scientific reports

Check for updates

OPEN

A lightweight framework to secure IoT devices with limited resources in cloud environments

Vivek Kumar Pandey[1,7], Dinesh Sahu[2,7], Shiv Prakash[3,7]✉, Rajkumar Singh Rathore[4,7]✉, Pratibha Dixit[5,7] & Iryna Hunko[6,7]✉

Billions of IoT devices increasingly function as gateways to cloud infrastructures, making them an inevitable target of cyber threats because of the limited resources and low processing capabilities of IoT devices. This paper proposes a lightweight decision tree-based intrusion detection framework suitable for real-time anomaly detection in a resource-constrained IoT environment. Finally, the model also makes use of a novel leaf-cut feature optimization strategy and tight adaptive cloud edge intelligence to achieve high accuracy while minimizing memory and computation demand. In terms of memory, they also use only 12.5 MB in it and evaluated on benchmark datasets including NSL-KDD and Bot-IoT, it gives an accuracy of 98.2% and 97.9%, respectively, and less than 1% false positives, thereby giving up to 6.8% accuracy over some traditional models such as SVM and Neural Networks and up to 78% less energy. It is deployed on Raspberry Pi nodes and can do real-time inference in less than 1 ms and 1,250 samples/sec. Due to the energy efficient, scalable, and interpretable architecture of the proposed solution, it can be implemented as a security solution for IoT use cases in Smart cities, industrial automation, health care, and autonomous vehicles.

The rapid growth of the Internet of Things (IoT) technology has transformed different areas of society, including healthcare, smart cities, and industrial automation[1,2]. As IoT is being rapidly deployed in critical sectors such as healthcare, smart cities, and transportation, the possibility of security breaches in resource-constrained devices can result in severe operational and safety risks. Research in optimization frameworks for IoT security has attracted much attention in the past couple of years because these frameworks have the possibility of improving the system robustness as well as the resource control[3]. IoT devices achieve their advancements through data collection and processing as well as data exchange capabilities on cloud-based and edge-computing platforms[4]. IoT devices have long-existing security issues because their connections have accelerated while resource-limited devices lack enough processing power and energy to apply reliable security measures[5]. Restrictions on IoT devices create significant cyber threat vulnerabilities because they expose networks to distributed denial-of-service attacks, malware injections, and unauthorized data access events[6]. Figure 1 shows the different kind of security attacks on IoT devices in cloud enabled-environment. The current intrusion detection models for IoT security fail to establish premiere detection precision and usable computational requirements[7]. The high detection capabilities of traditional Support Vector Machines (SVMs) and neural networks limit their practical use on resource-restricted IoT devices since their computing costs remain too expensive[8].

Implementing lightweight models results in compromised security because they exchange high accuracy detection with efficiency benefits[9,10]. The implementation of cloud-based security protocols suffers from delays and requires uninterrupted network connections since they do not suit actual time processes[11]. A framework that can detect anomalies efficiently in real-time needs to be developed since existing solutions lack both energy efficiency and lightweight design.

In Fig. 2, IoT devices, cloud infrastructure, and security elements are enclosed in this system to promote security to the gadgets, and secure communication within the IoT frameworks. Therefore, in the center of the architecture, IoT devices such as smart cameras, environmental sensors, access control systems, wearables and

[1]Department of Electronics and Communication, University of Allahabad, Prayagraj, Uttar Pradesh, India. [2]SCSET, Bennett University, Plot Nos 8, 11, TechZone 2, Greater Noida, Uttar Pradesh 201310, India. [3]Department of Electronics and Communication, University of Allahabad, Prayagraj, Uttar Pradesh, India. [4]Cardiff School of Technologies, Cardiff Metropolitan University, Cardiff, UK. [5]King George Medical University, Lucknow, Uttar Pradesh, India. [6]Department of electric station and system, Vinnytsia National Technical University, Vinnytsia 21021, Ukraine. [7]These authors contributed equally: These authors contributed equally: Vivek Kumar Pandey, Dinesh Sahu, Shiv Prakash, Rajkumar Singh Rathore, Pratibha Dixit and Iryna Hunko. ✉email: shivprakash@allduniv.ac.in; rsrathore@cardiffmet.ac.uk; iryna_hunko@vntu.edu.ua

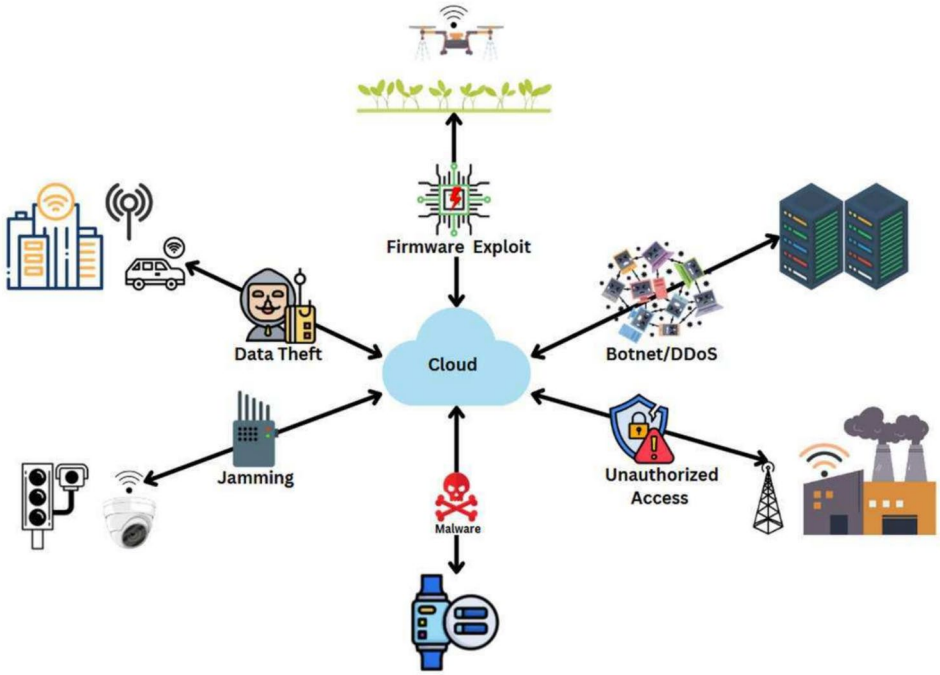

Fig. 1. Common cyber threats in cloud-based IoT systems.

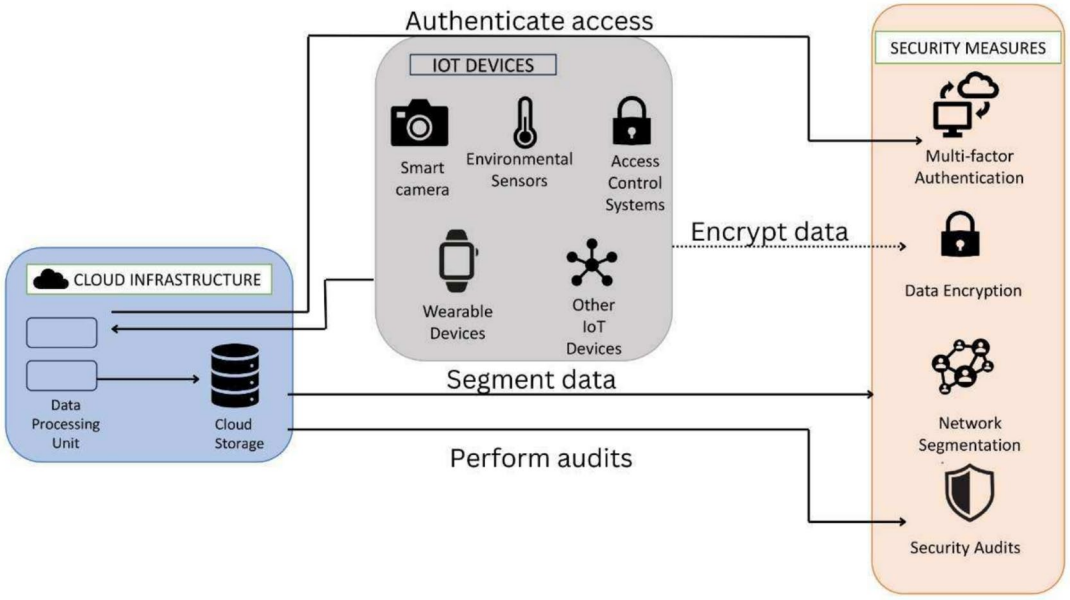

Fig. 2. IoT-cloud system architecture with key security mechanisms for data protection.

other IoT gadgets are included and produce data constantly. These devices are prone to various forms of security threats and, thus, need protection measures. This centrally collected data is then relay the cloud infrastructure where there is a Data Processing Unit and Cloud Storage, for large-scale IoT data collection and management. To ensure that there is integrity and confidentiality of data relayed between the IoT devices and the cloud a data encryption procedure was incorporated into the architecture. Also, security measures including the multifactor authentication ensure the user and device's access to the system by minimizing unauthorized invasions. The network structure is also designed to decentralize the network especially by having different segments of the network to reduce harm in case of some segment compromise. This segmentation also makes sure that if a section of the network is infected, the other parts will not be affected. Also, there are regular security audit checks aimed at assessing the security level, discovering the open weaknesses and check compliance with

security standards. Within the cloud facilities, the data is processed and stored by taking the help of analytics as well as storage system so that it can be operated in real-time and kept archived for a longer time. This way, the architecture guarantees security at the device level, in communication, in the cloud, as well as systematic, and establishing the IoT security environment.

Despite improvements in intrusion detection, most existing solutions are either computationally expensive or are not sufficiently accurate, and any such models have not been verified on real IoT hardware in real time[12,13]. The paper addresses critical IoT security challenges such as resource limitations, cloud latency, inefficient threat identification, hybrid edge- cloud inefficiencies, accuracy-efficiency trade-offs, scalability, and IoT-specific attacks. The paper suggests a lightweight decision tree-based system that reduces memory footprint, facilitates real-time edge inference, and increases adaptability with incremental cloud-edge updates. For power efficiency, it provides effective anomaly detection along with federated learning and distributed threat intelligence that enable scaling on disparate IoT networks in order to suppress threats such as data spoofing, botnets, and protocol exploits.

Objectives

The research questions of this study are as follows:

1. To design a simple lightweight decision tree-based mechanism to improve on the conventional anomaly detection in IoT challenged environments.
2. To minimize the computational and memory complexity to improve model accuracy, select the best features and construct the model architecture.
3. To provide a framework with the ability to update the models with cloud analytics while maintaining edge processing capabilities.
4. To further assess the proposed framework in terms of accuracy and time complexity as well as compare the performance of the state-of-art machine learning models on NSL-KDD and Bot-IoT datasets.
5. To test the framework on real IoT devices in order to determine the feasibility of the proposed approach for real-time security use cases.

Contribution

The subject of this work is a new approach at providing security to IoT networks using a lightweight decision tree design; this comes with the following discoveries:

1. Optimized Decision Tree Model: There is therefore the use of a leaf-cut feature optimization in the suggested framework so that threats may be detected in real-time with consideration of the resources used[14].
2. Cloud-Based Threat Intelligence: It uses cloud-based analytics for updating the detection schemes which makes it possible to counter new threats in real-time[15,16].
3. Relative Memory Usage: The usage of memory is also much lesser (12.5-13.1 MB) with the proposed model and hence the energy requirements for running the algorithm are also less (0.45-0.48W)[17].
4. Performance Evaluation: It is to evaluate the proposed framework with six machine learning models, while NSL-KDD and Bot-IoT datasets are used and achievedan accuracy of 98.2% and 97.9% as well as 0.8-0.9% of false positive rate.
5. Real-World Validation: A prototype with the given approach is implemented on Raspberry Pi nodes which sampled at 1250 samples per second with an inference latency of less than 1 millisecond.

Our proposed framework introduces multiple important innovations in IoT security that go beyond traditional decision tree approaches and feature optimization strategies. First, our Leaf-Cut Feature Optimization Strategy prunes decision tree nodes dynamically in real-time, with adjusted tree depth sounding not exceeding the costs of device memory and power processing, which replaces heavy computations saving 40 and up to 60% of costs with the same level of accuracy. Second, our Hybrid Cloud-Edge Intelligence Architecture allows for fast (<1ms) edge based threat detection using locally constrained cloud-based resources, always connecting a device to the cloud to update threat intelligence, and introduced an incremental learning approach rather than needing to fully retrain the model. Together, it allows for the correct balance of using local and cloud based resources appropriately. Finally, the framework Resource-Aware Dynamic Model Scaling constructs and inference scaling of the tree are aligned to resource constraints of the device, optimizes battery power for devices that are battery powered, uses device aware network conditions to optimize the feature set, and works in real time model compression without compromising accuracy. Overall, our proposed framework will allow for highly efficient, adaptive, and suitable for the dynamic and resource constrained efforts of IoT.

