when combined with its prediction time. This suggests that Regularized ELM is highly suitable for real-time applications where quick model training and prediction are critical. However, the regulated ELM has high precision and recall while maintaining 99.7% accuracy, slightly lower than other models. This combination of speed and reliability makes it the top choice if minimizing detection delay is critical. On the other end of the spectrum, ensemble methods like Random Forest, AdaBoost, XGBoost, and LightGBM have significantly longer training times, contributing to their higher overall latency times making them less suitable for real-time intrusion detection.

Furthermore, the memory usage across all ELM models is relatively smaller than the other models. An interesting outlier is the Decision Tree model, which has low latency similar to ELM methods. The models are fairly consistent regarding memory usage, ranging from 1-1.1 GB, which is acceptable for Edge-computing intrusion detection architecture.

B. *Devices with High-rate Traffic*

Fig. 3 and Fig. 4 show that the ELM variants (OS-ELM, Kernel ELM, Regularized ELM, Weighted ELM) have very low latency times, under two seconds. This makes them well-suited for real-time intrusion detection, where getting alerts quickly is critical. However, their detection accuracy is slightly lower than that of ensemble methods, which are slightly accurate but come at the cost of higher latency times. This suggests that Regularized ELM is particularly well-suited for environments where rapid detection is critical and resources may be limited, which fits our case. In contrast, tree ensemble methods show higher latency times, with AdaBoost reaching up to 15.9998 seconds. However, these models exhibit excellent detection performance. The tradeoff is between the higher computational and time costs against the benefit of potentially more accurate detection.

