Alwaisi, Zainab; Kumar, Tanesh; Harjula, Erkki; Soderi, Simone

Securing constrained IoT systems: A lightweight machine learning approach for anomaly detection and prevention

Published in:
Internet of Things (The Netherlands)

DOI:
10.1016/j.iot.2024.101398

Published: 01/12/2024

Document Version
Publisher's PDF, also known as Version of record

Published under the following license:
CC BY

Please cite the original version:
Alwaisi, Z., Kumar, T., Harjula, E., & Soderi, S. (2024). Securing constrained IoT systems: A lightweight machine learning approach for anomaly detection and prevention. *Internet of Things (The Netherlands)*, *28*, Article 101398. https://doi.org/10.1016/j.iot.2024.101398

Internet of Things 28 (2024) 101398

Contents lists available at ScienceDirect

Internet of Things

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

Research article

Securing constrained IoT systems: A lightweight machine learning approach for anomaly detection and prevention

Zainab Alwaisi [a,*], Tanesh Kumar [c], Erkki Harjula [b], Simone Soderi [a]

[a] *IMT School For Advanced Studies, Italy*
[b] *Centre for Wireless Communications, University of Oulu, Finland*
[c] *School of Electrical Engineering, Aalto University, Finland*

ARTICLE INFO

Keywords:
Internet of Things (IoT)
Smart devices
Energy
Memory
Resource constraints
Detection
TinyML
ML
Edge AI

ABSTRACT

With the advent of advanced technological developments such as IoT, edge, and fog computing, cyber attacks have become increasingly sophisticated. IoT networks facilitate collaborative and intelligent tasks across various domains, including Industry 4.0, digital healthcare, and home automation. However, the proliferation of IoT devices has raised concerns about severe attacks, particularly those targeting resource constraints such as energy and memory. In response to these challenges, Tiny Machine Learning (TinyML) has emerged as a new research area, focusing on machine learning techniques tailored for embedded and IoT systems. This study proposes an ML detection mechanism designed to categorize and detect resource-constrained attacks in IoT devices. We consider IoT devices to be integral components within the continuum of edge and cloud computing, leveraging EdgeML and CloudML for detection purposes. Our paper conducts a comparative analysis of ML models, with a specific focus on energy consumption and memory usage in IoT applications. We compare various ML methodologies, including cloud-based, edge-based, and device-based strategies for both training and detection. The evaluation encompasses the application of these ML techniques to petite IoT devices, utilizing TinyML, as well as cloud and edge devices. Our findings reveal that the Decision Tree algorithm deployed on smart devices surpasses other approaches in terms of training efficiency, resource utilization, and the ability to detect resource-constrained attacks on IoT devices. We demonstrate a high level of accuracy, exceeding 96.9%, across all presented ML models in detecting resource constraint attacks within IoT systems. In summary, this research serves as a guide for implementing effective security measures in the dynamic landscape of IoT and associated technologies.

1. Introduction

The Internet of Things (IoT) encompasses a vast array of interconnected devices ranging from everyday items like toasters and toothbrushes to sophisticated machinery like jet engines, all operating on seamless connectivity and communication between devices. However, IoT sensors often face constraints such as limited memory, power, battery capacity, and network capabilities, necessitating the efficient computation, storage, access, and analysis of IoT data. As data volumes continue to grow alongside increasing heterogeneity in both data and objects, effectively managing and handling these challenges becomes imperative [1–4].

The demand for energy-efficient IoT applications is rising, necessitating low latency, higher reliability, and robust security measures, particularly in applications such as smart cities, digital healthcare, and Industry 4.0 [5,6]. Edge and fog computing,

* Corresponding author.
E-mail address: zainabalwaise@yahoo.com (Z. Alwaisi).

https://doi.org/10.1016/j.iot.2024.101398
Received 2 May 2024; Received in revised form 20 September 2024; Accepted 8 October 2024

Available online 21 October 2024

integrated with traditional IoT network architecture, offer significant benefits by bringing computations and resources closer to devices/data sources. Moreover, the incorporation of AI and other advanced enabling technologies facilitates automated and faster data processing, analysis, and decision-making. However, ensuring authentication, security, and privacy in the face of vast data generation from IoT sensors remains crucial. Security risks, such as Distributed Denial of Service (DDoS) attacks and eavesdropping concerns in wireless IoT communications, highlight the need for robust security measures, including Intrusion Detection Systems (IDS) [7–9].

Despite security measures in LANs, RFID, ad hoc networks, and WLANs, IoT gateways often lack robust security, rendering them vulnerable to various threats. Default credentials and delayed updates exacerbate these security risks, particularly for resource-constrained IoT devices. To address these challenges, a novel paradigm known as Tiny Machine Learning (TinyML) has emerged, bridging the gap between machine learning (ML), deep learning, and edge devices. TinyML enables the deployment of small ML models on tiny IoT devices or gateways with strict resource constraints, allowing data analysis and interpretation to be performed locally, facilitating real-time actions [10].

Our paper proposes a novel ML detection mechanism designed to categorize and detect resource-constrained attacks in IoT devices. We conduct a comparative analysis of ML models, focusing on energy consumption and memory usage for IoT applications. Specifically, we investigate the feasibility and effectiveness of deploying pre-trained ML models on tiny IoT devices through techniques such as quantization and pruning, as referenced in Wardana et al.'s work [11]. By emphasizing the practical implementation of ML techniques on resource-constrained IoT devices, our work aims to contribute to the security enhancement of constrained IoT systems, particularly in the context of TinyML techniques. Additionally, we recognize the potential of more lightweight algorithms specifically designed for resource-constrained IoT environments, and exploring these in the context of security remains a future direction of this work.

1.1. Motivation

The motivation behind our proposed ML detection mechanism stems from the urgent need to address the evolving cybersecurity threats in IoT environments. Despite the widespread adoption of IoT devices, security vulnerabilities persist, leaving these interconnected systems susceptible to various attacks. Traditional security approaches often fall short in mitigating these threats, particularly concerning resource-constrained IoT devices.

Numerous scholars have emphasized the vulnerability of IoT ecosystems to attacks targeting resource limitations such as energy and memory capacity. IDS have been proposed as a countermeasure, operating at the IoT gateway level to detect and mitigate potential threats. However, the effectiveness of such systems is limited, especially in scenarios where IoT devices operate with minimal resources. Furthermore, the dynamic nature of IoT environments, characterized by heterogeneous devices and constantly evolving attack vectors, necessitates adaptive defence mechanisms. This highlights the critical need for innovative approaches that can detect and respond to resource-constrained attacks in real time, ensuring the integrity and security of IoT systems. Our research addresses this need by introducing an efficient and host-centric approach for detecting and defending against anomalies resulting from resource constraints in IoT applications. Leveraging TinyML and ML techniques tailored for IoT systems, our proposed mechanism can analyse real-time data to identify resource-constrained attacks promptly. Our technique can differentiate between normal operating conditions and potential attacks by analysing real data, allowing for timely response and mitigation strategies.

1.2. Our contributions

This study introduces a novel host-centric approach for detecting and defending against anomalies resulting from resource constraints in IoT applications. Leveraging TinyML and ML techniques, our approach is specifically tailored to address the challenges posed by resource-constrained IoT environments.

A significant contribution of our work is the comprehensive validation dataset derived from our previous research efforts [12–16]. This dataset includes measurements of resource utilization across various levels of the IoT infrastructure, such as individual devices, edge computing nodes, and cloud servers. We enhance this dataset with empirical data collected from diverse real-world scenarios, considering factors like varying system loads, device activities, and times of day (e.g., peak and off-peak periods). This approach ensures a thorough evaluation of our proposed mechanism and accounts for the complexity and variability of real-world conditions. We augment this dataset with additional malicious attack data obtained from external sources [17–19], enabling us to simulate a wide range of attack scenarios and assess the robustness of our defence mechanism. Leveraging this hybrid dataset, we develop an automated defence mechanism capable of filtering normal and abnormal behaviours related to energy and memory usage across all levels of the IoT infrastructure.

Our paper focuses on harnessing TinyML and ML techniques to detect resource-constrained attacks across various components of the IoT ecosystem. We categorize resource-constrained attacks into distinct types, such as energy and memory attacks, to comprehensively understand the threat landscape. Furthermore, our proposed mechanism offers real-time detection capabilities for future resource-constrained attacks. By learning from the current situation, our system can analyse patterns and trends to anticipate and mitigate potential attacks before they occur. This proactive approach enhances the resilience of IoT systems against emerging threats and ensures continuous protection of critical infrastructure. The evaluation of resource-constrained attack detection spans different computational platforms, including smart devices, edge nodes, and cloud servers. We compare various ML training models and assess their accuracy in detecting attacks across all levels of the IoT infrastructure. Notably, our results demonstrate the effectiveness of our approach in accurately identifying and mitigating resource-constrained attacks, thereby enhancing the security posture of IoT systems.

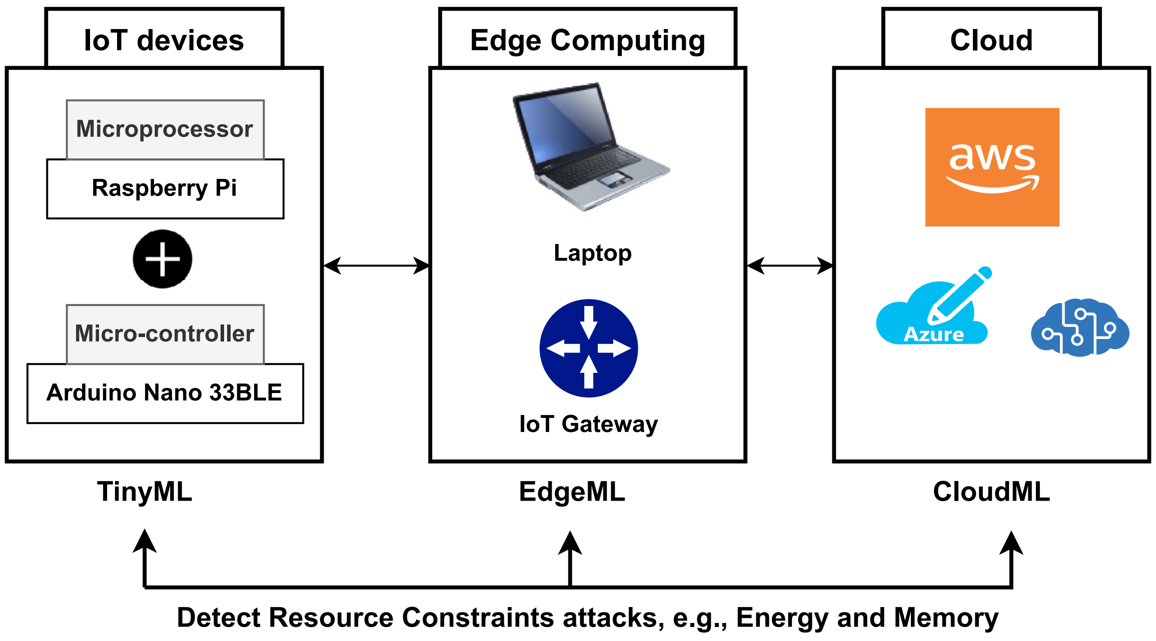


Fig. 1. Architecture of IoT System and ML deployment models.

1.3. Organization of the paper

This paper is organized as follows. Section 2 provides background reading and related work. We discuss the system architecture, testbed, attack scenarios, and threat model in Sections Section 3. Our proposal, including metrics definition, methodology, detection algorithm, and dataset selection and accuracy, is described in Section 4. Sections 6 and 7 present the results and evaluations. Section 8 offers the results discussion and a comparison with existing studies. Finally, Section 9 concludes with some final remarks and suggestions for future work.