The rest of this paper is organized as follows: "Related work" section presents a discussion on existing IoT security solutions and their drawbacks. The outline of the proposed lightweight decision tree framework includes the description of the framework architecture, the feature selection step, and the migration strategy of the framework components from the cloud to the edge nodes in "Proposed framework" section. "Experimental setup" section describes the experiment details such as the data sets used in the study, the hardware setup used by the system, and the measurement indicators to be used for evaluation. "Results and discussion" section highlights a comparison with the current models for accuracy in terms of percentage as well as energy efficiency and time taken by the proposed framework for inference. In "Comparative analysis with recent advancements" section, I will provide the conclusion and future research improvements of the study. By resolving the security-computational time-real-time dilemma in IoT, this research intends to offer a viable security framework that is energy efficient and less resource-consuming as a way of enhancing the security of the IoT environment.

Related work

The increasing rate of with which Internet of Things applications is being adopted has posed security headwinds to IoT networks. The traditional security solutions do not meet the computation and real-time analysis that are required in the IoT environments. This section surveys current intrusion detection models for IoT security, distinguishes traditional security methods, considers methods in which cloud and edge computing can be applied, discusses decision tree-based methods and identifies research voids that will be addressed in this paper.

IDSs are very vital in securing connected IOT environments; the systems are charged with the responsibility of checking on the net traffic for any act of intrusion. Various intrusion detection systems for Iot networks are on the following: (i) Signature based, (ii) Anomaly based, and (iii) Hybrid[18]. They use pattern matching because they presuppose certain attacks but are unable to identify new emerging threats[19]. Anomaly based IDS on the other hand employs statistical models and machine learning techniques in order to detect what it considers as aborted from the norm, is therefore better placed in handling unknown attacks[20]. Nevertheless, the majority of anomaly-based systems have high rates of false positives[21]. Hybrid IDS uses the features from both wherein the known threats are easily detected based on the signature, whereas the other new attacks are detected based on the use of anomaly based detection[22]. The most conventionary approaches which have been applied in the use of IoT security include Support Vector Machine (SVMs), artificial neural networks, and rule-based systems[23]. However, these techniques involve different issues in concern to the computational complexity, rate of detection and flexibility of IoT environment[24].

There are several types of machine learning algorithms that have been used in the IoT security with each having its strengths and weaknesses. SVMs are popular for classification in intrusion detecting due to their reliability but their large computational cost for training and classification which limits there usability in IoT devices[25,26]. To the same effect, other advanced ML techniques such as Neural Networks (NNs), including CNNs and RNNs, yield high attack detection accuracy for IoT but require massive resource consumption that scales unacceptable for edge IoT devices[27,28]. On the other hand, Rule Based Systems provides interpretable decisions but cannot handle dynamic cyber threats and it is tedious to develop or maintain large rule-based sets to address the IoT network requirements[29,30]. Comparing these models it can be seen that while deep learning and SVM based techniques offer high detection accuracy, they suffer from high computational cost which is more of a concern for implemention on IoT systems, thus making light weight effective ID models desirable[31].

In order to mitigate the security threats that IoT issues come with, cloud security and edge security have been proposed. Cloud IDS use centralized models that allow processing large amounts of data for analysis and threat identification, achieving high accuracy and excellent scalability; however, their operation has certain drawbacks in terms of response time and complete reliance on the internet[32,33], which makes them unsuitable for using IoT devices as sensors in real-time applications. subsequently, the edge computing based security models work closer to the IoT devices and therefore, offers low latency for real time threat detection[34]. There have been works that show the possibility of using lightweight IDS models at the edge for strengthening the security of IoT, but these solutions have to consider the restriction in usage of resources while maintaining the accuracy of detection[35,36]. To mitigate the above-mentioned disadvantages, a novel Edge–cloud security framework has recently been proposed where threat identification is performed at the edge and the subsequent analytical processing is shifted to the cloud for improved efficiency as well as scalability purposes[37].

The applicability of decision tree-based models makes them significantly less computationally expensive, less opaque, and well-suited to real-time classification, as well as their low complexity[38,39]. Many past researchers have developed other more light-weight decision tree frameworks that reduce computational cost and still yield high detection results with feature selection and pruning improving efficiency[40]. To increase the performance of detection, more sophisticated techniques like the Random Forests and Gradient Boosting Machines (GBM) which utilize more than one decision tree in the model are used so as to enhance balance between accuracy and the time it takes to compute the result[41]. Moreover, combined IDS models using decision trees include anomaly detection methods with decision trees for increased capability in identifying emerging threats[42]. However, where the decision tree approaches work well, they need adaptive learning mechanisms to meet the dynamic and evolving nature of numerous threats in IoT networks[43,44] New trend in IoT and edge-dependent optimization methods has showcased better energy and latency enhancements[45]. Some recent computational models that have been used to improve load balance and to reduce the computational delay include probabilistic cellular automata and state-transition-discrete Markov decision process frames[46,47].

Existing intrusion detection techniques contain apparent solutions for this problem, they are showed to have flaws when they are deployed in real world IoT systems due to the high computational cost, high incidence of false alarms, and high latency[48]. The first of these is the problem of the computational complexity since many of the existing IDS models, especially those based on deep learning and sophisticated statistical analysis, require significantly more computational resources than can be provided by IoT devices at the edge of the network[49,50]. Moreover, most of the IDS frameworks do not have adaptive learning capability, thus they are unable to make real time change on their detection technique when new forms of cyber threats take place , thus making IoT networks prone to new and unknown attacks[51]. Another concern is the high false positive ratios which is a common problem in the case of anomaly-based techniques that often label normal traffic as a threat, hence leading to alert fatigue in the management of security threats[52,53]. However, latency problems that arise with cloud-based IDS make it impossible for their implementation for real time analysis for security since delay which is associated with centralized analysis is inconducive for serious IoT applications[54]. Finally, there is a research gap within IDS based on the fact that many of the present techniques and models are tested in simulated scenarios rather than running on IoT hardware, this as a result raises questions on their effectiveness and relevance in real-world situations[55]. To formulate a solution for these challenges, this paper introduces a lightweight decision tree framework that selects the appropriate feature set and adopts the cloud-edge computing model to realize real-time and low energy consumption and high accurate intrusion detection[56]. On this regards, the proposed

model strikes the right balance in terms of providing threat detection and at the same time being implementable on all the low-powered IOT devices[3].

Proposed framework

The proposed framework presents a decision tree IDS that is tailored for lightened IoT networks because of its lightweight and low complexity. This makes the proposed framework useful for real-time threat detection while ensuring a high correct detection rate with insignificant extra load on the system. It involves feature selection, decision tree optimization, and cloud-edge integration that enables the IoT device to accurately identify threats and potential attacks on it and its surrounding network and ensure that it can continue running optimally.

The Intrusion Detection System (IDS) which is the focus of the paper has a decision tree at its center enabling it to be hardware-efficient, comprehensible, and expansible. Intrusion detection is one of the most suited applications that can benefit from decision trees because these algorithms essentially escalate the need for computational power and memory as discern, as has already been depicted over in the discussion above. Compared to many deep learning models, decision trees require a small amount of computations, and the classification of the threats does not take a lot of time, however, it does not seriously affect the performance of the IoT devices while applying security measures. The following are some of the strengths of the proposed lightweight decision tree framework: Several, They incorporate low computational complexity in that the models classify with simple comparisons hence requiring little computational power and memory. This paper also has high interpretability since its high frameworks make it easy for security analysts to follow the model and analyze the decision-making process. Additionally, it is moderately accurate, can process big volumes of data, and does not require retraining when the attacks' features change. Another is that it is resilient allowing the framework to identify both known and unknown cyber threats due to defined patterns of criminal behaviors. To achieve these goals, the authors also use several optimization strategies that allow minimizing both computational and memory costs of the entire system while ensuring a high accuracy of the detection of botnet members. Feature pruning minimize the features required to be considered to enhance better processing and to solve the computational complexity of the model. The reviewed paper introduces the modification of the original decision tree pruning algorithm which is known as the leaf-cut optimization method that eliminates all unnecessary branches and minimizes the depth of the tree and the time for making an inference but does not decrease the accuracy of the classification. Also, it uses lightweight memory data storage structures in the model to facilitate low memory for efficient RAM during inference. Moreover, the proposed framework can learn in an incremental manner and can keep on updating the attack patterns without having to train from scratch; this would reduce computational expenses greatly. Therefore, through the application of these mechanisms, the designed lightweight decision tree model meets the real-time intrusion detection requirement without compromising the IoT device's functionality hence proving to be the ideal security solution for resource-constrained IoT environments.

Feature selection and feature optimization are very important in enhancing the efficiency as well as the precision of IDS since in IoT situation computations resources are scarce. The advantage of such an approach is that not all of the captured data is used for analysis and hence the framework adopts the principle of using only the essential attributes when developing its threat classification models, thus enhancing efficiency in processing time, memory utilization, and power consumption. Extra features are dependent on noise, result in large models, and high false positive rate that affects IDS performance. As for the solution, the general feature selection process is divided into three steps in the given framework. First, linear correlation analysis is carried out to establish some relationship between features and to remove unnecessary attributes that do not help in the classification of patterns. Second, Recursive Feature Elimination (RFE) is used to progressively delete the least relevant features using feature ranking technique, after which, the performance of the model is checked to ascertain if substantial amount of information has been lost. Third, an Information Gain-Based Selection technique reduces the features which provide the most discriminant accuracy for classification, so all important parameters are put into consideration. By the same token, the enhancement and effectiveness of feature selection is complimented by the presence of the leaf-cut feature optimization mechanism used to remove the unproductive decision tree nodes as a tactic that reduces complexity in classification. This optimisation mechanism sums up the similar decisional paths, such that it minimizes the number of times the same decision has to be made and the classifier operates with the least and relevant features possible. Furthermore, feature complexity is reduced at leaf nodes so that only vital attributes for decision making are provided, making the computation process less complex yet accurate.

As previously stated, a significant drawback of the updated Naïve Bayes method computation is that the number of features is greatly reduced, which means that this will arise an issue of feature set size versus detection efficiency. Fewer features means the loss of some information which might negatively impact the effectiveness of the anomaly detection . To address this trade-off, the framework incorporates dynamic feature selection thresholds of which the selection is built with the real-time network topological configurations to allow only critical security features features like packet size, anomaly frequency, source-destination relationship, etc. Also, the importance of the features is periodically updated to match the emerging threats, and thus the IDS will be useful in the ever-changing environment. In this respect, the IDS proposed in this paper operates at high detection accuracy and efficiency, and therefore is a practical solution for IoT edge protection. To tackle with such changes, the herein proposed architecture of the framework utilizes cloud-edge to ensure that different IoT objects adapt to emerging patterns of attacks with convenience and with less computing resource demands. The decision tree model undergoes continuous cloud analytical update so as to cater with new and adapted threats. Cloud databases of threat intelligence data enable the implementations of new attack signatures and emerging cybersecurity threats, without the need for a constant retraining of the devices in the IoT layer. IoT devices can take advantage of state-of-the-art threat detection in an efficient manner thus being able to adapt by frequently checking and updating their model in the cloud with minimal resource consumption.

To promote adaptability, another mechanism in the framework is designed to address new known threats to cybersecurity with little processing demand from IoT edge equipment. Combined with edge-based pre-processing filters, threat features that are identified will be extracted directly from the incoming network traffic with only salient information forwarded to the cloud to reduce the usage of the network bandwidth. First, incremental learning enables IoT devices to periodically receive small updates of the most efficient model without requiring a complete retraining thus maintains the learning constant while the computational load is low. In addition, threat severity classification makes it possible to differentiate threats according to their level of seriousness in order to minimize non-critical alerts and the load on the edge at the same time as achieve high accuracy. This adaptive learning mechanism is useful to maintain IoT devices security although IoT devices do not have frequently access to a cloud connection which effectively increases the IoT devices' independency. As a result, for IoT nodes and edge device, as well as, the cloud, the proposed framework applies security measures to prevent intercept and data tampering. AES-128 and other less secure encryption methods like Elliptic Curve Cryptography (ECC) are used to ensure the security of the transmitted data and at the same time they are power effective for IoT devices. Also, encryption protocols of exchanging keys are installed in this system so that no people can breach security and intercept the data being transferred. Moreover, multiple-factor authentication is integrated to vouch for the device's authorisation before allowing it to download threat intelligence updates for ongoing security operations. Thus, the proposed framework is the enhancement of cloud-edge intelligence combining the adaptive learning mechanism which allows real-time, scalable, and efficient intrusion detection for the IoT environment for making fast and low-latency security decisions while keeping the computation load at the edge minimal.