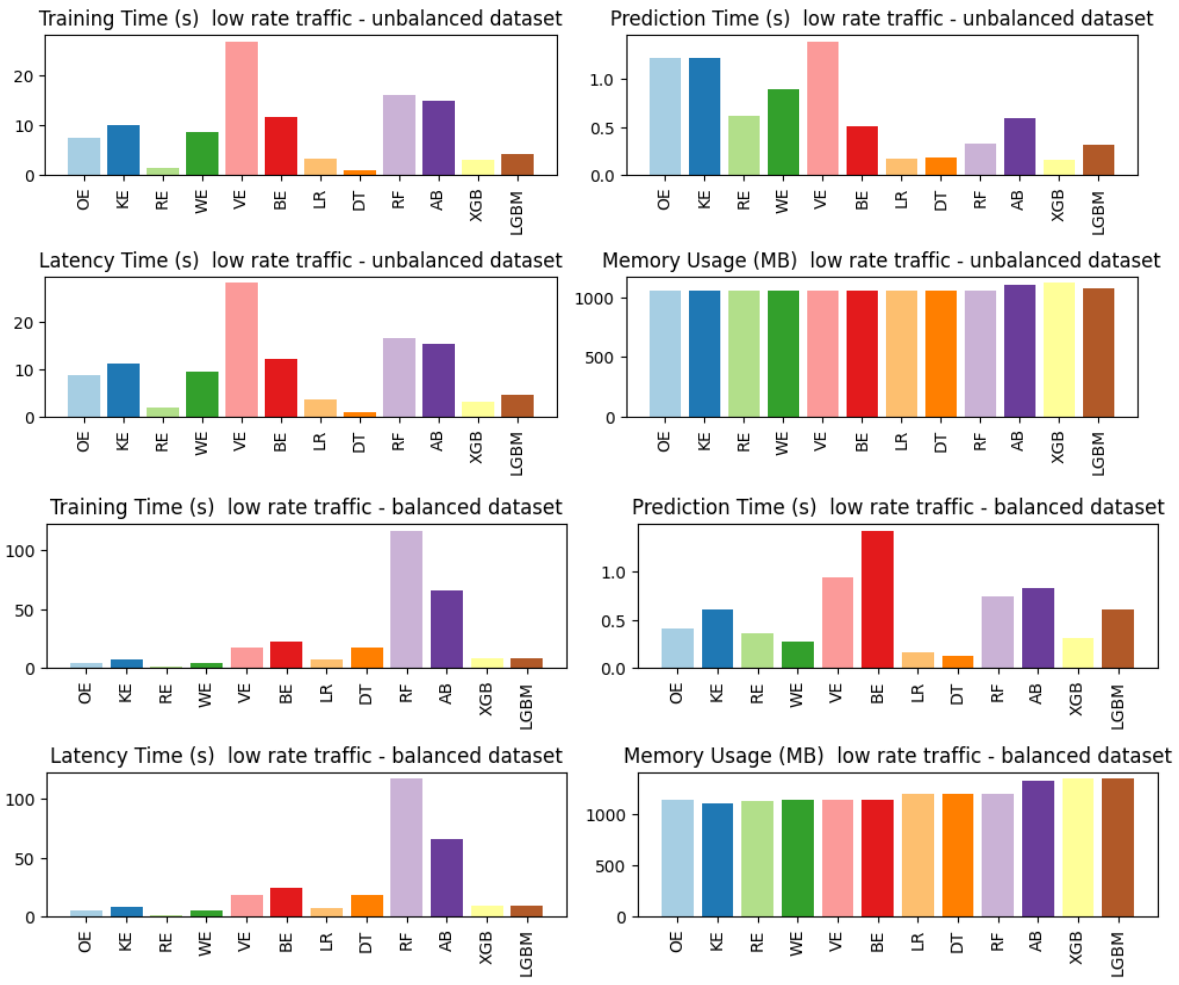

Fig. 1. Low-rate device training, prediction, latency times, and memory usage for unbalanced and balanced datasets.

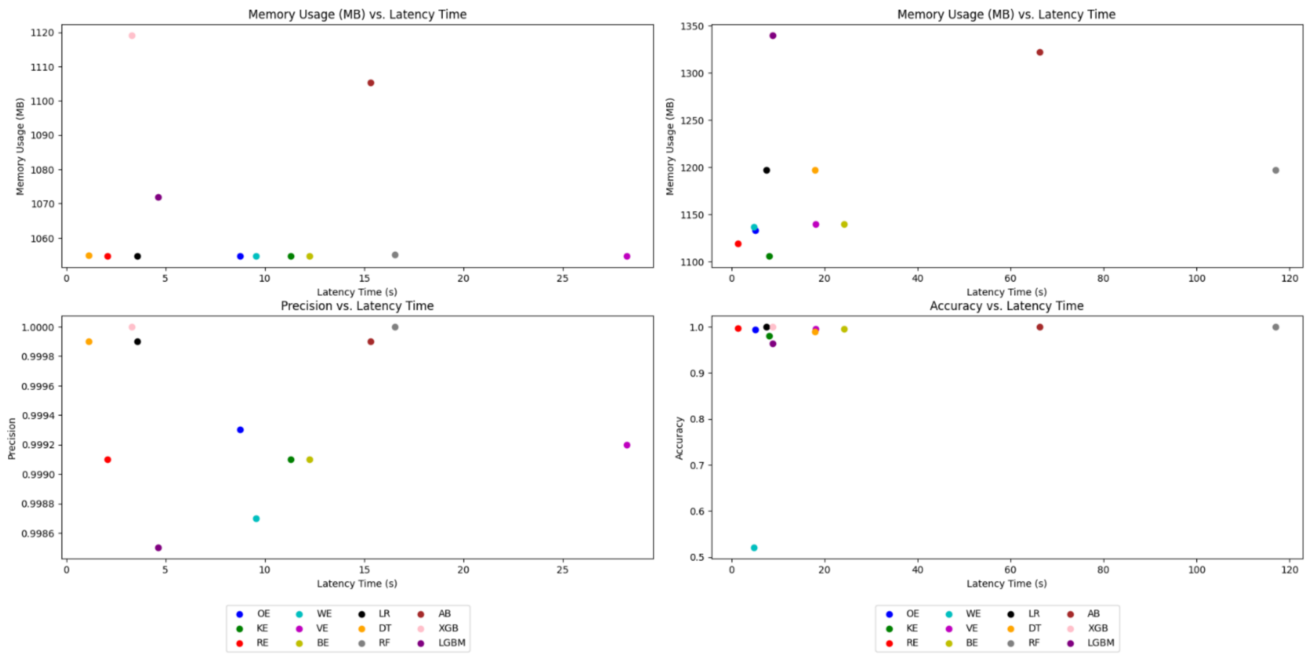

Fig. 2. Latency time according to precision (resp. accuracy) for unbalanced (resp balanced) datasets.