2. Related work and background reading

IDS play an active role in analysing network and system activity to identify and detect intrusions or malicious activities, triggering alarms when necessary. IDS systems provide crucial information about attackers, such as identity, location, intrusion type, and affected network layer [20,21]. TinyML brings ML to low-power microcontrollers [22]. As IoT devices handle advanced analysis tasks, challenges related to networking, security, and decision-making grow. Resource constraints like limited power, memory, and computing hinder reliable connections, security, and system fine-tuning. TinyML demands models capable of operating independently at edge nodes with low latency and resilience. To host such architectures, a three-tier edge-cloud architecture, including a local computational tier operating in, e.g., IoT clusters, has been introduced by Harjula et al. [23]. As IoT clusters rarely possess devices with the necessary stability and capacity to support a fully functional edge server, the authors proposed a decentralized three-tier Edge IoT architecture that leverages a serverless edge computing model. This approach allows the deployment of cloud functions in a distributed virtualized manner at the local tier. Addressing the challenges stemming from IoT technology limitations is imperative for ensuring the broad adoption and effectiveness of TinyML in diverse applications. Recent research efforts have focused on leveraging TinyML-based IDS to augment the security of IoT systems. Various researchers have explored TinyML for fault detection in smart devices within IoT systems. For instance, Krayden et al. [24] applied TinyML to enhance the performance of gas metal–oxide–semiconductor (GMOS) sensors, achieving flawless performance in detecting gas with efficient resource utilization. The field of intrusion detection in IoT systems has experienced a surge in interest in recent years. Chordia and Gupta [25] introduced an anomaly-based IDS focusing on identifying various types of attacks, while Thanigaivelan [26] presented an IDS framework where each IoT device autonomously surveils nearby devices for signs of abnormal activities and takes proactive measures when anomalies are detected. In the study by Tekin et al. [27], the authors investigate the application of TinyML on IoT devices and analyse the impact of ML algorithms on energy consumption. Conversely, our work focuses on utilizing ML techniques to detect attacks targeting energy and memory consumption in IoT systems. While Tekin et al. primarily address the energy consumption implications of ML deployment, our research extends this by specifically targeting the detection of resource constraint attacks that enhance security mechanisms in IoT environments. Other related works include studies by Sudharsan et al. [28] and Yılmaz et al. [29], which address botnet attacks detecting offline models for resource-constrained IoT devices, and a transfer learning approach for securing resource-constrained IoT devices, respectively. Therefore, our paper mainly focuses on detecting energy and memory attacks by utilizing TinyML techniques in the IoT system, e.g., IoT devices, cloud, and edge (see Fig. 1).

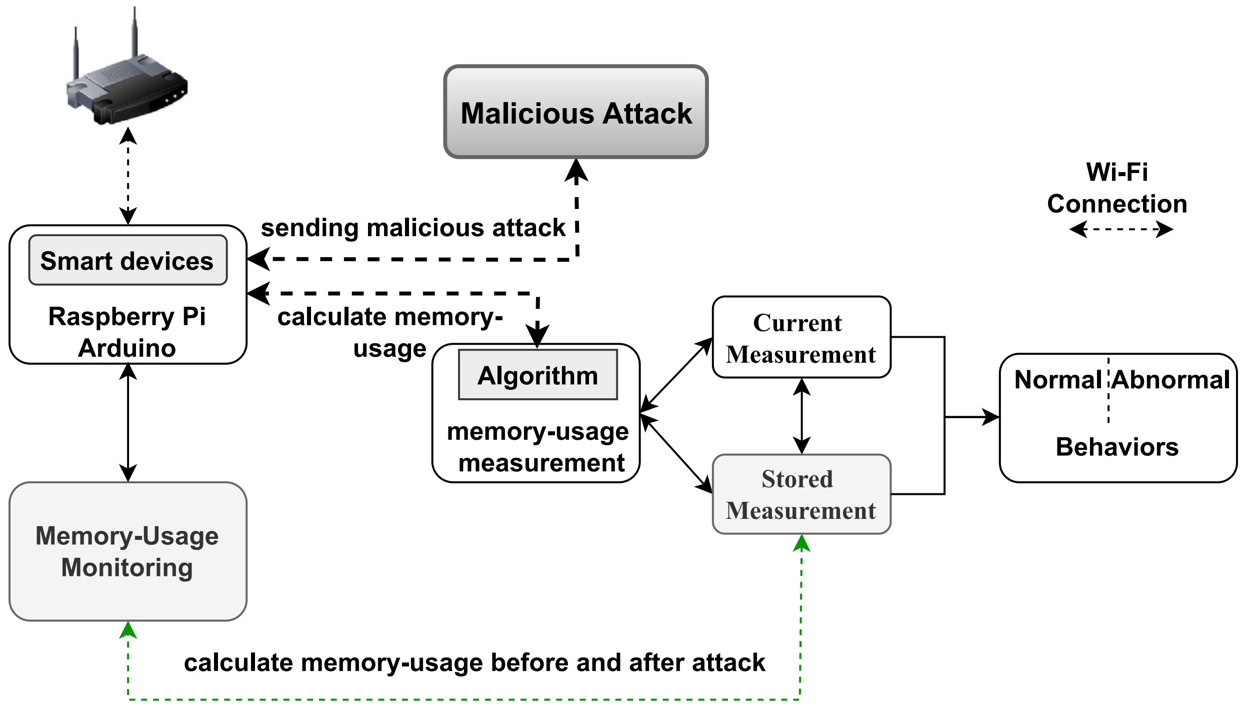


Fig. 2. Testing Environment.

Table 1
Devices used in this experiment.

Approach	Device	Hardware
Cloud	Amazon Cloud	Intel Xeon 16 core CPU 32 GB RAM
Edge	Asus Laptop	i7-10870H CPU @ 2.20 GHz 2.21 GHz 8 GB GPU
IoT	Raspberry Pi 4	Microprocessor IoT Device, 64-bit SoC 8 GB RAM
IoT	ArduinoNano 33BLE	Micro-controller IoT device, CPU 1 MB (nRF52840), SPRAM 256 KB (nRF52840)

3. System architecture

In this section, we delve into a comprehensive discussion of the proposed system architecture.

3.1. Architecture

The research employed a basic LAN as the foundational architecture, illustrated in Fig. 2. This architecture consisted of two IoT devices, with a laptop used as a malicious attacking node. All nodes were interconnected through a gateway router. This testbed assessed our proposed mechanism, as described in detail in the next section.

3.2. Testbed

Our research established a basic LAN architecture, depicted in Fig. 2. This architecture included two IoT devices, the Raspberry Pi and Arduino Nano 33 BLE, serving both as victim devices and a single computer acting as a malicious node. These devices were connected through a gateway router in a local network setup. Our testbed allowed us to evaluate our proposed mechanism. Additionally, our study extended to edge and cloud devices. For edge devices, a computer with an Intel (R) Core (TM) i7 CPU processor played the role. For cloud simulations, we utilized Amazon Cloud services. The key device specifications are summarized in Table 1.

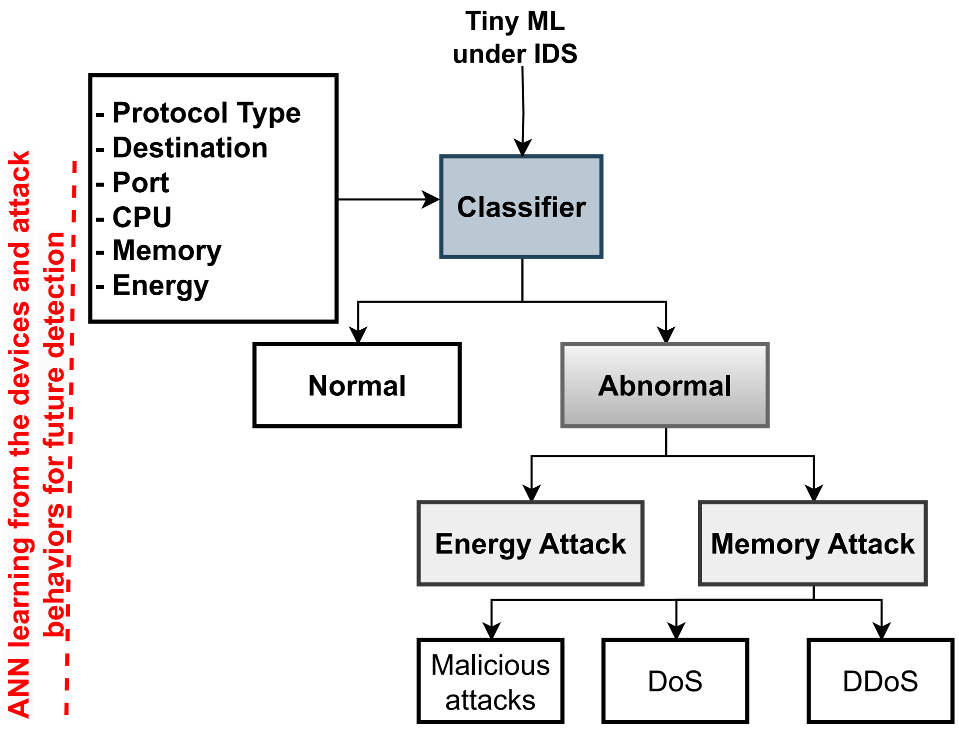


Fig. 3. TinyML Decision Model.

In this study, we consider IoT to encompass various smart devices that are constrained in terms of communication range, energy, memory, and processor capability. For simplicity, two smart devices were selected to represent the proposed work. We subjected these devices to malicious resource-constrained attacks to assess their impact on resource usage. The study also considers that an adversary may broadcast resource-constrained attacks to legitimate IoT, cloud, and edge devices. We measured the resource usage of these devices to distinguish between normal and abnormal behaviour. Additionally, a TinyML-IDS mechanism was developed to detect resource-constrained attacks, leveraging datasets specifically constructed for the smart, cloud, and edge devices used in the experiments.

3.3. Threat scenario and threat model

The threat scenario depicted in this study, as illustrated in Figs. 3 and 2, focuses on flooding attacks specifically targeting energy consumption and memory usage within the IoT system. These potent resource constraint attacks are modelled based on realistic behaviours observed in our test environment, where adversaries deploy various types of malicious activities such as flooding packets to excessively deplete the energy and memory capacity of smart devices, cloud resources, and network memory.

In our research scenario, it is confirmed that the adversary launches flooding attacks by inundating the IoT network with malicious packets from both outside and inside the network. This attack pattern has been validated through controlled experiments, where we observed significant impacts on the resources of smart devices and disruptions in communication between nodes, ultimately affecting the overall functionality and stability of the network. The intensity of these flooding attacks has been carefully calibrated and tested to reflect a realistic threat level that renders the targeted nodes incapable of further resource consumption, including energy and memory usage. This realistic modelling ensures that the scenario accurately represents the operational challenges posed by such attacks. Our assumptions in this scenario are grounded in known vulnerabilities of IoT devices, particularly in WiFi communication scenarios. While our study primarily focuses on WiFi-based attacks, this was intentional as we aimed to compare the effectiveness of various algorithms within this specific context. However, we acknowledge that IoT devices can also communicate via other protocols such as Ethernet, Bluetooth, Zigbee, LoRa, and Z-Wave. Extending our work to consider these alternative communication methods is a valuable direction for future research. To mitigate these threats, our study proposes the deployment of a TinyML and ML-based Intrusion Detection System (IDS). This system is designed to detect and identify both normal and abnormal behaviours related to resource-constraint usage and predict and prevent future flooding attacks. Specifically, the system leverages the sensing capabilities of the WiFi channel to collect readings, which are then evaluated against an onboard TinyML model. This model has been trained and validated to identify the specific types of resource constraints present within the Wireless Sensor Network (WSN), such as energy depletion or excessive memory usage, as caused by the observed flooding attacks.