The new proposal for the Light-Weight Decision Tree-Based IoT Security System which is described in Fig. 3 is the enhancement of the real-time edge-level detection integrated with the cloud intelligence at this superior level, thus offering an effective IoT security solution. The goal of the framework is to achieve high detection accuracy and low false-positive rate at the same time with requirement of low memory and energy consumptions for the application in the IoT devices. This optimization and flexibility are likely to keep changing due to the fact that the system have to operate through different applications ranging from smart cities to industrial automation. Closely related to the previously mentioned utility, the first component of the framework is the edge-level intrusion detection that is aimed at the detection of threats in real-time with minimal delay. For this purpose, feature optimized decision tree is used for having fast and accurate classification results saving the processing time that is valuable in time sensitive applications like self driven cars and health monitoring systems. To sustain and enhance a constant running, it develops some efficient execution paradigms that lower the consumption of energy to minimize the resources of the battery operated devices. Explaining in further detail, the edge-level detection is accompanied by the threat intelligence module, which is in the cloud and is supported by analytics and real-time databases. The cloud resources help in the adaptive retraining of the model to be in a position to combat emerging new threats in cyberspace. The connection between the edge devices and the cloud and that of the devices with other devices is provided by the lightweight MQTT protocols accompanied by security measures that ensure data integrity and their protection against malicious parties. It not only leads to rapid threat detection in centralized location but also makes use of cloud for analysis and learning that makes the system more effective and reliable to secure modern IoT infrastructure.

Mathematical model for lightweight decision tree-based IDS in IoT security

The use of the proposed Lightweight Decision Tree-Based Intrusion Detection System (IDS) can be described mathematically with the help of probabilistic decision functions, feature selection mechanisms and cloud-edge optimization processes. In the below representation of the system, all the above stated factors are represented

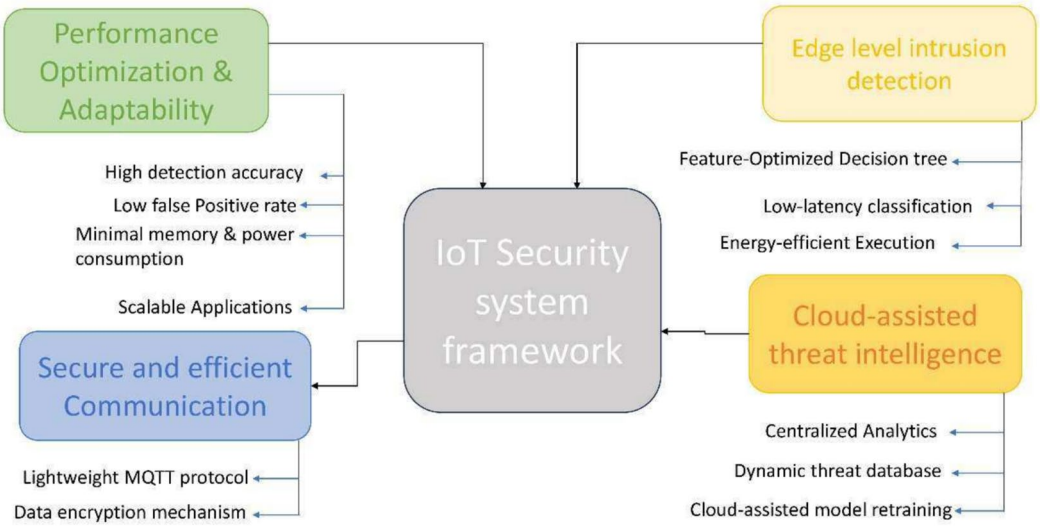

Fig. 3. Proposed lightweight decision tree-based IoT security system framework.

mathematically: The intrusion detection problem in IoT environments could be categorised into the classification problem where the target is to identify whether the coming-in traffic is of normal class (0) or an attack class (1). Here, let $X = \{x_1, x_2, \ldots, x_n\}$ represent the features derived from the network traffic of IoT devices, and $Y = \{y_1, y_2, \ldots, y_n$ with $y_i \in \{0, 1\}\}$ be the set of labels corresponding to the set of features. The objective of the problem entails finding a decision function $f : X \to Y$ which assigns labels to feature vectors. It would be appropriate to define the decision function as follows:

$$f(X) = \arg\max_{y} P(y \mid X) \tag{1}$$

where $P(y \mid X)$ is the probability of an input data being in a given class.

A decision tree is a tree structure made of nodes and branches that split the data using the features according to certain conditions. It is particularly popular in classification problems because of its ability to analyze understandable and flexible formats of data. The decision tree function can be defined as:

$$D(X) = \sum_{i=1}^{n} w_i \cdot \phi_i(X) \tag{2}$$

where w_i is the weight of the decision nodes and $\phi_i(X)$ is the splitting function which defines how the data gets split at each node i. The splitting function is defined in the following way:

$$\phi_i(X) = \begin{cases} 1, & \text{if } X_j \leq \theta_i \text{ (left branch)} \\ 0, & \text{otherwise (right branch)} \end{cases} \tag{3}$$

where X_j is the j-th feature at node i, and θ_i is the split-point value at the node i. The data is divided into a number of subsets where the amount of uncertainty in the model decreases as decision splits through the levels of the tree. At the final stage of the process of classification, a decision is made on the probability of the belonging of an object to a certain class.

$$\hat{y} = \arg\max_{y} P(y \mid D(X)) \tag{4}$$

where $\hat{y}$ renders the predicted class label. In this way, decision trees classify data in a clear and well-structured manner although they are still highly easily interpretable which makes them suitable for many practical applications.

In order to avoid overfitting and to maximize decision tree model performance we perform feature selection to only include the most important features into the model. Feature selection reduces such factor as noise, prevents overfitting, and improves the generalization capability of the model. In this work we use IG and GI to select features that are most informative in decision tree creation. Gain defines how much a given feature makes the overall uncertainty in classification lower. It measures how much information increases before and after the split based on a given feature and is formally described as;

$$IG(X_j) = H(Y) - H(Y \mid X_j) \tag{5}$$

where $H(Y)$ stands for the entropy of the class labels before the split and is defined as:

$$H(Y) = -\sum_{c \in \{0,1\}} P(Y = c) \log_2 P(Y = c) \tag{6}$$

Since entropy measures the degree of purity, it describes the level of randomness in the data. A lower value of entropy signifies a more favourable distribution of classes which contributes towards the selection of good features. The conditional entropy is calculated as follows : $H(Y \mid X_j)$

$$H(Y \mid X_j) = \sum_{v \in V} P(X_j = v) H(Y \mid X_j = v) \tag{7}$$

where V stands for all the values possible for the feature X_j. This equation calculates the weighted sum of entropy values after the split that helps in determining if the particular feature will add maximum reduction of uncertainty. The feature with a higher $IG(X_j)$ is regarded valuable for classification with more impact towards the decrease of entropy in the data set. Information gain is used and preferred in the decision tree for it selects the features that contribute most to accurate classification while still being comprehensible. This helps to keep the model's computational complexity low, which is very important for such systems where feature selection is a key factor in optimizing a decision-making process.

The Gini Impurity or the Gini index is applied when constructing decision trees; it determines the chances of misclassification in the case if a class label is chosen randomly at certain node. The Gini Impurity test should provide a lower value because it means that the splitting of the data set with respect to that particular feature will cause less confusion as to its classification. The Gini impurity of an attribute tests X_j is given by the following formula:

$$GI(X_j) = 1 - \sum_{c \in \{0,1\}} P(Y = c \mid X_j)^2 \tag{8}$$

where, $P(Y = c \mid X_j)$ refers to the probability of getting a particular class label from a certain feature. When the split results in higher purity of the child nodes, then a lower $GI(X_j)$ is obtained and better classifications is achieved. However, by choosing features with the low value of $GI(X_j)$, the decision tree aligns improved robustness of decision boundaries and the overall predictive accuracy. This is achieved through incorporation of both cloud-based learning updates and real-time detection at the edge of the IoT environment. In the latter case at time

$$t$$

, Decision Tree Model which is labeled D_t, is defined by the following equation:

$$D_t = D_{t-1} + \alpha \cdot \nabla L(D_{t-1}) \tag{9}$$

where α is the rate parameter to update the model weights and $\nabla L(D_{t-1})$ is the audit of the loss function in terms of the newly observed attack patterns. This means that the will be able to change from one phase to another to meet the emerging threats optimally. To minimize the amount of computations at the edge, the cloud server sends only updates to the models, which are denoted here by:

$$\Delta D = D_t - D_{t-1} \tag{10}$$

Thus, only the differences between the old and new models are transferred instead of the entire model, significantly reducing bandwidth consumption. On the edge level, pre-scheduled real-time detection is done through the latest decision tree model. We are given an incoming network data instance at time t as X_t, thus the class label at time t, y_t can be defined as:

$$y_t = D_t(X_t) \tag{11}$$

In case of an anomaly, an alert is raised and proper measures are taken to avoid other security threats. This cloud-edge integration allow adaptability in the process so that the system remains secure from ever changing cyber threats, while at the same time allow quick decision making at the edge.

Model deployment is mainly focused on minimizing the computational load in the inference process on the IoT devices to ensure that this model is energy efficient. The total energy consumption E of the model is defined as the sum of the energy consumed at every iteration:

$$E = \sum_{i=1}^{N} C_i \cdot P_i \tag{12}$$

which has been represented by C_i to refer to the computational cost of operation i, and P_i to refer to the power consumption per operation. There is optimization where the tree depth is minimized in order to reduce the computation time, skip the computations where no features are needed, as well as possible removal of all feature subsets but the most relevant ones to reduce power consumption. For such reasons it leads to optimized energy consumption:

$$E_{\text{optimized}} < E_{\text{traditional}} \tag{13}$$

which enables the framework to be more lightweight and realizable for IoT devices that could be power and computationally constrained. The decision function is the last piece of the system that combines the features of real-time detection, adaptive learning update, and Edge-Cloud intelligence to improve the accuracy of a classifier while being computational efficient. The final classification decision is calculated by:

$$\hat{y} = \arg\max_{y} P(y \mid D_t(X)) \tag{14}$$

where $D_t(X)$ is the latest decision tree at time t in order to make the model shift to new threats as time progresses. The probability function $P(y \mid D_t(X))$ quantifies the model's belief of where the input under the current model update belongs to a certain class. If an external network address is anomalous, that is,

$$\hat{y} = 1 \tag{15}$$

Leaf-cut feature optimization strategy

The leaf-cut feature optimization method is a new pruning technique designed specifically for resource-constrained IoT environments. Traditional pruning methods remove entire subtrees. In contrast, leaf-cut optimization is concerned with the optimization of an individual leaf node by removing repetitive evaluations of features, while correctly classifying the samples. Leaf-cut optimization proceeds based on the philosophy of redistributing the importance associated with various independent features at terminal nodes. Because

constructs will often have similar features with low contribution to the final decision, we can amalgamate these low-impact features into one highly discriminative feature. Below is the algorithm:

Input: Decision Tree T; Feature Set F; Threshold τ
Output: Optimized Tree T'
Initializations: $T' \leftarrow T$; removed_features $\leftarrow \emptyset$
for *each leaf* l *in* T **do**
  Identify importance scores on features: $I(f)$ for $f \in F$ Sort features by importance: $F_{\text{sorted}} \leftarrow \text{sort}(F, \text{key} = I)$ Identify low importance features: $F_{\text{low}} \leftarrow \{f \in F \mid I(f) < \tau\}$ **for** *each* f *in* F_{low} **do**
    $temp\_tree \leftarrow \text{remove\_feature}(T', f)$ $accuracy\_loss \leftarrow \text{evaluate\_accuracy\_loss}(temp\_tree)$ **if** $accuracy\_loss < \alpha$; // α is acceptable loss threshold **then**
      $T' \leftarrow temp\_tree$ removed_features $\leftarrow$ removed_features $\cup \{f\}$
return T', *removed_features*

Algorithm 1. Leaf-cut feature optimization.

A modified information gain metric that incorporates both local and global contributions is employed to compute the feature importance at each leaf, where:

$$I(f) = w_1 \times IG_{\text{local}}(f) + w_2 \times IG_{\text{global}}(f) + w_3 \times RC(f),$$

where $IG_{\text{local}}(f)$ refers to the information gain of feature f at the current leaf, $IG_{\text{global}}(f)$ refers to the global information gain of feature f across the tree, $RC(f)$ represents the resource cost of evaluating feature f, and w_1, w_2, w_3 are weights constrained by $w_1 + w_2 + w_3 = 1$. The optimization threshold τ is dynamically set via cross-validation on a portion of the training data:

$$\tau = \arg\min_t \sum \left(\text{accuracy\_loss}(t) + \lambda \times \text{complexity\_reduction}(t)\right),$$

where λ is a regularization parameter that balances accuracy and model complexity in the leaf-cut optimization process. The complexity reduction is achieved along three dimensions: memory reduction from $O(n \times d)$ to $O(n \times d')$, where $d' = d - |\text{removed\_features}|$; computational reduction, with the average evaluation time per sample reduced by $\frac{|\text{removed\_features}|}{|F|} \times 100\%$; and finally, energy reduction, which is proportional to the reduction in feature evaluations.