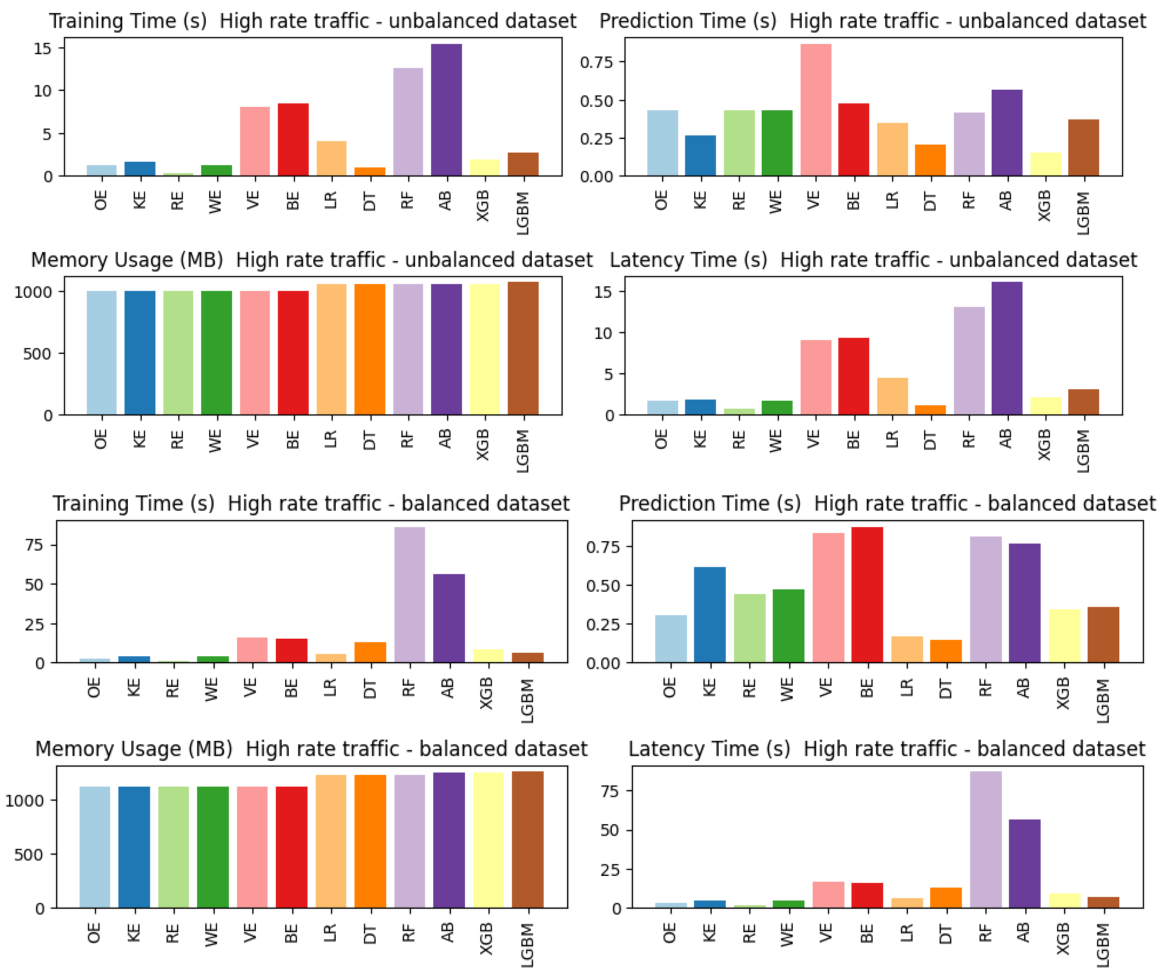

Fig. 3. High-rate device training, prediction, latency times, and memory usage for unbalanced and balanced datasets.