4. Methodology

Before we delve into the evaluation results of our IoT IDS based on TinyML, which takes into account different smart device behaviours such as *Idle*, *Active*, and Under Attack, this section elucidates the primary methods and resources employed. This includes the selection of datasets, the distribution of data among various IoT devices, and an overview of the classifier technique and aggregation functions we are exploring. Our approach to detection involves monitoring specific attributes on end devices when exposed to resource-constrained attacks like DoS, DDoS, and malicious attacks. For IoT devices operating on the Linux OS, the kernel offers the following attributes: (i) CPU frequency and usage (%). (ii) CPU count, including logical CPUs. (iii) CPU statistics involving IRQs and syscalls. (iv) CPU times, showing time spent in different modes. (v) Memory Utilization, covering virtual/swap memory. (vi) Load Average, indicating processes in the system run queue. (vii) Network connections, reflecting system-wide sockets. (viii) Network Interface statistics, including state, bytes, and MTU.

For other IoT devices, such as microcontroller devices, we determine their memory usage by employing various libraries in both Python and C languages. Specifically, we utilize libraries such as *os*, *psutil*, and *memory-monitoring*, and for the *C* language, we used *MemoryFree* and *pgmStrToRAM*. The measurement of energy consumption is conducted using a dedicated smart circuit, as detailed in the subsequent sections.

4.1. Training techniques

ML workflow involves two main stages: training and inference. Training encompasses the creation of a model using ML algorithms and dedicated datasets, while inference pertains to the subsequent use of the pre-trained model for making predictions. This section delves into the specifics of the machine-learning algorithms employed in this study. The selection of these particular models is grounded in their attention to both memory and energy efficiency considerations.

1. Naive Bayes (NB): A widely used algorithm for classification based on Bayes' theorem [30]. It assumes conditional independence between features given the class label, making it efficient for high-dimensional datasets. We trained the model using maximum likelihood estimation, applying Laplace smoothing (parameter $\alpha = 1.0$) via the `GaussianNB` implementation in `sklearn` to address zero-frequency issues.
2. Decision Tree (DT): A common non-parametric supervised learning technique for both regression and classification [31]. We trained the DT using the CART method, setting a maximum tree depth of 10 to prevent overfitting. With Gini impurity as the criterion, the `sklearn` implementation was used. Hyperparameter tuning determined a minimum sample split of 2 and a minimum sample leaf of 1 for optimal performance.
3. Random Forest (RF): This ensemble method reduces overfitting by aggregating 100 decision trees, each trained on random feature subsets and bootstrapped datasets. We used the *RandomForestClassifier* from `sklearn`, setting the maximum depth of each tree to 15 and considering the square root of the total number of features (*max_features = sqrt(n_features)*) for splitting at each node to ensure model diversity and efficiency.
4. K-nearest neighbour (K-NN): this non-parametric method classifies input data by calculating distances using the Euclidean metric and identifying the k nearest neighbours. We set k to 5 after cross-validation to balance bias and variance. To address the high dimensionality of the dataset, Principal Component Analysis (PCA) was used for dimensionality reduction.

Therefore, the training phase of a TinyML-based IDS can be performed on IoT devices, edge devices, and the cloud.

1. IoT devices pose resource constraints for ML applications. Store-bought ones lack programmability and have limited computational capacity. As a result, running ML strains memory, energy, and CPU. Moreover, these devices provide users with two power source alternatives: wired or battery-powered. However, it is worth noting that battery-powered devices can deplete their resources rapidly during the ML model training process. Raspberry Pi suits ML with its computing power. However, lightweight ML is vital for devices like Arduino. In this experiment, we emphasize leveraging TinyML at the IoT device level.
2. Edge computing involves moving compute and storage closer to IoT devices [32]. It hosts services processes and stores data locally. Devices like Raspberry Pi or smartphones can serve as edge computing nodes with limited resources. Multiple IoT devices can access services from these distributed edge devices. Edge computing supports delay-sensitive applications and reduces network traffic for data-heavy tasks. It also enhances privacy and reduces energy costs compared to cloud-based ML, addressing latency and bandwidth challenges.
3. Cloud computing grants convenient online access to various computing resources, encompassing memory, servers, databases, and networks. This encompasses Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). SaaS facilitates software delivery via the Internet, PaaS delivers platform resources, and IaaS provides processing, storage, and network resources. Cloud computing effectively addresses the computational requirements of ML by offering storage and processing capabilities. Notable platforms for ML in the cloud encompass Google Cloud, Amazon Web Services (AWS), and Microsoft Azure.

4.2. TinyML

TinyML represents a promising technology championed by the research community for creating self-reliant and secure devices capable of collecting, analysing, and generating data without transmitting it to remote entities. In the upcoming section, we will delve deeper into the significance of incorporating TinyML within the proposed system [33]. The selection of this approach is crucial, as it enables the incorporation of ML models into compact and energy-efficient hardware. This, in turn, empowers real-time detection and processing of resource-constrained attacks, eliminating the need to transmit data to a remote cloud server for analysis. The system is intricately crafted to execute direct data collection, analysis, and extraction, ensuring improved privacy and security by keeping collected data confidential and preventing external sharing. Leveraging the TinyML framework, we can fine-tune and compress our machine-learning models, rendering them compatible with hardware featuring limited resources, such as MCUs. Furthermore, the efficiency and lightweight nature of TinyML, compared to traditional ML algorithms, strongly motivate its utilization for comprehensive resource-constrained attack detection at the smart device level.

4.3. Dataset selection

A crucial consideration in formulating our TinyML-based IDS for IoT devices involves choosing a suitable dataset.

In this study, we leverage datasets created from our previous laboratory experiments [12–14] to detect resource-constrained attacks caused by various types of malicious activities. The first dataset comprises attributes such as protocol type, destination port, IoT device type, memory usage, and attack type, totalling 60,000 samples. The second dataset includes protocol type, destination port, IoT device type, energy consumption, and attack type. These datasets are tailored to detect attacks like DoS, DDoS, malicious attacks, energy consumption-distributed Denial of Service (EC-DDoS), memory usage attacks, and flood attacks. Specifically, a DoS attack exhausts resources with continuous requests, while a DDoS disrupts smart devices by disconnecting them from the local access point. Malicious attacks occur when an external entity consumes smart device resources, and flooding attacks inundate IoT networks with malicious packets to increase energy and memory consumption.

The proportion of attack to normal samples in these datasets is designed to reflect realistic conditions. Specifically, our datasets are balanced to include a representative mix of normal and attack samples. For instance, in the first dataset, normal and attack samples are distributed approximately equally, with 50% of the samples labelled as normal and 50% as various types of attacks. This balance is crucial for training models that can generalize well and avoid bias towards the majority class.

Our measurement dataset comprises 200,000 samples, including both normal and attack scenarios, to evaluate the optimal detection mechanism for resource-constrained attacks. This dataset provides a comprehensive basis for training and validating our models, ensuring they can effectively distinguish between normal and abnormal behaviours in various contexts. We also quantify energy and memory usage at both the edge and cloud levels in the presence or absence of malicious attacks, using this dataset to specifically identify resource-constrained attacks induced by various attack types, as previously mentioned.

4.4. Prepossessing the dataset

The initial phase of preparing the dataset for training and deploying a model on a microcontroller through the Edge Impulse (EI) platform involves acquiring sensor data through a data capture process. EI offers various data collection methods, such as device connections and file uploads, to create a specific dataset tailored to the task at hand. This dataset is then utilized to train the ML model for the designated purpose, such as anomaly detection. We include a balanced representation of attack and normal samples to ensure the dataset is well-suited for training.

Once the data is captured, it undergoes a rigorous preprocessing phase to ensure suitability for training. This preprocessing includes normalizing and filtering the data to remove noise and standardize it, thereby maintaining consistency across the dataset. Flattened processing blocks are used to convert the multidimensional input data into a one-dimensional array, which is essential for reducing data complexity and ensuring effective processing by the TinyML model. Following preprocessing, the data is channelled into the anomaly detection learning block, where the ML model is trained. This phase involves selecting key parameters, such as the learning rate, batch size, and number of epochs, which are critical in determining the model's performance. For each type of attack proposed in the study, the model is trained using supervised learning techniques, where the algorithm learns to identify patterns associated with both normal and abnormal behaviours.

We conduct hyperparameter tuning during training to optimize the model's accuracy and detection capabilities. This process includes adjusting parameters like the decision threshold for anomaly detection and evaluating different model architectures to determine the most effective configuration. Cross-validation is also employed to ensure the model generalizes well to unseen data. Once the model is adequately trained and validated, it is deployed onto a microcontroller using the Edge Impulse platform. EI supports various microcontrollers, including the Arduino Nano 33 BLE Sense board, and provides tools for efficient model deployment. By implementing the approach outlined in the following sections, developers can establish robust, real-time solutions for detecting energy attacks and generating alerts across a spectrum of IoT applications.

5. Energy consumption and memory usage calculations

This section discusses various methods used to measure the resources of the smart devices and generate the final datasets. We employ different techniques to calculate the resource consumption of IoT devices under normal conditions and in the presence of resource-constraints attacks. The specific methods used for these purposes are detailed in the following sections:

5.1. Energy consumption measurement

In our prior research [13], we devised a sophisticated smart circuit equipped with non-invasive current sensors, resistors, and capacitors to accurately measure the power consumption of smart devices. This circuit continuously samples voltage, current, and power consumption, storing the data for each smart healthcare device in a dedicated database (DB). We utilize Joule (J) values to quantify the energy consumption of these devices. The energy consumption of IoT devices is meticulously evaluated both before and after cyberattacks. This evaluation forms the basis for generating our primary dataset, which is determined by the equation: $E(d) = f(e(d), n, ATK) \quad \text{and} \quad n \in [0, 1]$. Where e_d represents the energy measurement of device d at a specific moment, considering the presence or absence of cyberattacks (ATK), and n denotes the number of energy measurements within a given time interval. The function $f(e(d), n) \in [0, 1]$ ensures that 0 corresponds to the minimum energy consumption measurement, while 1 indicates the maximum energy consumption measurement in the absence or presence of an attack [15].

5.2. Memory usage measurement

In our previous investigations [12,15], we utilized various libraries within the microcontroller and microprocessor to assess the current memory usage of smart devices. Our Memory Usage Measurement (MUM) approach takes into account diverse device statuses in the presence or absence of attacks. Specifically, we define $MUM(d) = f(MUM(d), ATK, n) \quad \text{and} \quad n \in [0, 1]$, where ($MUM_d$) represents the memory usage measurement of device d at a specific moment, considering the absence or presence of cyberattacks for a given attack scenario (ATK). Where n denotes the number of memory usage measurements within a specified time interval.

The function $f(MUM(d), ATK, n) \in [0, 1]$ ensures that 0 corresponds to the minimum memory usage measurement, while 1 indicates the maximum memory usage measurement under the condition of the attack or its absence [15].

5.3. Normal usage of memory and energy consumption of IoT devices

To assess the trained models for typical energy and memory consumption behaviours, we computed the standard usage patterns for both memory and energy consumption of the smart devices under normal conditions, including Idle and Active behaviours, as illustrated in Figs. 4, 5, and 6. This involved measuring energy consumption and memory usage when no attacks were active on the smart devices.