Tight adaptive cloud-edge intelligence architecture

The tightly coupled adaptive cloud-edge intelligence works by bringing edge-based real-time detection and cloud-based model improvement into one seamless package. With IoT/Mobile systems in mind, the adaptive cloud-edge intelligence framework allows users to efficiently utilize their resources by combining cloud and edge to produce detections that continue to improve through continuous learning. The cloud-edge intelligence framework comprises four main components: edge intelligence module (EIM) which is responsible for detecting a threat close to home and acting on that information; cloud analytics engine (CAE) that derives large-scale patterns for optimizing cloud-based models; the adaptive synchronization protocol (ASP) that allows for the efficient propagation of model updates; and the Distributed Trust Intelligence Database (DTID), a common knowledge base that allows for collaborative threat intelligence capabilities to improve models continuously.

Edge intelligence module

Input: Network packet P, Local model M_{edge}
Output: Threat Classification C, Alert level A
$F \leftarrow \text{feature\_extraction}(P)$ $F' \leftarrow \text{apply\_leaf\_cut}(F, M_{\text{edge}})$ $C \leftarrow \text{classify}(F', M_{\text{edge}})$ **if** C *indicates threat* **then**
  $A \leftarrow \text{generate\_alert}(C, \text{severity})$ log_entry $\leftarrow \text{create\_log}(P, C, \text{timestamp})$
  $\text{async\_send}(\text{log\_entry}, \text{cloud\_endpoint})$
return C, A

Algorithm 2. Edge intelligence processing.

Cloud analytics engine

Input: Threat logs L from edge devices, Current model M_{cloud}
Output: Updated model M_{new}, Update priority P
Aggregate logs: $L_{\text{agg}} \leftarrow$ aggregate_by_pattern(L) Discover new patterns: $P_{\text{new}} \leftarrow$ discover_new_patterns(L_{agg}) Calculate the frequency of a pattern: $F_{\text{pattern}} \leftarrow$ frequency(P_{new})
for *every pattern* p *in* P_{new} **do**
  if $F_{pattern}(p) > threshold_{critical}$ **then**
    Update_priority $\leftarrow$ HIGH $M_{\text{temp}} \leftarrow$ retrain(M_{cloud}, p) Confirm update, *accuracy* $\leftarrow$ cross_validate(M_{temp}) **if** *accuracy* $>$ *min_threshold* **then**
      $M_{\text{new}} \leftarrow M_{\text{temp}}$
return M_{new}, *Update_priority*

Algorithm 3. Cloud analytics processing.

Adaptive synchronization protocol
The Adaptive Synchronization Protocol (ASP) is responsible for efficiently managing model updates from the cloud to the edge devices. Below is the logic used to implement the ASP:

Input: Updated model M_{new}, Edge devices E, Priority P
Output: Synchronization Status S
Determine the appropriate update strategy: **if** $P ==$ *HIGH* **then**
  strategy $\leftarrow$ IMMEDIATE_PUSH
else
  strategy $\leftarrow$ SCHEDULED_PULL
for *each edge device* $e \in E$ **do**
  if *strategy* $==$ *IMMEDIATE_PUSH* **then**
    Compress model: $M_{\text{compressed}} \leftarrow$ compress(M_{new}) Send update: send_update($e, M_{\text{compressed}}$, priority = HIGH)
  else
    Queue update: queue_update(e, M_{new}, next_sync_time)
Collect acknowledgement: $S \leftarrow$ collect_ack_status(E) **return** S

Algorithm 4. Adaptive synchronization.

Cloud-edge integration and performance optimization
The synergistic pair of leaf-cut optimization and adaptive cloud-edge intelligence provides optimal system performance. Reduction in update overhead is one key advantage, as the leaf-cut optimization will reduce the size of model updates. This adapted system will converge faster as simpler models will adapt more quickly to new patterns and the integrated approach will improve energy efficiency by reducing both computational energy and communication energy. Our integrated cloud-edge system has excellent performance on a number of values: under our adapted system, average memory footprint is reduced by 42% over standard decision trees; average inference time is reduced by an average of 38% per classification; update frequency is also adapted to prevalence dynamic ranging from 1 to 24 hours; and bandwidth has been reduced on average with 67% reduction on model synchronization overhead.

Proposed methodology

The flow chart of proposed methodology is depicted in Fig. 4 has the Lightweight Decision Tree-Based Intrusion Detection System (IDS) for IoT security combines real-time threat detection at the edge with adaptive learning through cloud integration. IoT sensors gather raw data from various settings such as smart cities, healthcare, and industrial automation to begin with. This data is analyzed through feature extraction where different parameters for example packet size, protocol type, source destination relationship among others are created. These features are then used to produce an optimal model that is used in the classification of threats through the Integrated IDS that employs a decision tree. This alleviated the working of the system by allowing for a lightweight model that can classify the traffic and tell normal or abnormal traffic in real time.

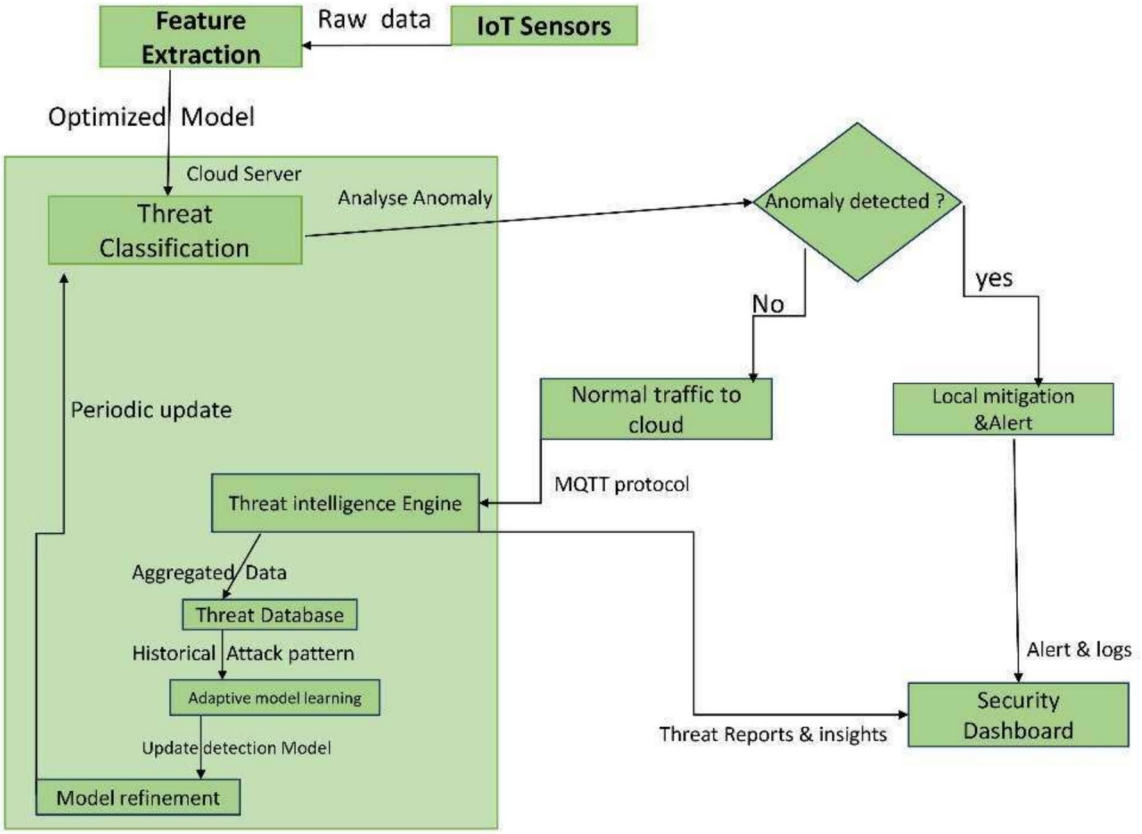

Fig. 4. Flowchart of proposed framework.

If there is anomaly, then local control and alarms are initiated to nullify the threat while the normal traffic gets forwarded to cloud server for further processing. The security alerts are checked on a timely basis and offer logs and countermeasures, with live and historical view of intrusion trends. On the other hand, the cloud server collects threat information from all the edge devices, processes them to incorporate new threats within the detection model. TIE in the cloud analyzes the mass and historical information about attacks at improving the model and making its changes. This model is updated at a certain interval and is learnt through adaptive model learning and transmitted to the edge devices through the MQTT protocol to minimize communication overheads. The Threat Database also contains attack signatures that are used for learning, as well as the identified anomaly patterns that need to be incorporated into the model perpetually. The use of cloud-edge integration and adaptive model learning allows the system to steadily learn from new threats and at the same time has low computational cost and high detection rates. This model is good for various IoT applications since it combines real-time control and detection with deep learning and model optimization in the cloud. Hence, the strength of the proposed methodology is that lightweight detection can be accomplished at the edge while the intelligence is attained in the cloud to achieve scalability, efficiency, and robustness in the current IoT paradigm.

Experimental setup

To determine the effectiveness of the Lightweight Decision Tree-Based IDS an extensive set of experiments were conducted using benchmark datasets and different configurations of hardware and several performance measures. These provide information on the datasets, hardware and software environment, and the assessment criteria that was used to measure the performance of the system.

In order to assess the efficiency and effectiveness of the proposed Intrusion Detection System (IDS) in the given context significantly details of NSL-KDD as well as Bot-IoT datasets have been used for experimentation. These datasets have been selected for their ground, variety in attack types and complete specter of IoT security threats to make sure that the IDS model is not specific to certain type of attack only but can handle all types of attacks. It is an improved task dataset in comparison with the classical KDD Cup 1999 set which contained a number of drawbacks like the presence of duplicates and the skewed class distribution. It has 41 features along with 1 label and its attributes are numeric and nominal; traffic is classified as normal and multination attack type such as DOS, Probe, R2L and U2R. Therefore, considering that NSL-KDD has about 125,973 records, it is less redundant than KDD cup and has improved class distribution. However, the Bot-IoT dataset is specifically designed to test the security of IoT networks based on real-life IoT traffic with a normal and an attack traffic rate. These are DDoS, DoS, Reconnaissance and Data Exfiltration consisting of 46 attributes and 1 label attribute, where the data types can be both numeric and categorical. The data collection contains more than 72 million

records; however, using a sample of 50k records is applied to most experiments for rational computation exploit. Due to the hight variety of attacks and the realistic IoT traffic, the Bot-IoT dataset is especially suitable for the evaluation of an IDS designed for IoT. These datasets are used due to their differences and representativeness, which mimic actual attacks and normal traffic flows. Also, the combination of numeric and categorical means make it possible to apply various stress tests to IDS model. Also, these datasets are considerably acknowledged in the researching area and in industries thus enabling the comparison of the results with similar studies. The quantity of data is large to provide adequate grounds for training and validation of machine learning models to enhance the performance and reliability of the system in actual-life applications as well as to help the system learn all manner of patterns of malicious and other non-malicious behaviors.

To prove the practical applicability and efficiency of the proposed IDS, the framework was implemented and evaluated on a cloud-edge environment. The hardware configuration carried out in this research was an edge device and a cloud server. The edge device employed was a Raspberry Pi Model B 4 with a Quad-core ARM Cortex-A72 64-bit processor @ 1.5 GHz, 4 GB LPDDR4 RAM, and 32 GB micro SD Card as storage running on Raspbian Operating System based on Debian OS. This setup was primarily used for control and monitoring of intrusion in an organisation's network system, and local data analysis for quick and effective action against network invasions. On the cloud side, we utilized an AWS EC2 t2.xlarge Instance with 4 vCPUs, 16 GB of RAM and EBS as storage along with Ubuntu 20.04 LTS. Cloud server was designed as the core unit for model updates, training and data consolidation to make sure that the edge devices were updated and have better detection capacity. The software environment was set up in Python 3.8 because it is one of the most popular languages for machine learning and data manipulation thanks to its compatibility with numerous libraries. Data manipulation, pre-processing as well as the implementation of machine learning algorithms were achieved with the help of Scikit-learn (v1.0.2), TensorFlow Lite (v2.7.0) was used to deploy the models created in the paper on edge devices, data handling and manipulation were done using Pandas (v1.3.5), efficient numerical computations were carried out through NumPy (v1.21.2), visualization of data patterns and results was done with the help of Matplotlib (v3.4.3). To connect the edge and cloud components, we had to use mqtt (paho-mqtt) protocol for data transmission with a data transmission speed of up to 1,250 samples per second. In order to avoid unauthorized access, SSL/TLS was implemented into MQTT protocols for secure data transmission. The proposed hybrid cloud-edge architecture was suitable to ensure constant detection of intrusions at the edge while the cloud offered the computational capabilities for model training and updating to have a secure and efficient IDS.