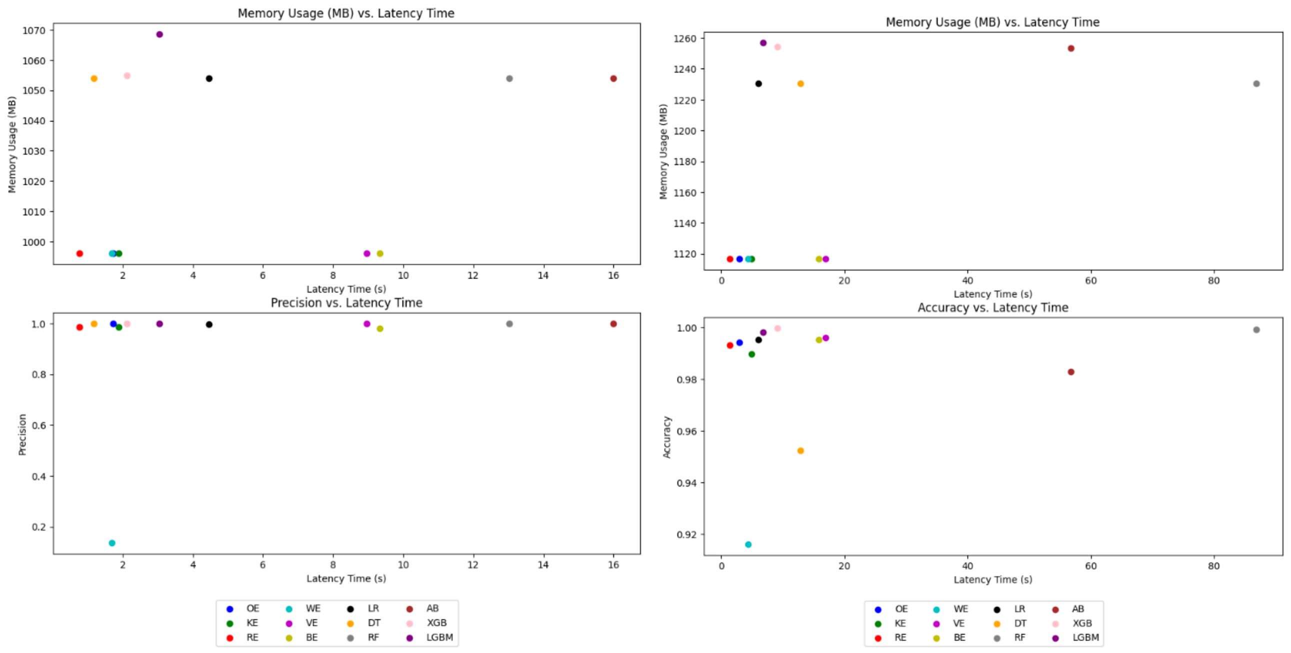

Fig. 4. Latency time according to precision (resp. accuracy) for unbalanced (resp balanced) datasets.

The ELM variants have the lowest training and prediction times for the balanced dataset. This enables very fast intrusion detection, suitable for real-time intrusion alerting. Regularized ELM stands out with high accuracy, precision, recall, and F1 scores at 99% despite having the lowest latency of just 1.44 seconds. This demonstrates it can deliver both speed and accuracy. The ensemble methods achieve nearly perfect evaluation metrics but with prohibitive training times. Their high latency makes them impractical for time-critical detection. Memory usage hovers between 1.1-1.3GB for most models with the lowest value of ELM variants. This confirms the adequacy of complex models for the high-rate devices reflecting the complexity of the traffic pattern.