Regarding normal energy consumption, we observed fluctuations for Raspberry Pi devices between 1.410 J and 1.425 J per second, with the minimum consumption detected at 1.410 J and the maximum at 1.425 J. For Arduino devices, the energy consumption varied between 1.060 J and 1.064 J, with the minimum at 1.060 J and the maximum at 1.064 J. However, the normal memory usage of Arduino devices ranged from 8% to 35%, with fluctuations between the minimum usage of 10% and the maximum of 30%. The Raspberry Pi device exhibited similar trends, with maximum normal usage at 10% and 35% as the maximum memory usage, as depicted in Figs. 4 and 6. The same equations were applied to calculate the memory usage of edge and cloud devices, with normal memory usage for edge devices below 70% and for cloud devices below 68%, as shown in Figs. 8 and 10.

$$E_{min} = P \times t \times (1 - \eta) \times ATK_F \times 3600 \tag{1}$$

$$E_{max} = P \times t \times ATK_F \times 3600 \tag{2}$$

In (1) we calculate the minimum energy consumption by subtracting the energy lost due to inefficiency from the total energy consumed. Whereas (2) represents the maximum energy consumption assuming no energy loss. Where E_{min} represents the minimum energy consumption, E_{max} represents the maximum energy consumption, P is the power (in watts) consumed by the device or system, t is the time (in hours) during which the device or system is in operation, and η is the efficiency of the device. The attack factor (ATK_F) represents the impact of the attack on the device's energy consumption.

Moreover, we conducted a comparative analysis of on-device ML deployment strategies, including cloud-based, edge-based, and IoT device-based solutions, focusing on their efficiency in detecting resource-constrained attacks, particularly in terms of training and detection times. Our investigation highlighted the remarkable efficiency of the NB algorithm, which not only demonstrated shorter training times compared to other ML algorithms but also exhibited a detection time of less than 40 seconds.

$$\overline{E} = \frac{E_d - E_{min}}{E_{max} - E_{min}} \tag{3}$$

To determine the Memory Usage (MU) of the smart devices, the following calculation is employed:

$$MU_{min,max} = \left(\frac{MA_{min,max}}{Total_{MU}} \right) \times 100, \tag{4}$$

Where MU_{min} represents the minimum memory usage in percentage, MU_{max} represents the maximum Memory Allocated (MA) in percentage, MA_{min} is the minimum amount of memory allocated, MA_{max} is the maximum amount of memory allocated, and $Total_{MU}$ is the total available memory.

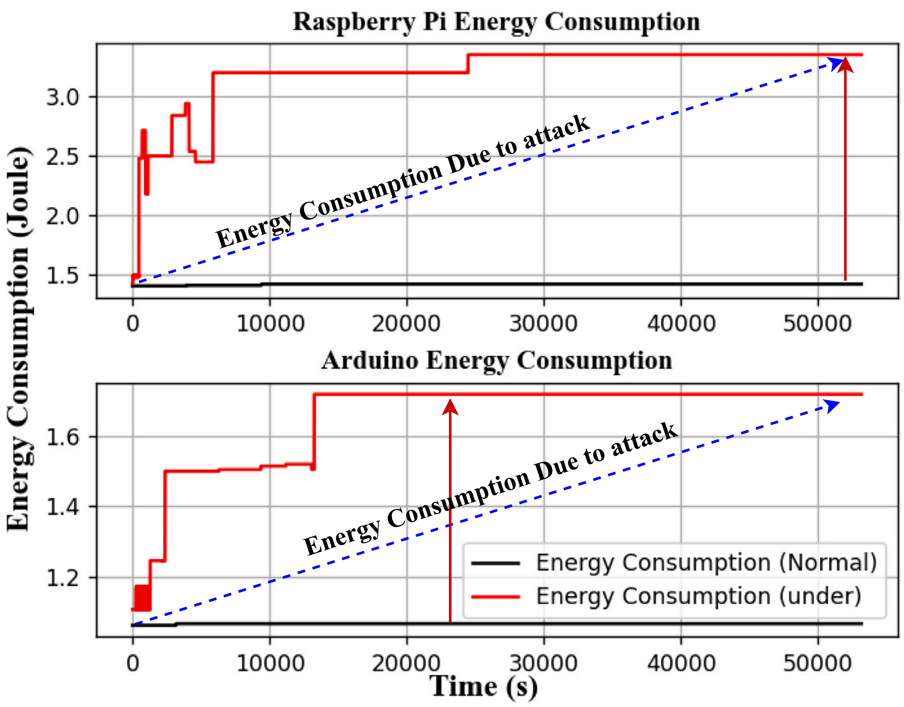


Fig. 4. Energy consumption for the Raspberry Pi and Arduino with or without the attack.

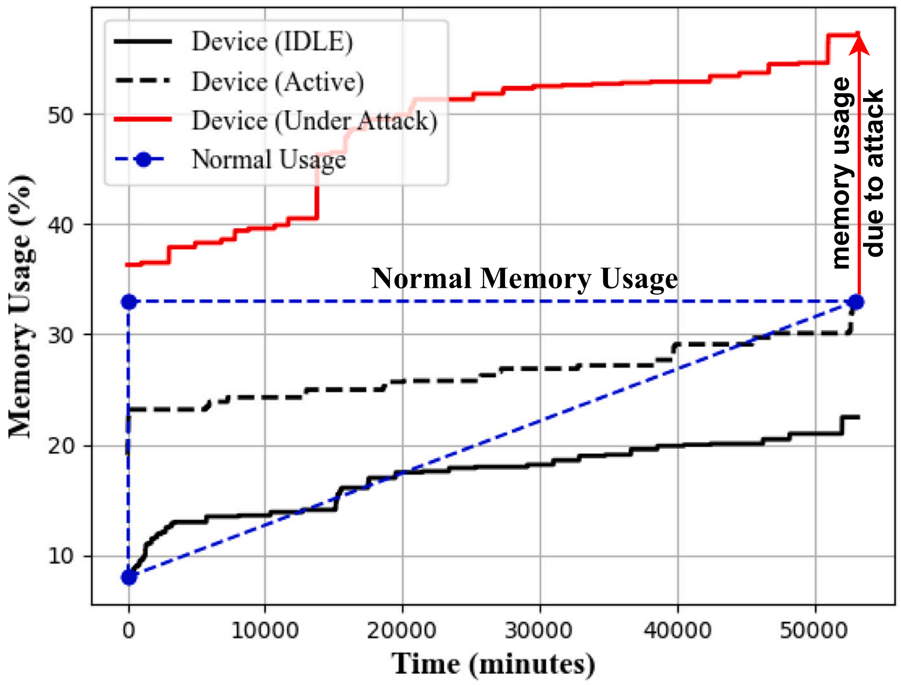


Fig. 5. Memory Usage for the Arduino under the attack.

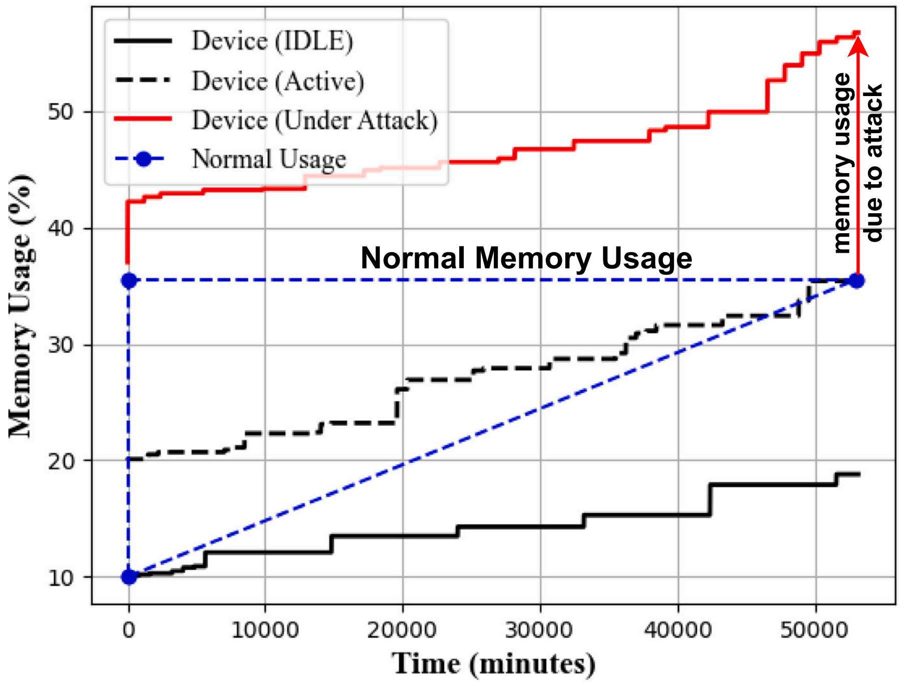


Fig. 6. Memory Usage for the Raspberry Pi under the attack.

However, we also calculate the normal energy consumption of edge and cloud devices using the same smart circuit presented in our previous work [13]. Therefore, the normal energy consumption ($\overline{E_C}$) of the edge device fluctuates between 10 J to 20 J per second, as shown in Fig. 7. The normal energy consumption of the cloud is measured to be less than 30 J per second, as shown in Fig. 9. Therefore, the following equation presented the calculation of energy consumption with or without the attack depending on the collected dataset.

$$\overline{E_C} = \frac{\sum_i E}{N}, \tag{5}$$

Where $\sum_i E$ is the sum of energy consumption measurements in the dataset, i represents the number of energy measurements of the IoT device, and N is the total number of measurements in the dataset.

Finally, Eq. (7) was utilized to establish a threshold value for distinguishing between normal and abnormal energy consumption behaviours, which will be described in detail in the next section.

6. The proposed mechanism

This section discusses the implementation and deployment of the ML mechanism to detect resource-constrained attacks on smart devices.

6.1. Evaluation of the attack detection

We are addressing the challenge of periodic intrusion detection for resource-constrained attacks. This task entails several steps: at time t_x, we upload the most recent resource-constraints usage samples from the devices used in our experiment to a remote database. Subsequently, the data collected in the database is processed to generate a detection outcome by time t_s. This result is then transmitted back to the devices and becomes available by time t_r. Following this, there might be an additional period of inactivity, denoted as t_i, until the next iteration commences (see Fig. 11). The time frame for detection, denoted as T_d, is calculated as $T_d = t_s + t_x + t_r + t_i$. To facilitate effective detection and prevent the accumulation of delayed results, it is vital to ensure that t_i is greater than or equal to zero. Assuming accurate detection and that the attack takes place at a random moment within the preceding interval T_d, the detection latency, denoted as t_l, is determined by the time between the attack's occurrence and the commencement of the subsequent detection cycle. This duration also includes the time required to upload and process resource usage data, such as memory and energy, to generate the final results.

By assuming a uniform distribution for the probability of an attack occurring at any random time within T_d, the associated mean delay is $\frac{T_d}{2}$. Consequently, the overall average detection latency can be expressed as: $t_l = \frac{T_d}{2} + t_x + t_s + t_r$.

It is important to note that the chosen algorithm, implementation strategy, and available processing resources all significantly determine the time required to process the data and generate the detection result.