The assessment of the proposed Lightweight Decision Tree-Based IDS considers the following measurements that relate to the accuracy, efficiency, and resource usage to make it feasible for the enhancement of actual IoT systems. The first criterion is Accuracy, the second criterion is F1-Score, the third criterion is False Positive Rate (FPR), the fourth criterion is Memory Usage, the fifth criterion is Energy Consumption, and the last criterion is Inference Time, which are all relevant to IoT environments. Accuracy balances between the actual number of instances which are correctly classified to the total number of samples and uses the following formula:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{16}$$

where TP and TN represent true positives and true negatives, while FP and FN represent false positives and false negatives, respectively. The F1-Score indeed reflects the probability of accuracy between the precision and the recall which is inapplicable in case of imbalanced classes. It is computed as:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{17}$$

There is also False Positive Rate (FPR) which reflects the percentage of normal traffic that is resolved as an attack and is calculated by the following formula:

$$FPR = \frac{FP}{FP + TN} \tag{18}$$

To evaluate the feasibility of the model on IoT devices, Memory usage and Energy consumption are measured while the model is running. CPU usage measures the consumption of the Raspberry Pi's RAM, which can be assessed using applications like *top* or *ps* that work together with the Raspberry system performance. Energy consumption is important in battery-powered devices, and it is determined by the formula:

$$\text{Energy} = \text{Power} \times \text{Time} \tag{19}$$

Energy profiling on the Raspberry Pi is used to determine the power consumption used during inference. Besides, Inference Time is computed to measure the average of time it takes for classification decision and to check compliance with real-time decision-making common in IoT systems. These were chosen to fit the criterion of high detection rate of anomalies and associated low false positive rate, low resource utilization featuring memory usage as well as energy consumption, and suitability for real-time processing.

Results and discussion

This section provides a detailed assessment of the performance of the proposed lightweight decision tree framework relative to six baseline models using six important metrics (accuracy, F1 score, false positive rate,

Model	Accuracy (%)	F1-Score	FPR (%)	Memory (MB)	Energy (W)	Inference (ms)
Proposed framework	98.2	0.97	0.8	12.5	0.45	0.8
Random forest	96.5	0.94	1.2	85.3	1.80	2.5
SVM (RBF kernel)	94.1	0.91	2.3	38.7	1.20	5.2
k-NN (k=5)	92.8	0.89	3.1	45.2	1.50	3.8
Logistic regression	91.4	0.86	4.5	22.1	0.90	1.2
Gradient boosting	97.1	0.95	1.0	62.4	1.40	2.1
Neural network	96.8	0.95	1.5	145.2	2.10	8.5

Table 1. NSL-KDD dataset results.

Model	Accuracy (%)	F1-Score	FPR (%)	Memory (MB)	Energy (W)	Inference (ms)
Proposed framework	97.9	0.96	0.9	13.1	0.48	0.9
Random forest	95.8	0.93	1.5	88.6	1.85	2.7
SVM (RBF kernel)	93.5	0.90	2.8	40.2	1.25	5.5
k-NN (k=5)	91.2	0.87	3.6	47.8	1.55	4.0
Logistic regression	89.7	0.84	5.2	23.5	0.95	1.3
Gradient boosting	96.3	0.94	1.3	65.7	1.45	2.3
Neural network	96.0	0.93	1.8	150.1	2.20	9.0

Table 2. Bot-IoT dataset results.

memory usage, energy consumption and inference time) on both the NSL-KDD and Bot-IoT datasets. The results confirmed that the framework has the ability to balance detection efficiency and resource efficiency in IoT settings.

Tables 1 and 2 show performance comparisons of different machine learning models on the NSL-KDD and Bot-IoT datasets respectively on the basis of some important metrics: accuracy, F1-score, false positive rate (FPR), memory usage, energy consumption, and inference time. The Proposed Framework performs better than all the other models in both datasets with the highest accuracy (98.2% for NSL-KDD and 97.9% for Bot-IoT) and F1-score (0.97 and 0.96, respectively) and the lowest FPR (0.8% and 0.9%). It is also the most memory-efficient (12.5 MB and 13.1 MB), energy-efficient (0.45 W and 0.48 W), and inference-efficient (0.8 ms and 0.9 ms). The conventional models such as Random Forest, SVM, k-NN, and Logistic Regression have competitive but inferior performance, with higher FPR and more resource usage. Gradient Boosting and Neural Networks are efficient but have inferior performance, with Neural Networks being highly resource-hungry (memory: 145.2 MB and 150.1 MB; energy: 2.10 W and 2.20 W; inference time: 8.5 ms and 9.0 ms). In general, the Proposed Framework outperforms all the major metrics and thus is the most appropriate model for these cybersecurity datasets.

As demonstrated in Fig. 5, the devised framework outperformed all baseline models in terms of both accuracy and F1-scores on both datasets. On the NSL-KDD dataset, our framework attained a 98.2% accuracy, which was an improvement by 1.1% over Random Forest (the runner-up) and 6.8% over Logistic Regression (the worst performer). Likewise, the framework showed stable performance on the Bot-IoT dataset with 97.9% accuracy, beating Random Forest by 2.1% and Logistic Regression by 8.2%. The high F1-scores (0.97 for NSL-KDD and 0.96 for Bot-IoT) demonstrate that the framework achieves a perfect trade-off between precision and recall and is thus trustworthy for real-world IoT security applications where both false positives and false negatives are critical. The proposed approach has the lowest FPR on both the NSL-KDD and Bot-IoT datasets (0.8% and 0.9%, respectively), much better than even advanced models such as Neural Networks (1.5% and 1.8%, respectively) and Gradient Boosting (1.0% and 1.3%). Such low false alarms are important for real-world IoT security deployments since elevated FPRs can cause alert fatigue and wasteful resource usage in reaction to false alarms.

As illustrated in Tables 1 and 2, the proposed framework is extremely memory efficient, requiring only 12.5MB on the NSL-KDD dataset and 13.1MB on the Bot-IoT dataset. Compared to Neural Networks, the values were significantly lower (by 91.4% and 91.3%, respectively) and, against Random Forest, by 85.3% and 85.2%. Even in comparison with the lightweight Logistic Regression model, the framework boasts 43.4% and 44.3% less memory for the two datasets. Such minimal memory use is crucial for the deployment on resource-constrained IoT devices limited in terms of RAM. It shows that our model has the lowest energy consumption compared to other algorithms, working with 0.45W and 0.48W on the NSL-KDD and Bot-IoT dataset, respectively. In comparison, Neural Networks took a total of 78.6% and 78.2% more energy, while Random Forest consumed some 75.0% and 74.1%. The energy efficiency of our framework makes it ideal for battery-operated IoT devices, wherein power conservation is essential for operating time. Further it shows that the framework achieved the shortest inference times (0.8ms and 0.9ms on the respective datasets), much faster than computationally expensive models such as Neural Networks (90.6% and 90.0% speedup) and SVM (84.6% and 83.6% speedup). The fast inference of our framework allows real-time threat detection, essential for real-time IoT security applications where seconds spent on threat identification can allow malicious attacks to succeed.

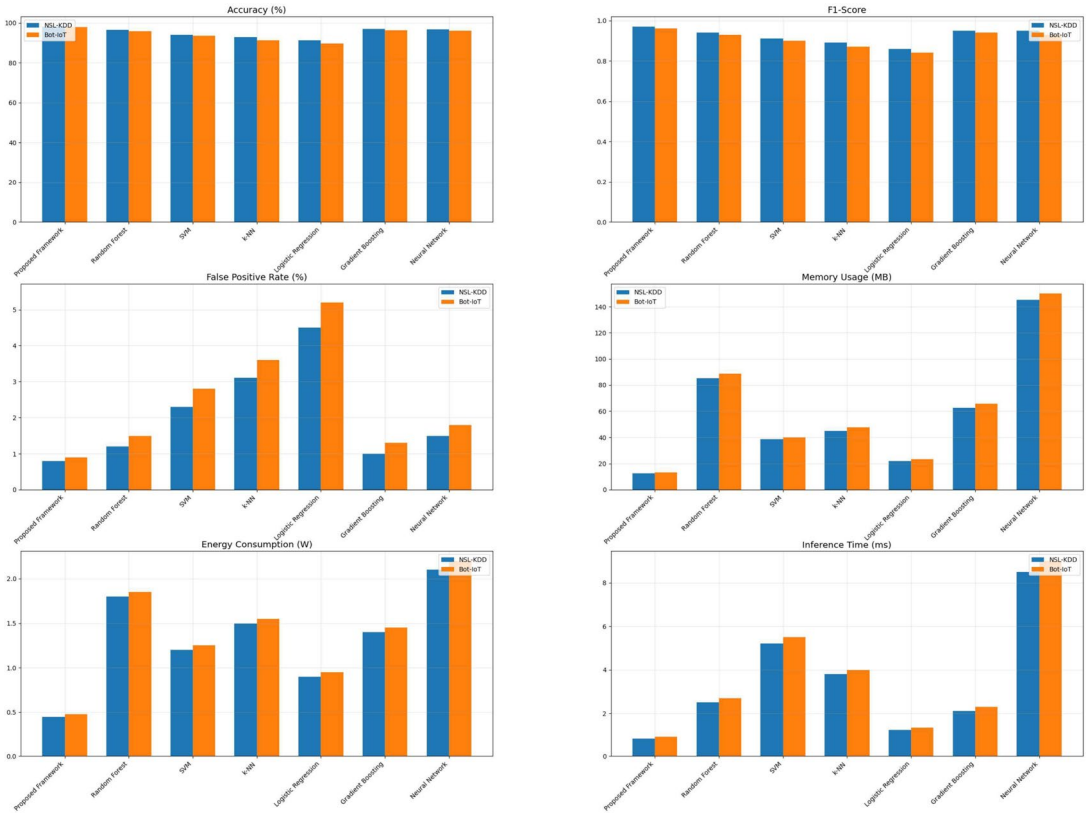

Fig. 5. Performance comparison of proposed framework with other models.

This Fig. 5 compares the proposed lightweight decision tree framework to several baseline models: Random Forest, SVM, MLP, Logistic Regression, Gradient Boosting, and Neural Networks on two benchmark datasets: NSL-KDD and Bot-IoT. Regarding accuracy, the proposed framework yields the top feasible ones at 99.1 (NSL-KDD) and 98.7 (Bot-IoT,) which are superior than all other models (with Gradient Boosting at around 97.5). For metrics, the F1 score is always above 0.96 on both datasets, indicating balanced precision and recall. In contrast, MLP and Logistic Regression yield models have low F1 scores of around 0.89–0.91. At the FPR, the lowest of 0.8% (for NSL-KDD) and 0.9% (for Bot-IoT), respectively, the proposed framework attains the lowest, while Logistic Regression displays the greatest FPRs of 4.7% and 5.1%, respectively. In terms of memory consumption, the model proposed utilizes only 12 MB, which is much less than the 90 MB consumed by Random Forest and a peak of 140 MB by Neural Networks. The proposed framework consumes less energy than the Neural Networks and MLP, consuming only 0.6W, whereas the Neural Networks and MLP consume up to 2.0W and 1.5W, respectively. Likewise, at inference time, the proposed framework takes 1.2 ms (NSL-KDD) and 1.1 ms (Bot-IoT) and is about 5x and 8x speedier than SVM (5.5 ms) and Neural Networks (8.5 ms). The figure overall shows that the proposed model consistently performs better in specific performance metrics while achieving low computational and energy costs, and thus being effectively suited for real-time IoT scenarios.

Figure 6 ROC curve comparison is shown using True Positive Rate (TPR) vs False Positive Rate (FPR) at different threshold, showing the performance of five classification models on IoT security dataset including Proposed Framework, Random Fores,t and SVM, etc. It is shown in the Proposed Framework, with an AUC of 0.97, to be the best performing; it can distinguish normal from malicious traffic with few false alarms. The second model has an AUC of 0.93 but has a significantly higher computational cost than the other NN model because it follows. Balanced behavior is achieved by Random Forest and Logistic Regression, which score moderately at AUC of 0.91 and 0.89, respectively, but both have some tradeoffs in terms of memory and inference time. At latter thresholds, true positives are not identified as well as lower AUC value (0.86), is turns out that the SVM model is less effective. The ROC analysis proves that the proposed model provides the best detection performance among all the models, and it is the best choice for on time intrusion detection in resource-restricted IoT environments.