C. Devices with Low+high Rate Traffic

Fig. 5 and Fig. 6 show that the Regularized ELM offers an impressive balance between speed and performance, with the lowest training time of 0.9867 seconds and a very competitive prediction time of 0.3748 seconds, resulting in a total latency of 1.3615 seconds. This is complemented by high accuracy (0.9986), precision (0.9989), and an F1 score (0.9993), making it a strong candidate for real-time applications where both speed and accuracy are critical. In contrast, ensemble methods exhibit higher latency times, ranging from 7.9752 to 31.2592 seconds, but with high accuracy and precision.

The memory usage metric indicates that all models are relatively memory-intensive, with usage ranging from 1601.1875 to 1720.7383 MB. This is a limiting factor in resource-constrained environments, such as our edge computing, and confirms our architecture's suitability, which separates the traffic into two groups to master the time and memory usage consumption.

The training times align with overall latency, with the ELM models being the fastest to train while the ensemble methods are slower. Prediction times are consistently low across the models. Memory usage hovers between 1.6-1.7GB for most models, up to 1.72 GB for LightGBM. So, memory is unlikely to be a constraint.

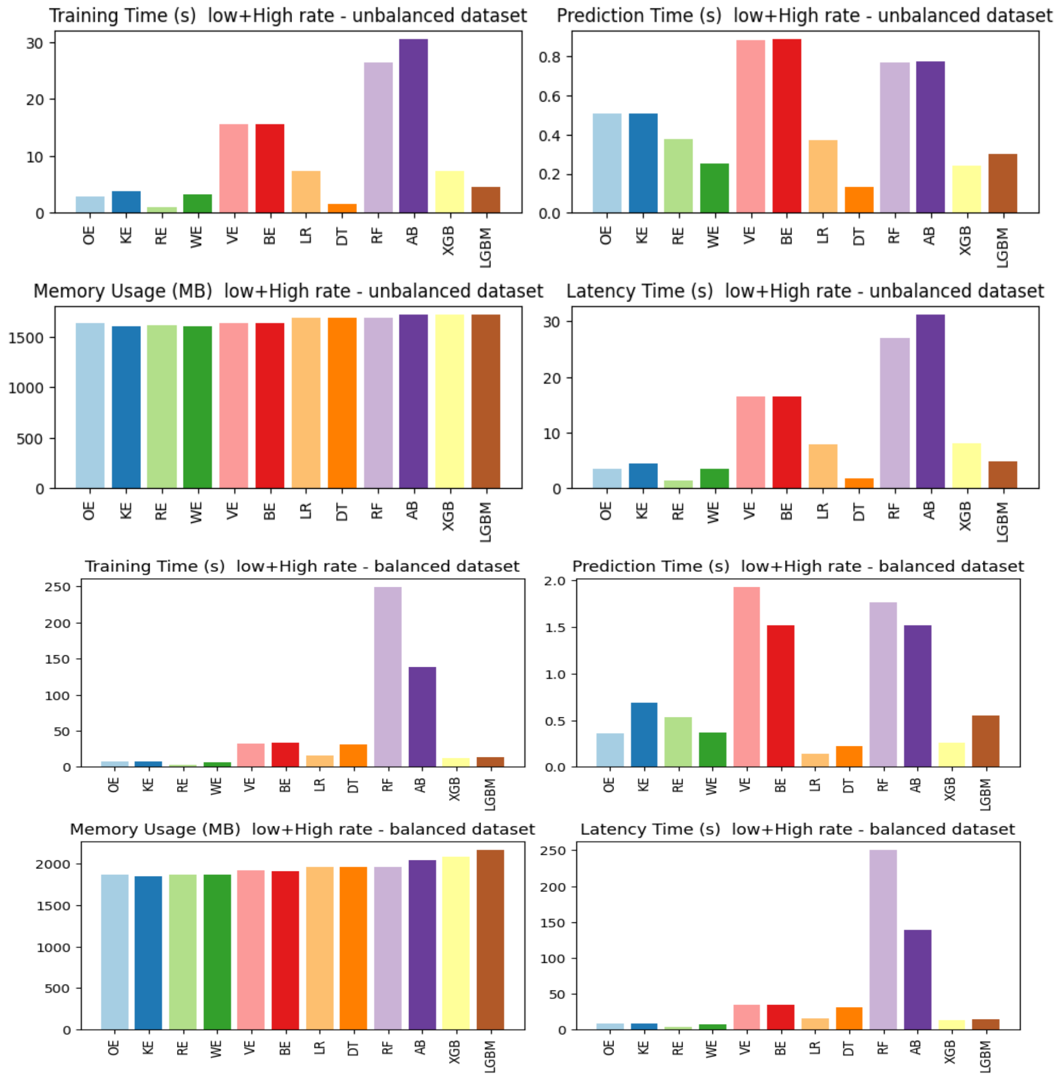

Fig. 5. High-rate device training, prediction, latency times, and memory usage for unbalanced and balanced datasets.

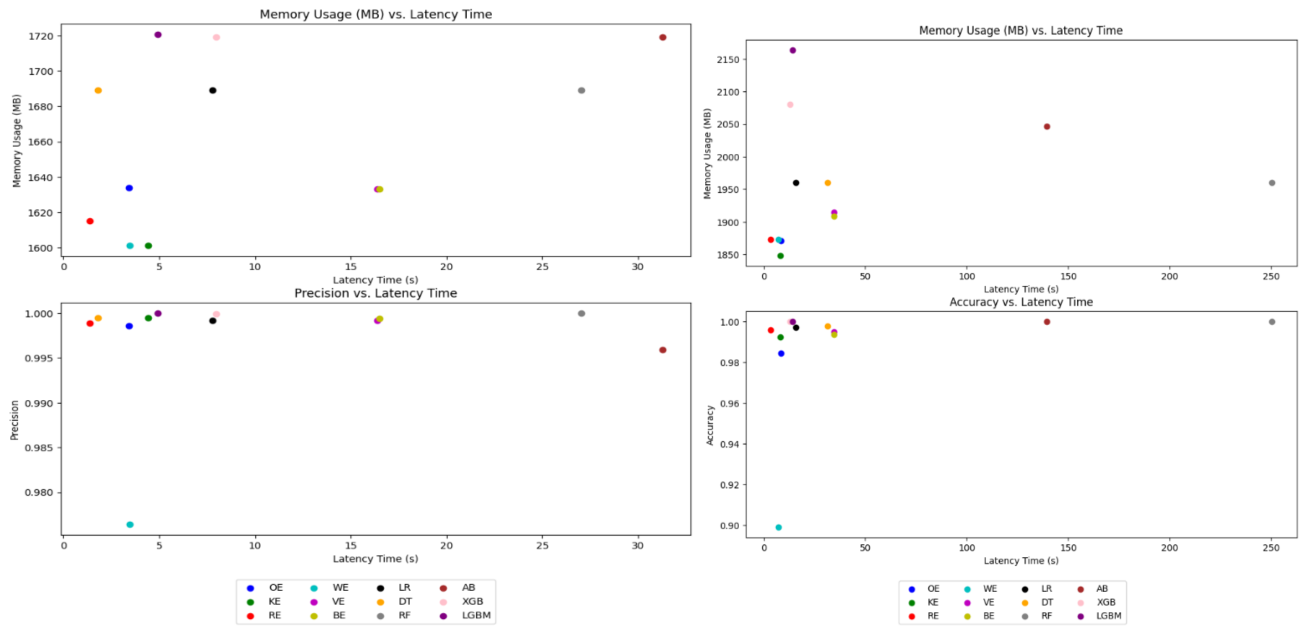

Fig. 6. Latency time according to precision (resp. accuracy) for unbalanced (resp balanced) datasets.

For the balanced dataset, the ELM variants (OS-ELM, Kernel ELM, Regularized ELM) deliver the fastest performance with training and prediction times under 1 second, maintaining near-perfect accuracy. The low latency makes them optimal for real-time usage. Conversely, ensemble methods like Random Forest, AdaBoost, XGBoost, and LightGBM achieve near-perfect scores on all metrics but at the cost of very high training times, from 137 to over 248 seconds. Their latency is prohibitive for time-critical detection.