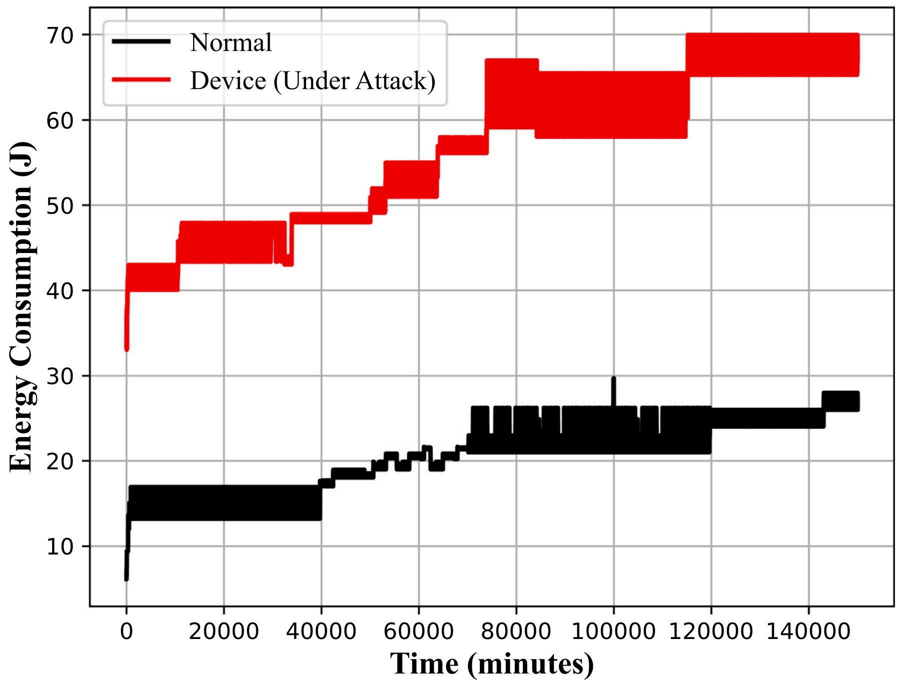


Fig. 7. Energy Consumption for the Edge device.

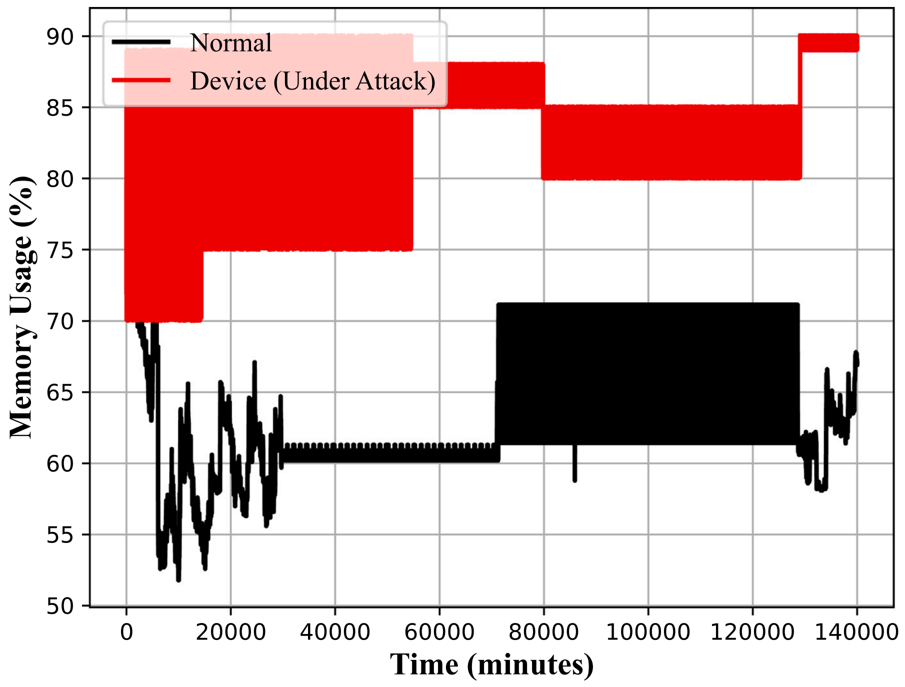


Fig. 8. Memory Usage for the Edge device.

6.2. Detecting resource constraints attacks

This section outlines the proposed methodologies for identifying resource-constrained attacks within IoT systems. Such attacks can manifest at various levels of the IoT architecture. The proposed approach leverages data analysis from different components of smart devices, including smart devices themselves, edge computing nodes, and cloud servers. By monitoring smart devices across

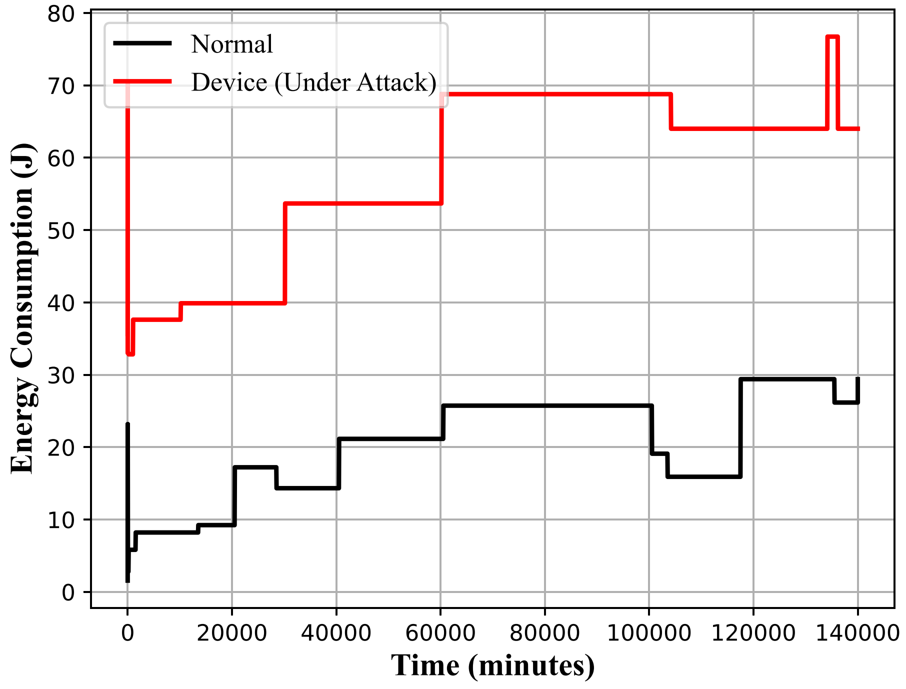


Fig. 9. Energy Consumption for the Cloud.

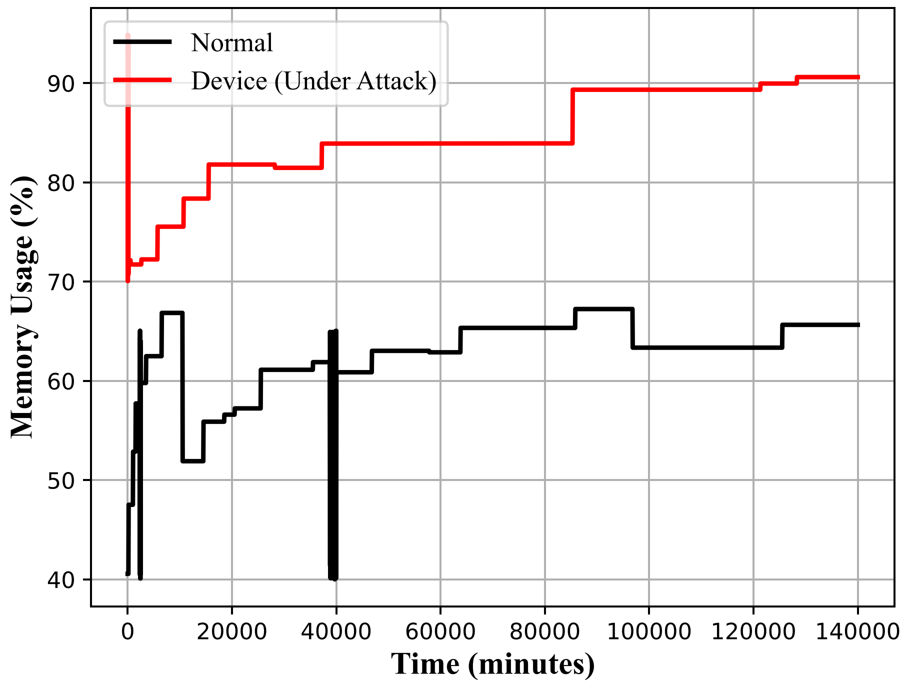


Fig. 10. Memory Usage for the Cloud.

different operational scenarios, such as with or without resource-constrained attacks, valuable data is collected. Subsequently, the following equations, detailed in this section, are utilized in conjunction with ML techniques to establish threshold values and identify potential attacks. To detect energy-based attacks, we employ the following equation:

$$Z = \frac{E_d - \mu}{\sigma} \tag{6}$$

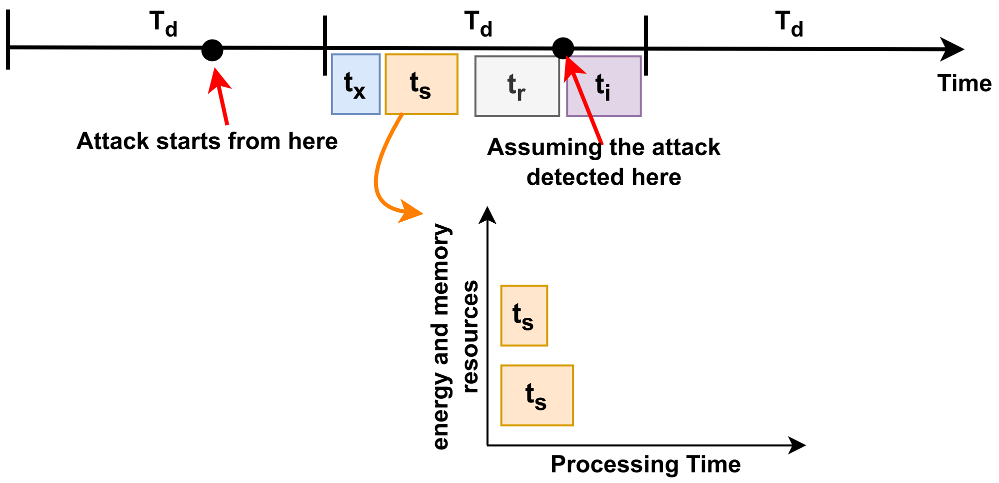


Fig. 11. Time sequence for real-time detecting resource constraint attacks using IDS detection with detection period T_d.

The Z-score measures the deviation of a data point from the mean in terms of standard deviations. By analysing the Z-score of energy consumption data, abnormal energy consumption patterns indicative of potential energy attacks can be identified. The mathematical equation, presented in (6), calculates the Z-score for energy consumption. Here, Z represents the Z-score, and E_d is the energy consumption value of the device d. Furthermore, μ is the mean energy consumption, and σ is the standard deviation of energy consumption.

A threshold value for the Z-score can be utilized to detect energy attacks. If the calculated Z-score surpasses the threshold value, which corresponds to the normal energy consumption $\overline{E}$, it suggests an anomaly or potential energy attack.

$$TH_{\overline{E}} = \frac{E_{min} + E_{max}}{2}, \tag{7}$$

In Eq. (7), $TH_{\overline{E}}$ denotes the threshold value for normal energy consumption, where E_{min} represents the minimum energy consumption and E_{max} signifies the maximum energy consumption observed in the absence of an attack. Fig. 4 illustrates the detection of energy attacks on smart IoT devices, categorizing results into normal and abnormal behaviours. Real-time detection of energy attacks depicted in this figure demonstrates high efficiency in identifying normal and abnormal energy consumption behaviours within a short timeframe. TinyML-based IDS are employed to classify normal and abnormal behaviours and generate final results after training datasets using various ML models, such as DT, RF, k-NN, and NB. In the next section, we also present the best ML model for IoT devices. Moreover, normal energy consumption for Raspberry Pi devices is observed to be below 1.425 J per second, whereas abnormal behaviour exceeds 2.5 J per second. Similarly, normal energy consumption for Arduino devices is below 1.064 J per second, while abnormal behaviour surpasses 1.6 J per second. Abnormal energy consumption for edge devices starts from 30.5 J and can exceed 68.5J per second, as depicted in Fig. 7. For cloud servers, normal energy consumption is below 30.3 J per second, whereas abnormal behaviour surpasses 30.5 J and can exceed 80J per minute, as shown in Fig. 9.