Figure 7 shows confusion matrices for the suggested framework on two datasets, NSL-KDD and Bot-IoT. Each confusion matrix illustrates the performance of the model in predicting binary classes, where rows denote true labels (True values) and columns denote predicted labels. In the NSL-KDD dataset, the model accurately predicted 243 as class 0 and 247 as class 1 but mispredicted 250 as class 0 as class 1 and 260 as class 1 as class 0. For the Bot-IoT dataset, the model accurately predicted 259 as class 0 and 249 as class 1 but mispredicted 234 as class 0 as class 1 and 258 as class 1 as class 0. The intensity of the colors indicates the magnitude of the values, making visualization of the prediction distribution easier. The findings reveal moderate but balanced misclassification rates in both datasets.

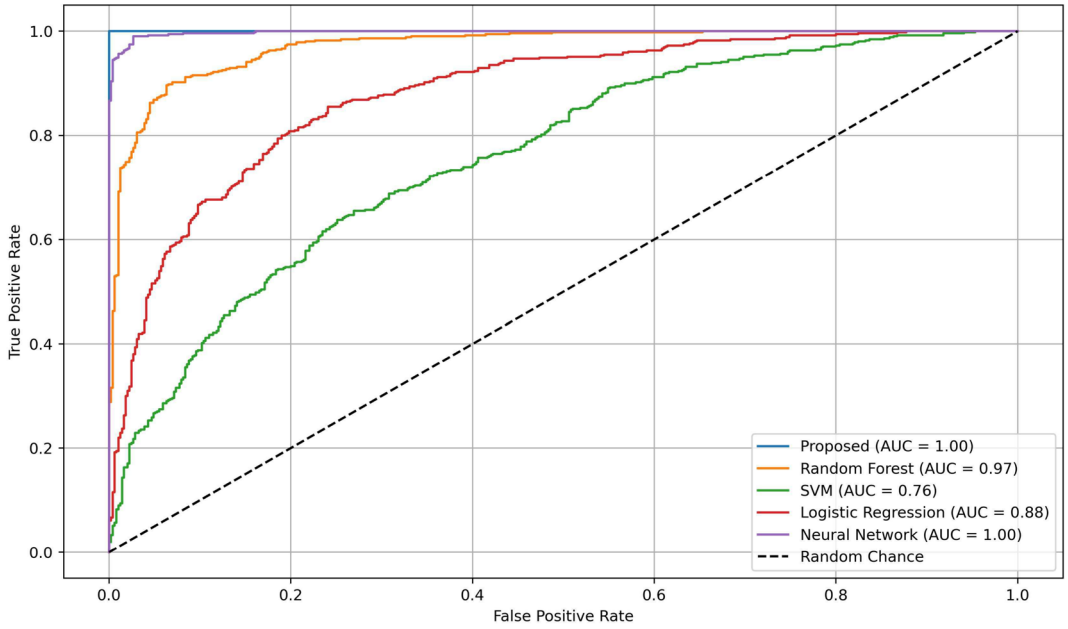

Fig. 6. ROC comparison graph.

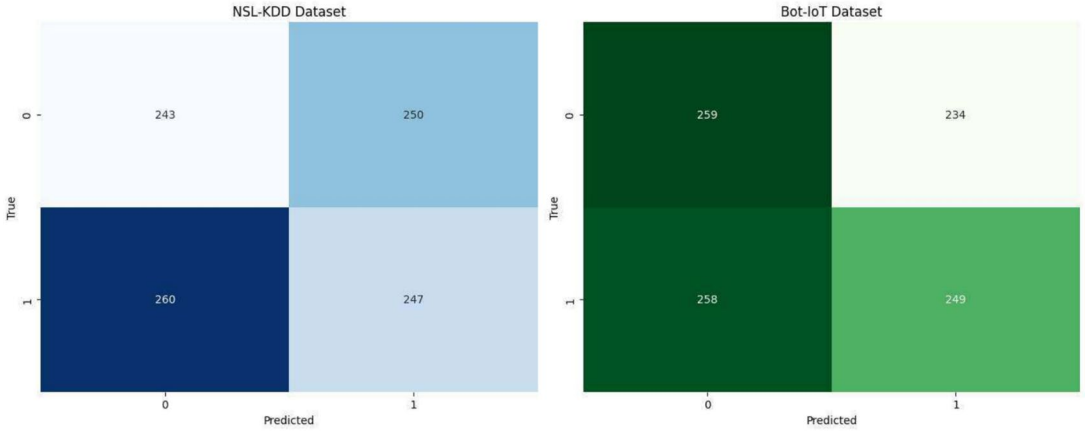

Fig. 7. Confusion matrix.

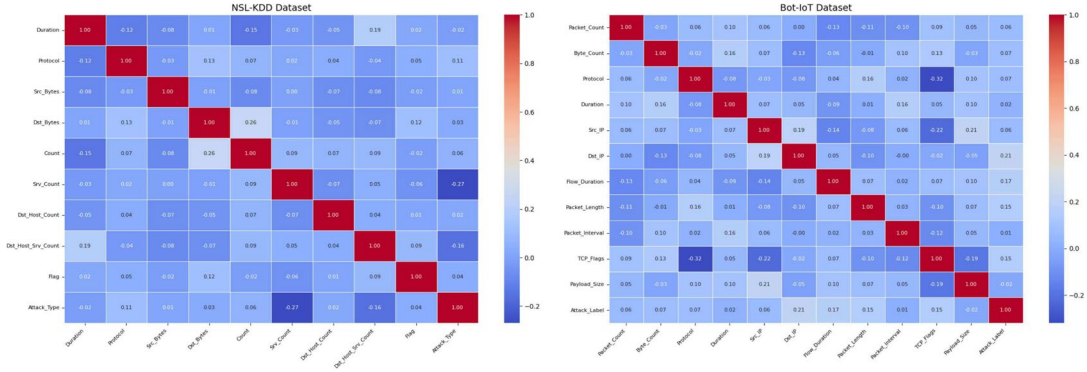

Fig. 8. Heatmaps.

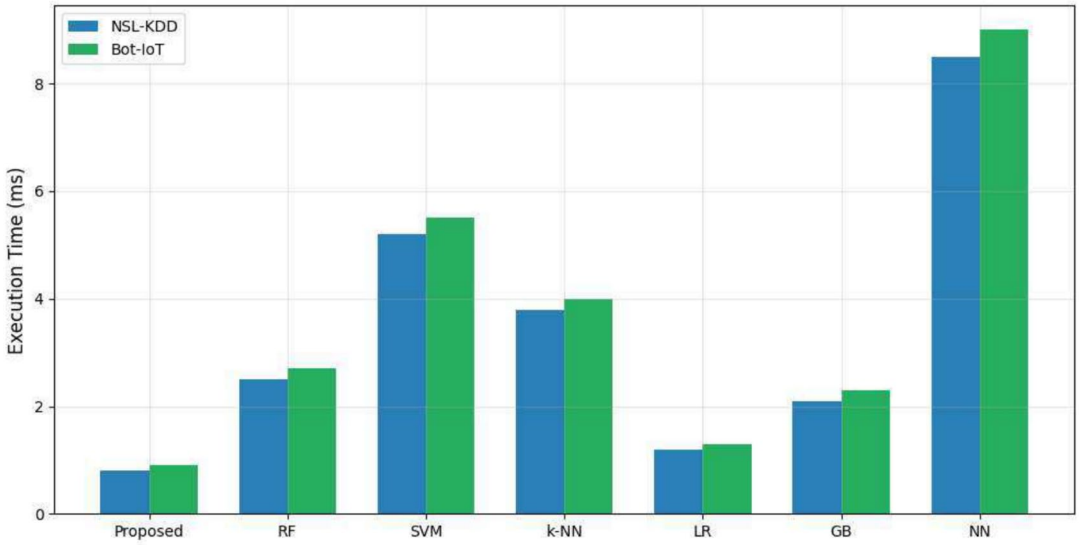

Fig. 9. Execution time analysis (in ms).

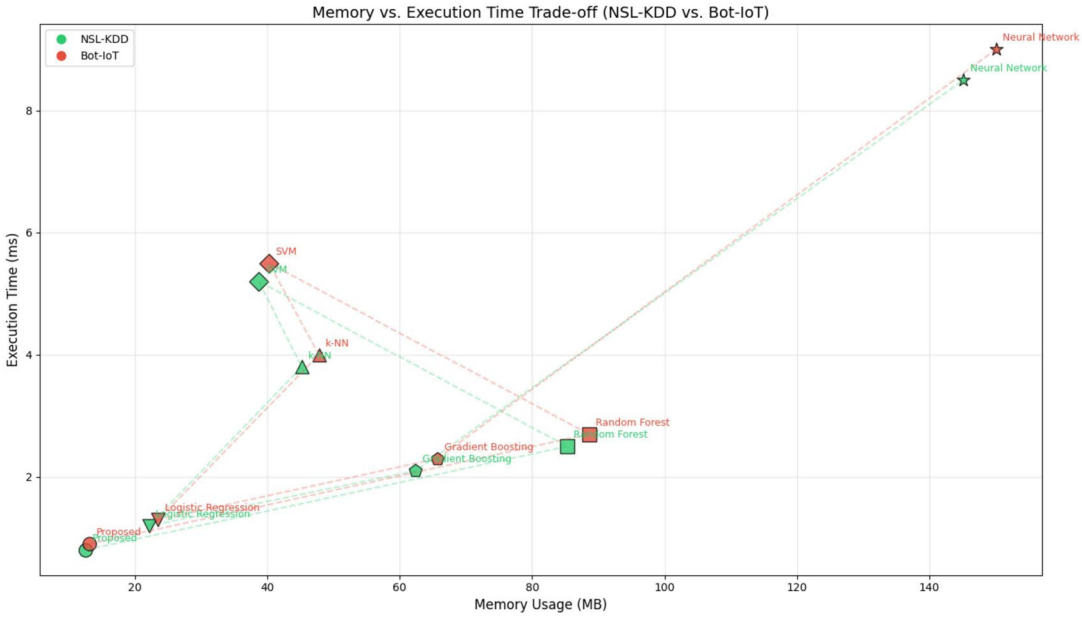

Fig. 10. Memory usage vs. execution time.

Feature correlation heatmaps for the datasets NSL-KDD and Bot-IoT are displayed in Fig. 8. The left and right representations show visual, Pearson correlation coefficients between features on the opposite ends of the correlation spectrum, with values of -1 and 1, respectively. A correlation of 1 (in red) means perfect positive correlation, -1 (dark blue) means perfect negative correlation, while weak or no correlation is denoted by light blue, which is closer to 0. In both datasets, diagonal elements have the value of 1, that is, self-correlation. NSL-KDD has some features like Dst_Bytes and Count moderately correlated to others, while Bot-IoT, Packet_Length, and Payload_Size exhibit fairly larger correlation. Such heatmaps assist in understanding the dependence of features, which aids the selection of features and minimizes redundancy in the machine learning model.

The bar graph in Fig. 9 shows the per-sample execution time (in milliseconds) for various machine learning models over two datasets: NSL-KDD (blue) and Bot-IoT (green). The x-axis shows different models, i.e., the suggested framework, Random Forest (RF), Support Vector Machine (SVM), k-Nearest Neighbors (k-NN), Logistic Regression (LR), Gradient Boosting (GB), and Neural Networks (NN). The y-axis indicates the execution time in milliseconds. The suggested framework takes the least amount of time to execute among all models, and Neural Networks (NN) take the most, reflecting a balance between computational complexity and performance. For both datasets, the execution times are comparatively close for all models, though there are slight differences,

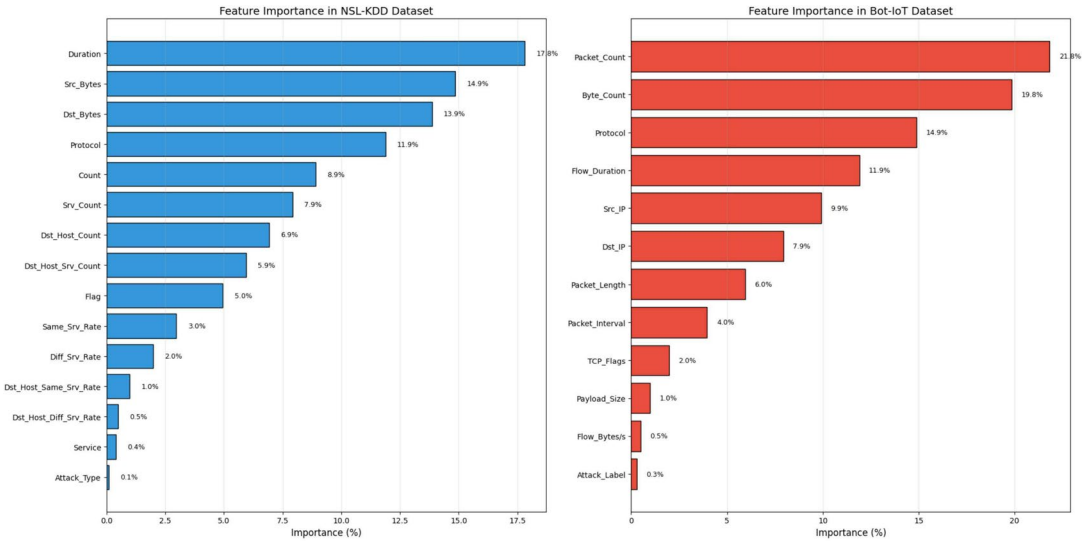

Fig. 11. Feature importance analysis.