Memory usage is reasonably consistent for most models between 1.8 GB and 2GB, which could be limiting for memory-constrained environments. The extreme learner methods have lower memory requirements that may suit resource-limited IoT devices better.

VII. Conclusion and Future Work

In this paper, we built a simple taxonomy to separate devices into two broad categories: one with high-volume and complex patterns and low-rate and simple traffic patterns. Based on this categorization, we design a suitable NIDS architecture and select machine learning models that are most adequate for each device type and traffic profile. The models are chosen based on detection accuracy, computational efficiency, and ability to handle complex traffic patterns. For that, we leverage the new CICIoT2023 dataset containing up to date IoT network traffic data with different realistic attacks. Using this dataset, we evaluate various machine learning models to develop an IDS focused on real-time, adaptive detection of intrusions specific to IoT devices.

We optimized the intrusion detection capabilities across smart home IoT deployments by matching the right models to the specific use case requirements around timing, accuracy, and resource constraints. The benchmark analysis guides on selecting between fast ELM variants versus slower but more precise ensemble methods. For low throughput IoT devices with minimal, regular traffic patterns, simpler models like decision trees provide efficient and fast anomaly detection. Their basic architectures allow quick training and scoring to enable real-time intrusion alerting. In contrast, high throughput multimedia devices require more advanced models like Regularized ELM to capture complex and evolving traffic patterns while maintaining low latency. The nonlinear mappings and optimized complexity in Regularized ELM balance speed and accuracy. So, the device traffic profiles and characteristics directly inform the machine learning model selection to optimize detection capabilities. The benchmark analysis maps models to the specific performance needs driven by the IoT taxonomy of low and high throughput groups. This specialized, profile-based model assignment enhances both efficiency and security.

As a future work, we intend to deploy and evaluate the system in a real-world smart home environment at scale to assess performance with live traffic and attacks. We will also enhance the capability of the system to become intrusion detection and prevention system by correlating intrusion alerts with device vulnerabilities and risk profiles to harden IoT device configurations through SDN dynamically.

Acknowledgment

The author would like to thank the Deanship of Scientific Research at Umm Al-Qura University for supporting this work by Grant Code: 24UQU4350605DSR02.

References

[1] J., Asharf, N., et all. A review of intrusion detection systems using machine and deep learning in Internet of things: Challenges, solutions and future directions. Electronics, vol. 9 no. 7, pp. 11-77, 2020.

[2] N., Chaabouni, M., Mosbah, A., Zemmari, C., Sauvignac, and P. Faruki, Network intrusion detection for IoT security based on learning techniques. IEEE Communications Surveys & Tutorials, vol. 21. no. 3, pp. 2671-2701. 2019.

[3] G. Altan, SecureDeepNet-IoT: A deep learning application for invasion detection in industrial Internet of things sensing systems. Transactions

on Emerging Telecommunications Technologies, vol.32, no 4, pp. 42-28.2021.

[4] H. Sallay, An integrated multilayered framework for IoT security intrusion decisions. Intelligent Automation & Soft Computing, vol 36. no 1.pp. 429-444.2023.

[5] Li, Y., Qiu, R., & Jing, S. Intrusion detection system using Online Sequence Extreme Learning Machine (OS-ELM) in advanced metering infrastructure of smart grid. PloS one, vol. 13. no 2. 2018.

[6] X., An, X., Zhou, X., Lü, F., Lin, and L. Yang. Sample selected extreme learning machine based intrusion detection in fog computing and MEC. Wireless Communications and Mobile Computing, pp.1-10.2018.

[7] R., Heartfield, G., Loukas, A., Bezemskij, and E. Panaousis, Self-configurable cyber-physical intrusion detection for smart homes using reinforcement learning. IEEE Transactions on Information Forensics and Security, vol. 16, pp. 1720-1735. 2020.