The TinyML models detect normal and abnormal memory usage attacks depending on the dataset used in this experiment. So the normal memory usage of the Raspberry Pi devices is less than 35%, while the abnormal usage exceeds to more 55%. In contrast, the normal usage of the Arduino device is less than 30%, and the abnormal behaviours start from 35% to reach more than 45%. Moreover, the memory usage of the edge and cloud are also calculated; therefore, the normal usage of memory is registered to be less than 70%, while the abnormal behaviour fluctuates between 70% and 90%. And for the cloud, the normal usage is less than 68%, while the abnormal is more than 70%.

Figs. 6, 5, 8, and 10 illustrate normal and abnormal memory usage behaviours across smart devices, edge nodes, and cloud servers.

7. Evaluation results

Considering various facets of the proposed methodology, this section delves into the description and analysis of our evaluation results. To achieve this, we examine the following metrics:

- True Positive Rate (TPR):

$$TPR = \frac{TP}{TP + TN} \tag{8}$$

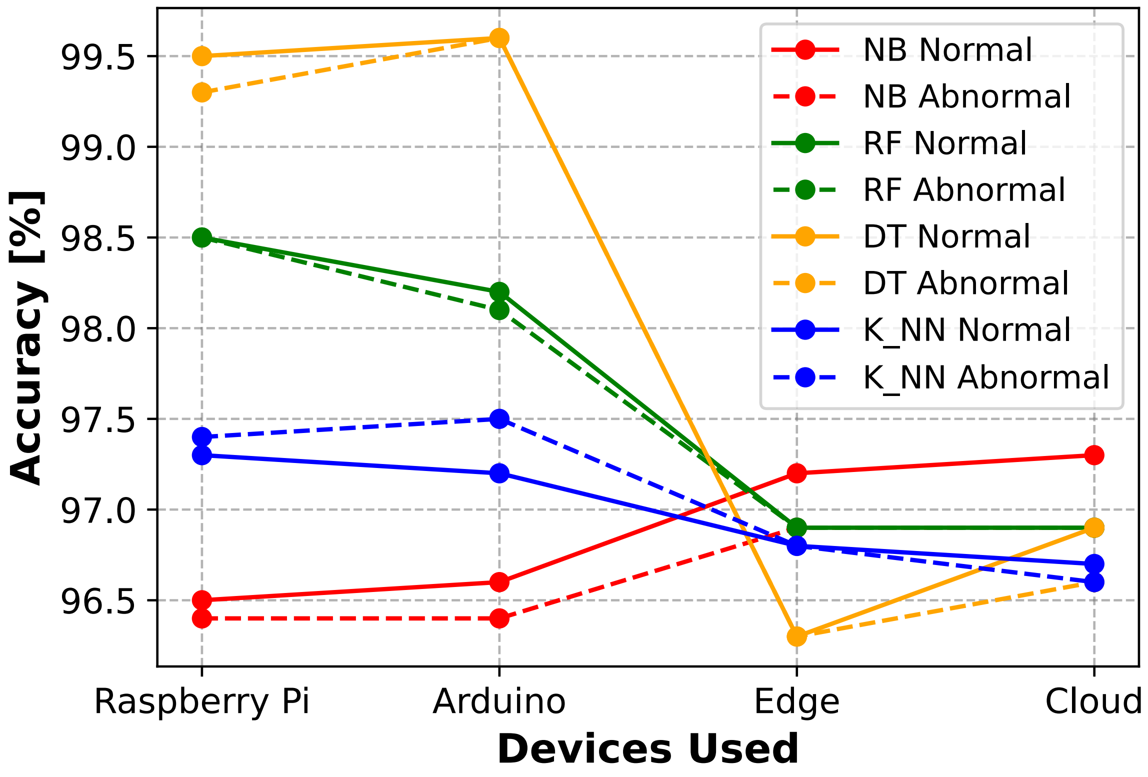


Fig. 12. Accuracy curve for different ML models.

- False Positive Rate (FPR):

$$FPR = \frac{FP}{FP + TN} \tag{9}$$

- Accuracy (A_{cc}): Accuracy is a statistical metric that quantifies the percentage of accurate predictions, encompassing both true positives and true negatives, within the overall number of cases evaluated. So, accuracy is calculated as follows [34]:

$$A_{cc} = \frac{TP + TN}{TP + FP + FN + TN} \times 100 \tag{10}$$

- Precision (Δ): is a measurement for determining the amount of confidence in forecasts. Compared to all favourably predicted observations, the fraction of True Positive anticipated observations. In addition, Ω_{TP} represents the total number of true positives, whereas Ω_{FP} represents the total number of false positives [34].

$$\Delta = \frac{\Omega_{TP}}{\Omega_{TP} + \Omega_{FP}} \tag{11}$$

- Recall (R): is a metric used to evaluate the performance of a classification model, particularly in binary classification tasks. It measures the ability of the model to correctly identify positive samples (instances belonging to the positive class) out of all the actual positive samples present in the dataset.

$$R = \frac{TP}{TP + FN} \tag{12}$$

- F1-score: which is defined as the harmonic mean of precision and recall

$$F1 = 2 * \frac{(\Delta * R)}{(\Delta + R)} \tag{13}$$

where TP are the true positives of correctly classified attack samples, TN are the true negatives of correctly classified samples are benign, FP is the false positives of benign samples, erroneously classified as malicious, and FN are the false negatives of attack samples, erroneously classified as benign.

To better elucidate the experimental results and provide a comprehensive understanding, we present Table 2, which details the training and detection times (in seconds) of various ML algorithms using datasets on both the Arduino Nano 33 BLE and Raspberry

Table 2
Time (s) consumed for training and detection of different ML models trained with IoT devices (Raspberry Pi and Arduino Nano 33 ble datasets.

ML model	Device	Training [s]	Detection [s]
NB	Raspberry Pi	0.02	29.4
	Arduino	0.05	39.9
	Edge	0.01	23.3
	Cloud	0.01	24.2
RF	Raspberry Pi	0.30	25.3
	Arduino	2.82	31.6
	Edge	0.26	20.1
	Cloud	0.25	20.3
DT	Raspberry Pi	0.09	24.6
	Arduino	0.19	27.5
	Edge	0.04	19.02
	Cloud	0.05	19.04
k-NN	Raspberry Pi	0.09	30.4
	Arduino	0.13	37.2
	Edge	0.08	23.6
	Cloud	0.09	23.7

Pi, serving as representative IoT devices. Our analysis highlights a direct correlation between dataset size and the training times of all ML algorithms across computational platforms. As dataset size increases, training time proportionally rises. Notably, the NB algorithm exhibits significantly shorter training times compared to other algorithms across all computational platforms, indicating its energy-efficient nature.

Conversely, the RF algorithm demonstrates markedly longer training times, especially when executed on IoT devices. This can be attributed to the algorithm's compatibility with multicore processors, which is prevalent in high-computational devices like cloud and edge platforms. Consequently, employing the RF algorithm for training on IoT devices is less resource-efficient. Additionally, due to the relatively limited computational capabilities of the Arduino, training execution times peak across all ML models. A notable issue specific to the Arduino device is encountered when the dataset size exceeds 4K, leading to disconnections. Consequently, this necessitates reinitiating dataset training and suspending any ongoing processes on the IoT device.

To evaluate the detection performance, we deploy TinyML models on smart devices and conduct experiments to assess detection time. Detection time is calculated by considering the start and end of detection by the ML models, expressed as $DT = T_{start} + T_{end}$. Table 2 illustrates the detection time calculated for different ML models and smart devices, namely Raspberry Pi and Arduino. Observations indicate varying detection times among different smart devices and ML models. Notably, DT's detection time is shorter than RF, k-NN, and NB. The detection time of the Raspberry Pi is 24.6 s, Arduino is 27.5 s, the edge is 19.2 s, and the cloud is 19.04. However, the k-NN consumes more detection time in the Raspberry Pi device, with access to 30.4 s, compared to the Arduino, which accessed 37.2 s. The Arduino takes more detection time and access to 39.9 s with the NB model compared to the Raspberry Pi, which takes 29.4. For the edge and cloud devices, the detection time of NB and K-NN is almost the same, which takes about 23.3 s for NB and 23.6 s for the k-NN for the edge and for the cloud is 23.2 s for NB and 23.7 s for the K-NN. Moreover, Fig. 13 compares four machine learning algorithms based on their False Positive Rate (FPR) and False Negative Rate (FNR). The DT demonstrates superior performance with the lowest FNR across various FPR values, indicating a strong ability to minimize false negatives while maintaining a low false positive rate. This makes the DT particularly effective for high classification accuracy. RF also performs well, offering a balanced trade-off between FPR and FNR, though it does not consistently outperform the DT. NB shows a higher FNR than the DT and RF, suggesting it is less effective at minimizing false negatives, despite its relatively low FPR. KNN generally exhibits a higher FPR, resulting in more false positives, but maintains a reasonable FNR, indicating it is less effective in reducing false negatives relative to its higher FPR. In summary, considering both training and detection times, the Naive Bayes algorithm emerges as the best-performing model. Its efficient training process and relatively short detection times make it suitable for resource-constrained IoT environments. Additionally, its lightweight nature aligns well with the limited computational capabilities of IoT devices. Thus, NB is the preferred choice, particularly in environments where energy efficiency and computational resource constraints are paramount considerations.

8. Discussion

The widespread adoption of IoT technologies in mission-critical systems like traffic control, industrial automation, and healthcare underscores the importance of their reliable and secure operation. These systems are vulnerable to various attacks, including DoS attacks, which can pose severe risks to human safety, especially in scenarios such as hospital ICUs or railway traffic control. Many modern IoT systems depend on cloud-based data processing to offer intelligent, personalized solutions through ML and DL technologies. However, this reliance on cloud connectivity introduces network issues and security vulnerabilities. TinyML provides a promising solution by enabling the local execution of ML and DL models on resource-constrained devices without the need for constant internet connectivity. This local processing can improve the privacy, security, and availability of IoT systems, particularly

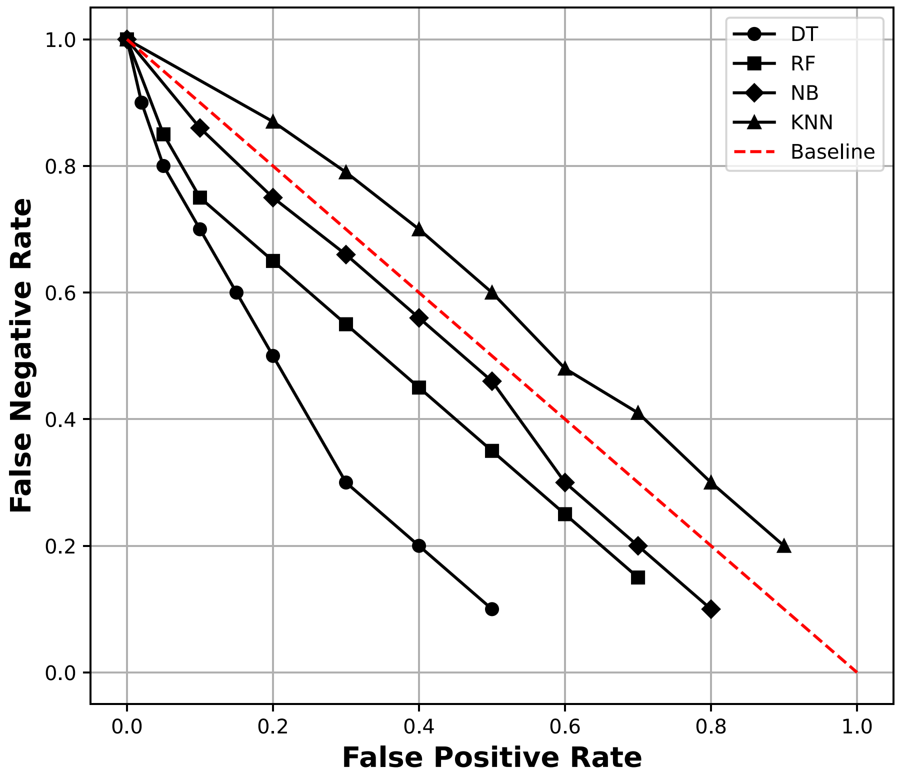


Fig. 13. Comparison of Algorithm Performances between Algorithms using FPR and FNR.