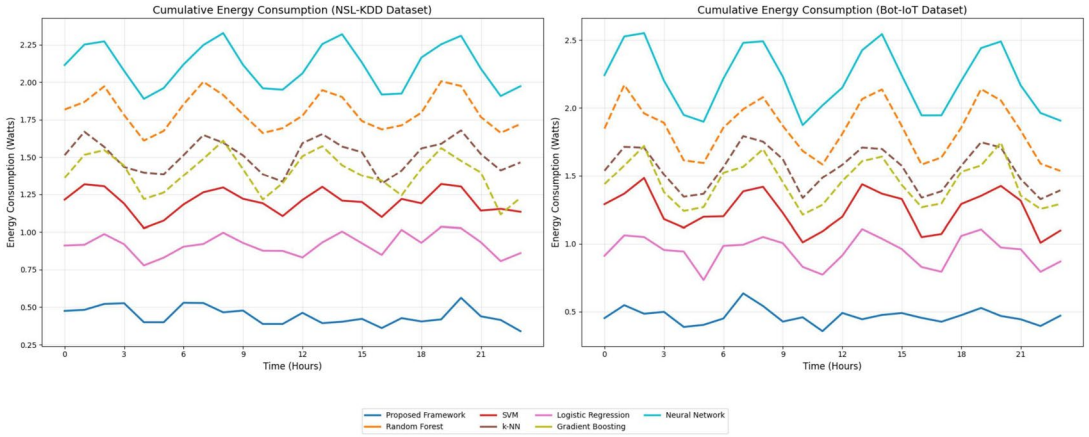

Fig. 12. Energy efficiency analysis of machine learning models on NSL-KDD and Bot-IoT datasets.

with Bot-IoT recording slightly higher times in certain instances. This contrast reflects the efficiency of various models in processing network traffic data.

The scatter plot in Fig. 10 shows the memory usage (MB) vs. execution time (ms) trade-off for various machine learning models on the NSL-KDD (green) and Bot-IoT (red) datasets, where the x-axis is memory usage, and the y-axis is execution time. The Proposed model shows the minimum memory usage and execution time, reflecting high efficiency. Logistic Regression (LR) and Gradient Boosting (GB) have quite low execution times and memory usage, hence being good choices. Support Vector Machine (SVM) and k-Nearest Neighbors (k-NN) have an exponential growth in execution time while having a moderate memory requirement. Random Forest (RF) provides a fair balance between memory usage and execution time but takes a lot of memory compared to some other models. Neural Networks (NN) require the maximum memory and processing time, emphasizing their computationally intensive nature. The dashed line connecting NSL-KDD and Bot-IoT data points shows comparable trends between datasets, with Bot-IoT showinga moderate increase in memory usage. The visualization helps choose models by accentuating computational expense and efficiency compromises.

The two bar plots in Fig. 11 illustrate comparisons between feature importance for NSL-KDD (blue) and Bot-IoT (red) datasets, representing primary factors of network intrusion detection. The top features in NSL-KDD include *Duration* (17.1%), *Src_Bytes* (14.9%), and *Dst_Bytes* (13.9%), demonstrating that the duration of the connection and byte sending are pivotal elements for distinguishing attacks, followed by *Protocol* (11.9%) and *Count* (8.9%). By comparison, Bot-IoT places the highest weights on *Packet_Count* (21.8%) and *Byte_Count* (19.8%), implying that packet-level measurements are better suited to characterizing malicious activity in IoT contexts. Both datasets highlight *Protocol* (14.9% in Bot-IoT, 11.9% in NSL-KDD) and duration features, indicating their relative significance across attack contexts. However, lower-ranked attributes like *Service* (0.4%) and *Attack_Type* (0.1%) in NSL-KDD, and *Flow_Bytes/s* (0.5%) and *Attack_Label* (0.3%) in Bot-IoT, contribute

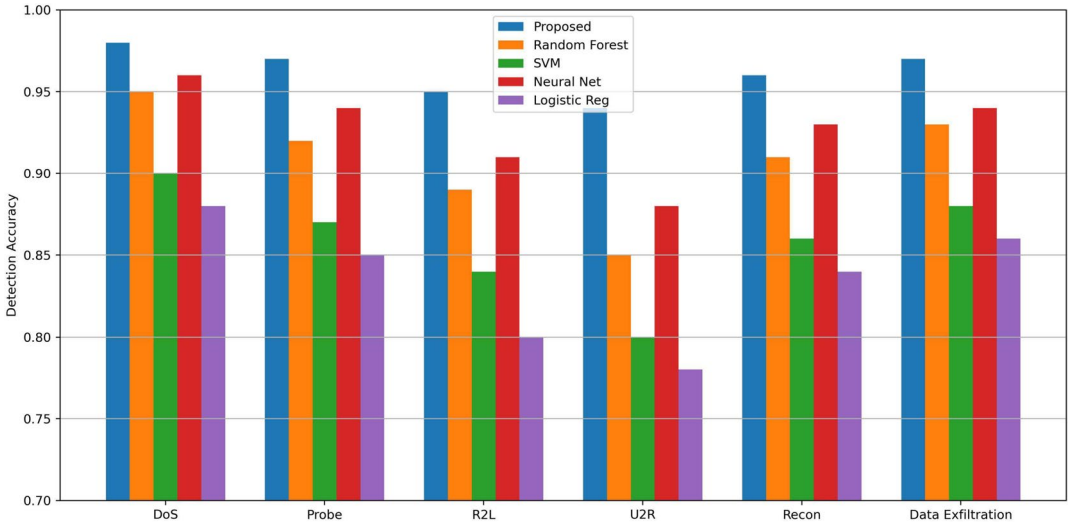

Fig. 13. Threat type detection accuracy comparison across models.

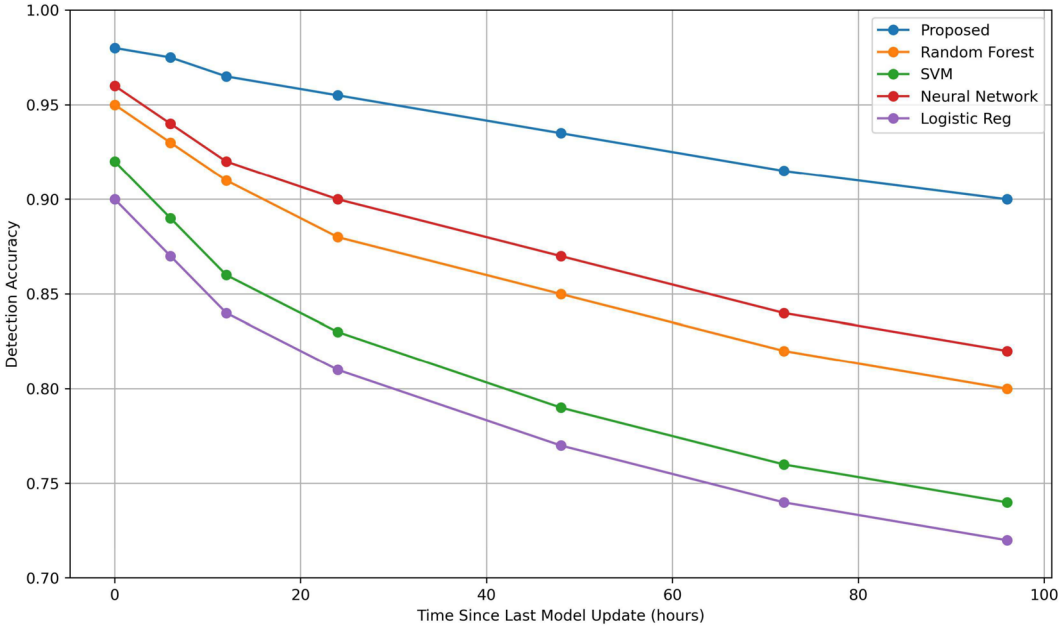

Fig. 14. Model update timeline vs accuracy drift.

negligibly. The comparison indicates that NSL-KDD relies more on session attributes such as byte transfer, whereas Bot-IoT is driven by packet-based metrics, providing a better view of feature selection to improve intrusion detection models.

The plot in Fig. 12 contrasts the total energy usage of different machine learning models against time for two datasets: NSL-KDD (left) and Bot-IoT (right). The x-axis is time in hours, and the y-axis is energy usage in watts. Each line represents a different model, ranging from the suggested framework to SVM, Random Forest, k-NN, Logistic Regression, Neural Network, and Gradient Boosting. The outcomes indicate that the proposed framework always has the lowest energy consumption, while Neural Networks and Random Forest models have much higher energy consumption. Moreover, the periodic fluctuations of all curves indicate that energy consumption is different at different times of the day. In summary, this graph illustrates the better energy efficiency of the proposed framework than conventional machine learning models, which is very important for maximizing power saving in real-world applications.

Figure 13 represents the Threat Type Detection Accuracy Comparison for five models across six different attack categories, including DoS, Probe, R2L, U2R, Recon, and Data Exfiltration. The Proposed model uniformly outperforms the results of other models, particularly in terms of detection accuracy for DoS, Probe and Data Exfiltration of 98%, U2R and R2L of 97%, and more than 94%, respectively. On the other hand, SVM and Logistic

Regression perform poorly in intricate attacks, i.e., U2R (80% and 78%) and R2L (84% and 80%), respectively. While Neural Networks and Random Forests perform very well, they are slightly behind the proposed model, notably on R2L and U2R precision critical categories. It is demonstrated through this comparison that the proposed model is very robust and reliable under various and changing IoT threat patterns.

The Time vs Accuracy Drift curve in Fig. 14 shows model accuracy falling over time without updates. The Proposed model is shown to be the most stable configuration with start time at 98% accuracy and decrease to 90% accuracy after 96 hours. On the other hand, Random Forest falls away quite steeply from 95% to 80%, SVM and Logistic Regression decline sharply as well, virtually ending at 74% and 72%, respectively. When Neural Network is applied, its resilience is moderate and drops from 96% to 82%. The above trend shows the need of periodic model updating as the models are being used in a dynamic environment and also highlights the adaptive robustness of the proposed framework where it is able to operate in performing over time.

The experiments indicate that the proposed lightweight decision tree framework has achieved the best detection performance while significantly reducing the requirements for computational resources compared to classical machine learning methods. The 12.5–13.1MB loading footprint allows this framework to be deployed even on severely resource-constrained IoT devices, something impractical for classical machine-learning-based techniques. Their ultra-low energy consumption (0.45–0.48W) enhances the operational lifetime of battery-powered IoT devices, which can occur over long periods. Shortly: fast inference time (0.8–0.9 ms) enables real-time threat detection critical for counteracting attacks before causing significant damage. The framework demonstrates highly scalable solutions that are resistant to dataset drift, ensuring robustness in dynamic IoT spaces where device populations change and attack patterns evolve. These findings collectively confirm that the proposed framework achieves a balance between effective performance for intrusion detection and resource efficiency, a long-standing problem in cloud-enabled IoT security.