[8] E. D. Alalade. Intrusion detection system in smart home network using artificial immune system and extreme learning machine hybrid approach. In 2020 IEEE 6th World Forum on Internet of Things (WF-IoT). pp. 1-2. 2020.

[9] M., Noman S., Rosli A.H., Mohammad and, Z. Muhammad. SDN based intrusion detection and prevention systems using manufacturer usage description: a survey. (IJACSA) International Journal of Advanced Computer Science and Applications, vol 11, no 12, 2020.

[10] A. S., Ibrahim, K. Y., Youssef, H., Kamel, and M. Abouelatta. Traffic modelling of smart city internet of things architecture. IET Communications, vol 4, no 8, pp.1275-1284.2020.

[11] S., Kumar, et all. Characterizing IoT traffic in smart home and campus environments. Proceedings IEEE INFOCOM pp.2706-2715.2020.

[12] N., Feamster, J., Rexford, and E. Zegura, The road to SDN: an intellectual history of programmable networks. ACM SIGCOMM Computer Communication Review, vol. 44, no. 2, pp. 87-98, 2014.

[13] K., Benzekki, A., El Fergougui, and A. Elbelrhiti Elalaoui. Software defined networking (SDN): a survey. Security and communication networks, vol. 9, no. 18, pp. 5803-5833.2016.

[14] L., Mamushiane, A., Lysko, and S. Dlamini,. A comparative evaluation of the performance of popular SDN controllers. In 2018 Wireless Days (WD) (pp. 54-59). 2018.

[15] F., Liu, et all. A survey on edge computing systems and tools. Proceedings of the IEEE, vol. 107, no. 8, pp.1537-1562.2019.

[16] S., Hamdan, M., Ayyash, and S. Almajali, Edge-computing architectures for Internet of things applications: A survey. Sensors, vol. 20, no. 22, 6441.2020.

[17] D., Dholakiya, T., Kshirsagar, and A. Nayak, Survey of mininet challenges, opportunities, and application in software-defined network (sdn). Information and Communication Technology for Intelligent Systems: Proceedings of ICTIS 2020, vol 2, pp. 213-221.2021.

[18] E.,Neto, et all. CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment. Sensors, vol. 23, no. 13, 5941, 2023.

[19] G. B., Huang, Q. Y., Zhu, and C. K. Siew, Extreme learning machine: theory and applications. Neurocomputing, vol. 70, no 1-3, pp. 489-501.2006.

[20] G. B., Huang, D. H., Wang, and Y. Lan, Extreme learning machines: a survey. International journal of machine learning and cybernetics, vol. 2, pp. 107-122. 2011.

[21] W., Deng, Q., Zheng, and L.Chen, Regularized extreme learning machine. In 2009 IEEE symposium on computational intelligence and data mining. pp. 389-395. 2009.

[22] G. B., Huang, S., Song, and K. You, Trends in extreme learning machines: A review. Neural Networks, vol. 61, pp. 32-48.2015.

[23] O. A., Alade, A., Selamat, and R.Sallehuddin. A review of advances in extreme learning machine techniques and its applications. In Recent Trends in Information and Communication Technology: Proceedings of the 2nd International Conference of Reliable Information and Communication Technology. pp. 885-895. 2018.

[24] I. D., Mienye, Y.Sun. A survey of ensemble learning: Concepts, algorithms, applications, and prospects. IEEE Access, vol. 10, pp. 129-149. 2022.

[25] D. Kumar, Priyanka. Decision tree classifier: a detailed survey. International Journal of Information and Decision Sciences, vol. 12, no. 3, pp. 246-269.2020.

[26] P. A. A., Resende, A. C. Drummond . A survey of random forest based methods for intrusion detection systems. ACM Computing Surveys (CSUR), vol. 51, no. 3, pp. 1-36.2018.

[27] G., Ke, et all. Lightgbm: A highly efficient gradient boosting decision tree. Advances in neural information processing systems, vol. 30.2017.