Table 3
The proposed solution in comparison with the existing solution.

Ref.	Method	Device type	Data sources	Resource constraints attack detection	Accuracy [%]
This Work	DT, K_NN, NB, RF	Raspberry Pi, Arduino, Edge device, Cloud	Simulation and Real Dataset	Energy attack Memory attack	99.5, 96.5, 97.3 and 98.3
[35]	LSTM AE, MSM AE	Edge device	Simulation dataset	malicious attacks zero-day attack	99.9
[36]	IDs	IoT devices	real data	–	–
[37]	TinyML	smart IoT device	Simulation dataset	gas leakage attacks	0.70 ÷ 0.83

in mission-critical applications. Our work leverages different ML techniques to address these challenges, including TinyML for IoT devices, Edge ML for edge computing, and Cloud ML for cloud-based systems.

We deploy ML models developed using TensorFlow on IoT edge devices. Traditional ML models are difficult to deploy on small devices due to size and accuracy constraints. To overcome this, we optimize models using TensorFlow Lite, ensuring they fit within the memory constraints of the target devices. Optimization techniques such as quantization, pruning, and clustering are applied to reduce model size and latency while preserving accuracy.

(1) We convert the trained model to a ".tflite" format using the TFlite converter.
(2) The converted model is deployed on the IoT device.
(3) The model is loaded onto the IoT device, where the interpreter allocates memory and activates the model for inference.

In addition to presenting TinyML results in Tables 2 and 4, we evaluate metrics such as accuracy, training time, recall, and precision. Algorithms such as k-NN, DT, and RF showed strong performance, though RF's high training time makes it less suitable for resource-constrained IoT devices. NB, with its shorter training time, emerges as a viable alternative. We measured energy consumption and memory usage to further evaluate system performance under malicious attacks, focusing on detecting attacks that specifically target these critical resources. This allows us to identify whether the system is under attack by analysing energy depletion or memory exhaustion patterns.

Table 4
Performance comparison of different ML models on various devices and conditions.

ML model	Device	Condition	Accuracy [%]	Recall	Precision	F1-Score
NB	Raspberry Pi	Normal	96.5	0.86	0.97	0.91
		Abnormal	96.4	0.86	0.96	0.90
	Arduino	Normal	96.6	0.86	0.97	0.91
		Abnormal	96.5	0.85	0.97	0.90
	Edge	Normal	97.2	0.89	0.99	0.94
		Abnormal	96.9	0.88	0.98	0.92
	Cloud	Normal	97.3	0.90	0.99	0.94
		Abnormal	96.9	0.89	0.98	0.93
RF	Raspberry Pi	Normal	98.5	0.95	0.99	0.96
		Abnormal	98.5	0.95	0.99	0.96
	Arduino	Normal	98.2	0.94	0.98	0.95
		Abnormal	98.1	0.94	0.98	0.95
	Edge	Normal	96.9	0.89	0.98	0.93
		Abnormal	96.5	0.85	0.97	0.90
	Cloud	Normal	96.9	0.89	0.98	0.94
		Abnormal	96.6	0.85	0.98	0.91
DT	Raspberry Pi	Normal	99.5	0.96	0.99	0.97
		Abnormal	99.3	0.96	0.98	0.97
	Arduino	Normal	99.6	0.96	0.99	0.97
		Abnormal	99.6	0.96	0.99	0.97
	Edge	Normal	96.9	0.96	0.98	0.96
		Abnormal	96.3	0.95	0.99	0.97
	Cloud	Normal	96.9	0.96	0.98	0.96
		Abnormal	96.3	0.95	0.99	0.97
K_NN	Raspberry Pi	Normal	97.3	0.85	0.97	0.90
		Abnormal	97.2	0.85	0.97	0.90
	Arduino	Normal	97.5	0.86	0.97	0.91
		Abnormal	97.5	0.86	0.98	0.91
	Edge	Normal	96.8	0.88	0.98	0.92
		Abnormal	96.6	0.86	0.96	0.90
	Cloud	Normal	96.8	0.88	0.98	0.92
		Abnormal	96.6	0.86	0.96	0.90

Our analysis shows that NB offers significantly faster training times (ranging from 0.01 s to 0.05 s) and detection times (between 23.3 s to 39.9 s) compared to RF, which, while achieving higher accuracy (up to 98.5%), requires substantially more training time. For instance, on resource-constrained devices like the Arduino, RF's training time reaches as high as 2.82 s. This trade-off between speed and accuracy is a key factor when selecting the appropriate algorithm based on the specific application requirements. NB's rapid processing capabilities make it particularly advantageous for real-time applications, such as healthcare monitoring, where quick responses are critical. In such scenarios, NB's slightly lower accuracy (ranging from 96.4% to 97.3%) is an acceptable trade-off in exchange for faster detection times, ensuring timely alerts that could potentially save lives. On the other hand, RF's higher accuracy, despite its longer training and detection times, is more suitable for scenarios where precision is paramount and computational resources are less of a constraint. For example, RF is ideal for predictive modelling or long-term data analysis, where high-quality predictions outweigh the need for rapid processing, as also presented in Fig. 12. Moreover, Fig. 14 illustrates the comparative performance of four machine learning models regarding loss reduction and accuracy improvement over 20 training epochs. The left y-axis represents the loss, while the right y-axis shows the corresponding accuracy. The loss curves (in red) depict how each algorithm minimizes the error during training, with models like RF and DT showing faster convergence and lower final loss values. Conversely, K-NN exhibits slower convergence, as indicated by its higher loss over epochs. The accuracy curves (in black) demonstrate how well the models perform regarding correct predictions, improving accuracy as loss decreases. RF consistently achieves the highest accuracy, followed by DT, while NB and K-NN show slightly lower but stable performance. This figure provides a clear visualization of the trade-offs between convergence speed and final model accuracy, allowing for the verification of the models' learning effectiveness over time. Our proposed mechanism detects resource constraint attacks using ML techniques across the edge-cloud continuum. By tailoring TinyML models for the IoT, edge, and cloud levels, we enable flexible detection of resource constraint attacks, such as energy and memory attacks. Although our system was initially designed for specific smart devices, it can be extended to other devices with similar architectures.

Therefore, Table 3 illustrates the existing solutions in terms of several parameters against the solution proposed in this paper. We highlight several points that we consider crucial, e.g., Accuracy, data sources, attack type, and methods. In [35], an advanced

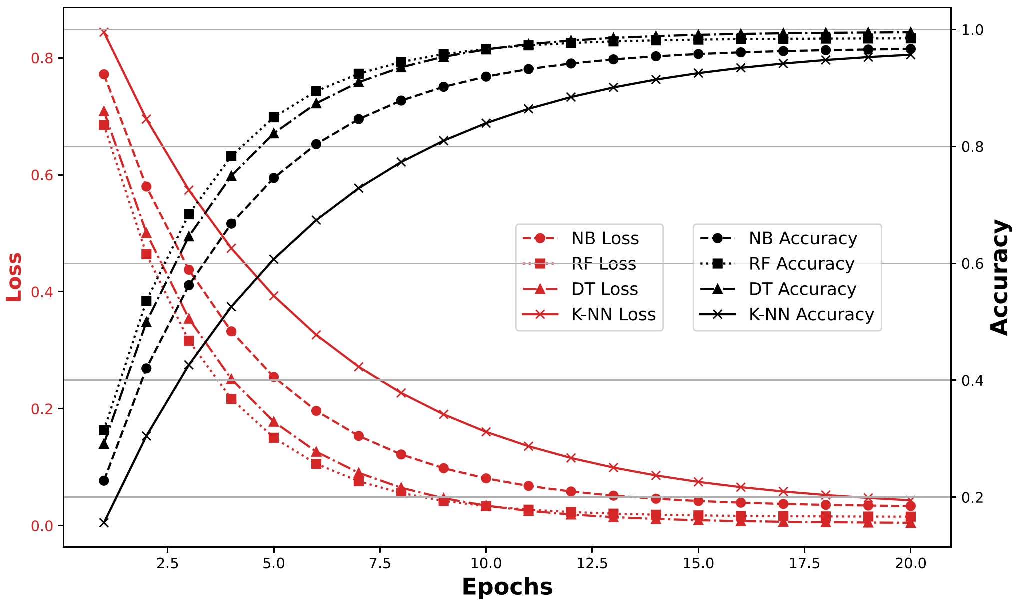


Fig. 14. Comparative Analysis of Loss and Accuracy Curves for Different Machine Learning Models.

autoencoder-based IDS significantly improved detection accuracy and versatility for malicious packets. Extensive experiments validated its superior performance and suitability for resource-constrained edge devices. In [36], an efficient intrusion detection framework is proposed for energy-constrained IoT devices, which is crucial in IIoT. We explore host-edge collaboration to detect intrusions effectively, minimizing energy and communication overhead. In [37], a TinyML-based gas leakage detection system is presented, capable of detecting anomalies and alerting occupants using BLE technology and an LCD screen. In contrast, our focus is specifically on detecting energy and memory attacks in different levels, IoT devices, Edge, and cloud compared to other presented works.

While our approach focuses on optimizing traditional ML models for resource-constrained IoT devices, future research could explore more lightweight versions of these algorithms or new models specifically designed for such environments. Approaches like Spiking Neural Networks [38] or resource-aware ML models could offer further performance gains, particularly for energy and memory-constrained devices. Incorporating such algorithms would provide an additional layer of adaptability, further enhancing real-time security detection.

9. Conclusion and future work

In conclusion, the pressing challenge of resource constraint attacks on IoT applications underscores the vulnerabilities inherent in smart devices burdened by limited resources. While ML models offer promise in fortifying IoT security with their real-time responsiveness and privacy-preserving capabilities, their effective deployment in resource-constrained environments remains a significant hurdle. This study addressed the escalating threat of resource-constrained attacks by evaluating various ML deployment strategies for intrusion detection in edge-cloud IoT systems, particularly focusing on energy and memory-based attacks. Our comprehensive comparison of deployment approaches, encompassing cloud-based, edge-based, and IoT device-based methods, sheds light on their performance in detecting such attacks, alongside their associated training and detection times. Our findings highlight the efficiency of the NB algorithm, demonstrating quicker training and detection processes, both completed in under 40 seconds. At the same time, the RF algorithm requires less detection time. The experimental results, a testament to the robustness of our proposed approach, underscore its ability to effectively detect resource-constrained attacks on smart devices, achieving an impressive accuracy of over 96.9%. As we navigate the IoT landscape, bolstering security through adept utilization of ML and targeted deployment strategies is not just a need but a necessity, ensuring a safer and more reliable IoT ecosystem for all. In future research, we aim to expand upon this by exploring lightweight versions of existing algorithms or entirely new models optimized for resource constraints, such as Spiking Neural Networks or other resource-aware deep learning techniques. This exploration will help further optimize security performance across a broader range of IoT devices, particularly those with stricter memory, power, and energy limits.

CRediT authorship contribution statement

Zainab Alwaisi: Writing – review & editing, Writing – original draft, Validation, Software, Resources, Project administration, Methodology, Investigation. **Tanesh Kumar:** Writing – review & editing. **Erkki Harjula:** Writing – review & editing. **Simone Soderi:** Supervision.

Declaration of competing interest

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

Acknowledgements

This work has been partially supported by Business Finland Eware-6G (Grant Number : 8819/31/2022) and Research Council of Finland 6G Flagship (Grant Number : 346208) programmes, and SUSTAINSEC-6G project (Grant Number: 97581308) as part of the India Pilot initiative (FICORE) from the Finnish Ministry of Education and Culture.

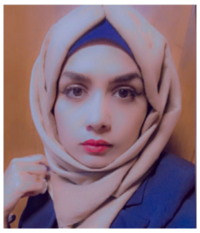
Data availability

No data was used for the research described in the article.

References

[1] A. Boyanapalli, A. Shanthini, A comparative study of techniques, datasets and performances for intrusion detection systems in IoT, in: D.J. Hemanth, G. Vadivu, M. Sangeetha, V.E. Balas (Eds.), Artificial Intelligence Techniques for Advanced Computing Applications, Springer Singapore, Singapore, 2021, pp. 225–236.
[2] M. Ramaiah, V. Chandrasekaran, V. Ravi, N. Kumar, An intrusion detection system using optimized deep neural network architecture, Trans. Emerg. Telecommun. Technol. 32 (4) (2021) http://dx.doi.org/10.1002/ett.4221.
[3] M. Ghobaei-Arani, A. Souri, A.A. Rahmanian, Resource management approaches in fog computing: a comprehensive review, J. Grid Comput. 18 (2019) 1–42.
[4] A. Souri, M. Ghobaei-Arani, Cloud manufacturing service composition in IoT applications: A formal verification-based approach, Multimedia Tools Appl. 81 (19) (2022) 26759–26778, http://dx.doi.org/10.1007/s11042-021-10645-1.
[5] Kamaldeep, M. Dutta, J. Granjal, Towards a secure internet of things: A comprehensive study of second line defense mechanisms, IEEE Access 8 (2020) 127272–127312.
[6] G. Simoglou, G. Violettas, S. Petridou, L. Mamatas, Intrusion detection systems for RPL security: A comparative analysis, Comput. Secur. 104 (C) (2021) http://dx.doi.org/10.1016/j.cose.2021.102219.
[7] J. Jamali, B. Bahrami, A. Heidari, P. Allahverdizadeh, F. Norouzi, Towards the internet of things, Springer, 2020.
[8] J. Balasundaram, Retracted: A novel optimized bat extreme learning intrusion detection system for smart internet of things networks, Int. J. Commun. Syst. 34 (7) (2021) e4729.
[9] I. Stojmenovic, S. Wen, X. Huang, H. Luan, An overview of fog computing and its security issues, Concurr. Comput.: Pract. Exper. 28 (10) (2016) 2991–3005.
[10] P. Warden, D. Situnayake, Tinyml: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers, O'Reilly Media, 2019.
[11] I.N.K. Wardana, J.W. Gardner, S.A. Fahmy, Optimising deep learning at the edge for accurate hourly air quality prediction, Sensors 21 (4) (2021) 1064.
[12] Z. Alwaisi, S. Soderi, R. De Nicola, Mitigating and analysis of memory usage attack in IoE system, in: N.-S. Vo, H.-A. Tran (Eds.), Industrial Networks and Intelligent Systems, Springer Nature Switzerland, Cham, 2023, pp. 296–314.
[13] Z. Alwaisi, S. Soderi, R.D. Nicola, Energy cyber attacks to smart healthcare devices: A testbed, in: Y. Chen, D. Yao, T. Nakano (Eds.), Bio-Inspired Information and Communications Technologies, Springer Nature Switzerland, Cham, 2023, pp. 246–265.
[14] Z. Al-Waisi, S. Soderi, R. De Nicola, Detection of energy consumption cyber attacks on smart devices, 12, (4) EAI-SPRINGER, 2023, p. 1927,
[15] Z.A. AlWaisi, Optimized Monitoring and Detection of Internet of Things resources-constraints Cyber Attacks, IMT Institute for Advanced Studies Lucca, 2023.
[16] Z. Alwaisi, S. Soderi, Towards robust IoT defense: Comparative statistics of attack detection in resource-constrained scenarios, 2024.
[17] M.P. F. Aubet, DS2os traffic traces, 2018, URL https://rb.gy/mwba3a.
[18] I. Sharafaldin, A.H. Lashkari, A.A. Ghorbani, Toward generating a new intrusion detection dataset and intrusion traffic characterization., ICISSp 1 (2018) 108–116.
[19] M.A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, H. Janicke, Edge-iIoTset: A new comprehensive realistic cyber security dataset of IoT and iIoT applications for centralized and federated learning, IEEE Access 10 (2022) 40281–40306, http://dx.doi.org/10.1109/ACCESS.2022.3165809.
[20] A. Alotaibi, M.A. Rassam, Adversarial machine learning attacks against intrusion detection systems: A survey on strategies and defense, Future Internet 15 (2) (2023) 62.
[21] A. Sabovic, M. Aernouts, D. Subotic, J. Fontaine, E. De Poorter, J. Famaey, Towards energy-aware tinyml on battery-less IoT devices, Internet of Things 22 (2023) 100736.
[22] V. Tsoukas, A. Gkogkidis, A. Kampa, G. Spathoulas, A. Kakarountas, Enhancing food supply chain security through the use of blockchain and tinyml, Information 13 (5) (2022) 213.
[23] E. Harjula, P. Karhula, J. Islam, T. Leppänen, A. Manzoor, M. Liyanage, J. Chauhan, T. Kumar, I. Ahmad, M. Ylianttila, Decentralized IoT edge nanoservice architecture for future gadget-free computing, IEEE Access 7 (2019) 119856–119872.
[24] A. Krayden, M. Schohet, O. Shmueli, D. Shlenkevitch, T. Blank, S. Stolyarova, Y. Nemirovsky, CMOS-mems gas sensor dubbed GMOS for SelectiveAnalysis of gases with tiny edge machine learning, Eng. Proc. 27 (1) (2022) 81.
[25] S. Gupta, et al., An effective model for anomaly IDS to improve the efficiency, in: 2015 International Conference on Green Computing and Internet of Things (ICGCIoT), IEEE, 2015, pp. 190–194.
[26] N.K. Thanigaivelan, E. Nigussie, R.K. Kanth, S. Virtanen, J. Isoaho, Distributed internal anomaly detection system for internet-of-things, in: 2016 13th IEEE Annual Consumer Communications & Networking Conference, CCNC, IEEE, 2016, pp. 319–320.
[27] N. Tekin, A. Acar, A. Aris, A.S. Uluagac, V.C. Gungor, Energy consumption of on-device machine learning models for IoT intrusion detection, Internet Things 21 (2023) 100670.
[28] B. Sudharsan, D. Sundaram, P. Patel, J.G. Breslin, M.I. Ali, Edge2guard: Botnet attacks detecting offline models for resource-constrained iot devices, in: 2021 IEEE International Conference on Pervasive Computing and Communications Workshops and Other Affiliated Events (PerCom Workshops), IEEE, 2021, pp. 680–685.

[29] S. Yılmaz, E. Aydogan, S. Sen, A transfer learning approach for securing resource-constrained iot devices, IEEE Trans. Inf. Forensics Secur. 16 (2021) 4405–4418.
[30] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J. McLachlan, A. Ng, B. Liu, P.S. Yu, et al., Top 10 algorithms in data mining, Knowl. Inf. Syst. 14 (2008) 1–37.
[31] B. Charbuty, A. Abdulazeez, Classification based on decision tree algorithm for machine learning, J. Appl. Sci. Technol. Trends 2 (01) (2021) 20–28.
[32] K. Cao, Y. Liu, G. Meng, Q. Sun, An overview on edge computing research, IEEE Access 8 (2020) 85714–85728, http://dx.doi.org/10.1109/ACCESS.2020.2991734.
[33] L. Dutta, S. Bharali, Tinyml meets iot: A comprehensive survey, Internet of Things 16 (2021) 100461.
[34] Z. Yang, X. Liu, T. Li, D. Wu, J. Wang, Y. Zhao, H. Han, A systematic literature review of methods and datasets for anomaly-based network intrusion detection, Comput. Secur. 116 (C) (2022) http://dx.doi.org/10.1016/j.cose.2022.102675.
[35] N. Borgioli, L. Thi Xuan Phan, F. Aromolo, A. Biondi, G. Buttazzo, Real-time packet-based intrusion detection on edge devices, in: CPS-IoT Week '23, Association for Computing Machinery, New York, NY, USA, 2023, pp. 234–240, http://dx.doi.org/10.1145/3576914.3587551.
[36] J. Arshad, M.A. Azad, M.M. Abdeltaif, K. Salah, An intrusion detection framework for energy constrained IoT devices, Mech. Syst. Signal Process. 136 (2020) 106436.
[37] V. Tsoukas, A. Gkogkidis, E. Boumpa, S. Papafotikas, A. Kakarountas, A gas leakage detection device based on the technology of tinyml, Technologies 11 (2) (2023) 45.
[38] R.V. Florian, A reinforcement learning algorithm for spiking neural networks, in: Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC'05, IEEE, 2005, pp. 8–pp.



Zainab Alwaisi received her Bachelor's and Master's degrees in Software Engineering from the University of Northampton, UK, in 2016 and 2017, respectively, where she was the top student across the entire university in the Computer Science program. She earned her PhD in Computer science and systems engineering in 2023 at IMT School for Advanced Studies in Lucca, Italy. Since July 2023, she has worked as a research collaborator in cyber security at IMT School for Advanced Studies in Lucca, Italy. Her primary research interest is in securing IoT, smart devices, 6G security, and addressing resource constraints. She aims to provide lightweight detection mechanisms and applies TinyML and ML techniques to enhance protection against cyberattacks. Additionally, she is actively engaged in research related to 6G security. She has served as a reviewer for Springer Journals and has published several conference and journal papers.

Tanesh Kumar (Member, IEEE) is currently working as a Staff Scientist in the School of Electrical Engineering at Aalto University, Finland. He received the D.Sc. degree in communications engineering from the University of Oulu, Finland, in 2020, the M.Sc. degree in computer science from South Asian University, New Delhi, India, in 2014, and the B.E. degree in computer engineering from the National University of Sciences and Technology (EME), Pakistan, in 2012. Previously he has worked as a postdoctoral researcher and project manager at CWC-NS, University of Oulu (2021–2023). He has co-authored over 40 peer-reviewed scientific articles. His current research interests include IoT security, privacy and trust, 5G/6G, Edge-AI, Blockchain/DLTs, and Wireless Communication.

Erkki Harjula is a tenure-track assistant professor at the CWC-NS research unit, Faculty of Information Technology and Electrical Engineering (ITEE), University of Oulu, Finland. He received a D.Sc. degree in 2016 and a M.Sc. degree in 2007 from the University of Oulu. Currently, he works on wireless system-level architectures for future digital healthcare, focusing on intelligent, trustworthy distributed computing and communication architectures. He has a background in edge computing, mobile and IoT networks and green computing. He has co-authored more than 90 international peer-reviewed articles. He is also an Associate Editor of Springer Wireless Networks journal.

Simone Soderi (SMIEEE) received his M.Sc. degree in 2002 from the University of Florence and his Dr.Sc. Degree in 2016 from the University of Oulu, Finland. His expertise ranges from cybersecurity and wireless communications to embedded systems. He is currently an Assistant Professor at the IMT School for Advanced Studies in Lucca, Italy, and an Adjunct Professor at the University of Padua, Italy, where he teaches in the master's degree program in cybersecurity. His research topics include cybersecurity for critical infrastructure systems, 6G, covert channels, network security, physical layer security, electromagnetic emission security, VLC, and UWB. He has been a TPC member of several conferences and served as a reviewer of many IEEE Transactions. Dr. Soderi has published journal and conference papers and book chapters. He holds five patents on wireless communications and positioning.