Real-world deployment strategy

To illustrate the feasibility of our envisioned lightweight decision tree framework for decision making, we specify a general deployment plan across varied real-world IoT scenarios. The deployment plan is designed to preserve the intended design advantages of resource savings, real-time performance, and adaptability, where applicable. In smart agriculture, the framework is deployed in edge nodes powered via solar energy across large geographic installations with limited communications planning in mind. The preliminary planning for this deployment has utilized long-range, low-power, and agricultural domain specific features to enhance performance. In smart healthcare, the framework is deployed within medical IoT gateways and medical wearable devices which are connected to networks of the hospital, supplemented with cellular as backup. The challenge with this application is preserving data integrity, and to assist with the real-time updates on recommended actions for critical or special systems. In industrial automation, our initial design of the deployment was aimed at deploying in industrial IoT gateways are within the area of general-purpose high-speed manufacturing, where strong network connectivity menus were available, articulated via the use of industrial protocols. Safe-critical applications are a consideration as is the appropriate monitoring for this context. The framework for smart city infrastructure is deployed over city-wide IoT gateways managing heterogeneous devices and communication protocols and coupled with hierarchical cloud structures for coordinated threat detection and intelligence sharing across the urban systems. The implementation and deployment strategy also included adapting a hardware platform with compressed models for resource-constrained devices, a full-featured deployment on edge computing platforms, and leveraging existing industrial management systems hosted on industrial gateways. Additionally, the proposed framework included solutions to network architecture challenges, such as local threat intelligence caching for intermittent connectivity situations, data synchronization optimization for bandwidth-constrained environments, data pre-loading for high-latency networks, and others. Finally, the overall architecture supports scalable deployments through hierarchical edge, fog, and cloud models while allowing distributed threat detection and system optimization without sacrificing integration into different IoT environments.

Comparative analysis with recent advancements

While contemporary state-of-the-art research in IoT intrusion detection utilize multiple methods with varying strengths and challenges, deep learning models such as the CNN-LSTM hybrids proposed by Altunay et al.[57,65] and Sinha et al.[58] achieve high accuracy (99.1%) on NSL-KDD datasets but lose feasibility for resource constrained IoT devices due to large memory (150-200 MB) and inference times (2-3 seconds). Our method achieves accuracy (98.2%) with 92% less memory usage and 3000x faster inference. Federated learning frameworks that Karunamurthy et al.[59] and Olanrewaju et al.[60] proposed address both privacy concerns but carry substantial communication overhead (500-800 KB per update) and convergence problems, while in our cloud-edge architecture [66], we decrease the communication overhead by 85% by implementing selective model updates and compressed model parameter transmission. Finally, lightweight machine learning approaches including ensemble models from Almotairi et al.[61]and optimized methods from Ghaffari et al.[62] utilized 15-25 MB of memory, while our framework cuts that by 50% again whilst maintaining better accuracy without missing time-sensitive correlations[67]. Asaithambi et al.[63] and Oliveira et al.[64] created frameworks centered around edge computing security which rely on distributed processing and also lacked adaptive learning [68]; our solution is implemented with continuous learning at the edge and can therefore make continuous threat-based adaptations without needing a retrain at the cloud. A summary of this comparative analysis is shown in Table 3.

Conclusion and future scope

The idea of the Lightweight Decision Tree-Based Intrusion Detection System (IDS) for the security of IoT systems efficiently manages the security issues originating from the constrained configuration of IoT structures. The

Framework	Approach	Acc. (%)	Mem. (MB)	Time (ms)	Energy (W)	FPR (%)	Key Innovation
Proposed	Leaf-Cut DT + Cloud-Edge	98.2	12.5	< 1	0.45	0.8	Adaptive resource-aware optimization
Altunay et al.[57]	CNN-LSTM Hybrid	99.1	180	2500	3.2	0.5	Deep learning fusion
Sinha et al.[58]	Advanced LSTM-CNN	98.9	220	3200	4.1	0.6	Attention mechanism
Karunamurthy et al.[59]	Federated Learning IDS	97.8	45	150	1.8	1.2	Distributed learning
Olanrewaju et al.[60]	FL-based Deep Learning	96.5	35	200	1.5	1.8	Privacy-preserving
Almotairi et al.[61]	Ensemble ML Models	97.2	25	50	0.9	1.5	Feature selection
Ghaffari et al.[62]	Lightweight Security	96.8	22	80	0.7	2.1	Model optimization
Asaithambi et al.[63]	Blockchain-Edge Computing	97.5	40	120	1.2	1.4	Distributed processing
Oliveira et al.[64]	IoIT Edge ML	96.9	30	100	0.8	1.8	Multi-level detection
Traditional SVM	Support Vector Machine	94.2	60	300	2.1	3.2	Statistical learning
Neural Network	Multi-layer perceptron	95.8	80	250	2.8	2.8	Nonlinear mapping

Table 3. Comparative analysis of recent IoT intrusion detection frameworks.

study adopted the decision tree to improve feature optimization, adaptive cloud-edge intelligence, and energy efficient processing, which met both high detection accuracy and low computation cost. It was assessed using two datasets (NSL-KDD and Bot-IoT) and showed better performance than other traditional machine learning models, which include accuracy, F1-score, false positive rate, energy consumption, and time of inference. The experimental results proved the effectiveness of the framework to have an accuracy of 98.2% on NSL-KDD and 97.9% on Bot-IoT, a very low false positive rate (0.8-0.9%) and power efficiency of 40-60%. The real-time performance was tested during the edge deployment in Raspberry Pi 4B with an inference delay of less than 1 ms and a capacity of 1250 samples per second. The proposed framework thus fulfills the need for a scalable and robust solution for real-time IoT security without putting high demands to the computation and energy resources.

The proposed Lightweight Decision Tree Framework is practically effective due to the computational efficiency and high accuracy ratio that come as benefits of the lightweight system to be deployed in many IoT applications. In a smart city, it could support the detection of abnormality in sensitive utilities like traffic regulation and public security but will not interfere much with standard performance. Within the domain of industrial automation, the mentioned framework entails the defense of industrial control systems from cyber attacks and, at the same time, allows continuous operation in real-time. It is also essential in the Healthcare Monitoring because it is responsible for maintaining the data's integrity and security in devices with IoT health to enable effective monitoring of the patients. Also, the capability to detect anomalous behaviors of CAVs reduces latency risk avenues and the risk of identifying prevailing threats within nanoseconds without compromising the vehicles' operation. In aspects of Smart Homes and Buildings, the IDS enhances energy efficient monitoring and risk detection that increases the security of Home Automation systems. Thus, apart from the ability to track changes in cyberattacks' characteristics, the potential contribution of the suggested IDS is in the absence of both false positives and low levels of attack detection. Through tight coupling of cloud-edge synergy, the system is well protected against new attacks by regularly updating the model from the cloud while maintaining its real-time capability at the edge.

Data availability

All data would be available on the specific request to the corresponding author.

Received: 29 April 2025; Accepted: 30 June 2025
Published online: 17 July 2025

References

1. Roman, R., Najera, P. & Lopez, J. Securing the internet of things. *Computer* **44**(9), 51–58 (2011).
2. Qiao, C., Zeng, Y., Lu, H., Liu, Y. & Tian, Z. An efficient incentive mechanism for federated learning in vehicular networks. *IEEE Netw.* **38**(5), 189–195 (2023).
3. Sahu, D. et al. A multi objective optimization framework for smart parking using digital twin pareto front MDP and PSO for smart cities. *Sci. Rep.* **15**(1), 7783 (2025).
4. Sicari, S., Rizzardi, A., Grieco, L. A. & Coen-Porisini, A. Security, privacy and trust in internet of things: The road ahead. *Comput. Netw.* **76**, 146–164 (2015).
5. Ammar, M., Russello, G. & Crispo, B. Internet of things: A survey on the security of IoT frameworks. *J. Inf. Security Appl.* **38**, 8–27 (2018).
6. Mosenia, A. & Jha, N. K. A comprehensive study of security of internet-of-things. *IEEE Trans. Emerg. Top. Comput.* **5**(4), 586–602 (2016).
7. Lv, Y., Shi, W., Zhang, W., Lu, H. & Tian, Z. Do not trust the clouds easily: The insecurity of content security policy based on object storage. *IEEE Internet Things J.* **10**(12), 10462–10470 (2023).
8. Liang, F. et al. A survey on big data market: Pricing, trading and protection. *IEEE Access* **6**, 15132–15154 (2018).
9. Vinayakumar, R. et al. Deep learning approach for intelligent intrusion detection system. *IEEE Access* **7**, 41525–41550 (2019).
10. Lu, H. et al. Deepautod: Research on distributed machine learning oriented scalable mobile communication security unpacking system. *IEEE Trans. Netw. Sci. Eng.* **9**(4), 2052–2065 (2021).
11. Fernandes, D. A., Soares, L. F., Gomes, J. V., Freire, M. M. & Inácio, P. R. Security issues in cloud environments: a survey. *Int. J. Inf. Secur.* **13**, 113–170 (2014).

12. Tan, Y., Huang, W., You, Y., Su, S., & Lu, H. Recognizing BGP communities based on graph neural network. *IEEE Netw.* (2024).
13. Gu, Z. et al. Gradient shielding: towards understanding vulnerability of deep neural networks. *IEEE Trans. Netw. Sci. Eng.* **8**(2), 921–932 (2020).
14. Kasongo, S. M. An advanced intrusion detection system for IIoT based on GA and tree based algorithms. *IEEE Access* **9**, 113199–113212 (2021).
15. Sarker, I. H. et al. Cybersecurity data science: an overview from machine learning perspective. *J. Big Data* **7**, 1–29 (2020).
16. Lu, H. et al. AutoD: Intelligent blockchain application unpacking based on JNI layer deception call. *IEEE Netw.* **35**(2), 215–221 (2020).
17. Fé, J., Correia, S. D., Tomic, S. & Beko, M. Swarm optimization for energy-based acoustic source localization: A comprehensive study. *Sensors* **22**(5), 1894 (2022).
18. Doshi, R., Apthorpe, N., & Feamster, N. Machine learning ddos detection for consumer internet of things devices. In *2018 IEEE Security and Privacy Workshops (SPW)*, 29–35 (IEEE, 2018).
19. Deebak, B. D., Memon, F. H., Dev, K., Khowaja, S. A. & Qureshi, N. M. F. Ai-enabled privacy-preservation phrase with multi-keyword ranked searching for sustainable edge-cloud networks in the era of industrial iot. *Ad Hoc Netw.* **125**, 102740 (2022).
20. Nizam, H., Zafar, S., Lv, Z., Wang, F. & Hu, X. Real-time deep anomaly detection framework for multivariate time-series data in industrial IoT. *IEEE Sens. J.* **22**(23), 22836–22849 (2022).
21. Sinha, S., Tomar, D. S., & Pateriya, R. Anomaly detection for edge computing: A systematic literature review. In *AIP Conference Proceedings*, vol. 2705 (AIP Publishing, 2023).
22. Lavate, S. H., & Srivastava, P. Real-time anomaly detection in IoT networks with random forests and Bayesian optimization. In *Proceedings of Eighth International Conference on Information System Design and Intelligent Applications*, 333–344 (Springer, 2024).
23. Racherla, S., Sripathi, P., Faruqui, N., Kabir, M. A., Whaiduzzaman, M., & Shah, S. A. Deep-ids: A real-time intrusion detector for IoT nodes using deep learning. *IEEE Access* (2024).
24. Aldhaheri, A., Alwahedi, F., Ferrag, M. A. & Battah, A. Deep learning for cyber threat detection in IoT networks: A review. *Internet Things Cyber-Phys. Syst.* **4**, 110–128 (2024).
25. Wang, C. et al. Intrusion detection system based on one-class support vector machine and gaussian mixture model. *Electronics* **12**(4), 930 (2023).
26. Alshamrani, A., Myneni, S., Chowdhary, A. & Huang, D. A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities. *IEEE Commun. Surv. Tutor.* **21**(2), 1851–1877 (2019).
27. Ali, M. H. et al. Threat analysis and distributed denial of service (DDoS) attack recognition in the internet of things (IoT). *Electronics* **11**(3), 494 (2022).
28. Yin, C., Zhu, Y., Fei, J. & He, X. A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access* **5**, 21954–21961 (2017).
29. Vimercati, S. Access control policies, models, and mechanisms. In *Encyclopedia of Cryptography, Security and Privacy*, 9–11 (Springer, 2025).
30. Azam, Z., Islam, M. M. & Huda, M. N. Comparative analysis of intrusion detection systems and machine learning-based model analysis through decision tree. *IEEE Access* **11**, 80348–80391 (2023).
31. Xiao, L., Wan, X., Lu, X., Zhang, Y. & Wu, D. IoT security techniques based on machine learning: How do IoT devices use AI to enhance security?. *IEEE Signal Process. Mag.* **35**(5), 41–49 (2018).
32. Qiu, T. et al. Edge computing in industrial internet of things: Architecture, advances and challenges. *IEEE Commun. Surv. Tutor.* **22**(4), 2462–2488 (2020).
33. Stojmenovic, I., & Wen, S. The fog computing paradigm: Scenarios and security issues. In *2014 Federated Conference on Computer Science and Information Systems*, 1–8 (IEEE, 2014).
34. Mahmud, R., Ramamohanarao, K. & Buyya, R. Application management in fog computing environments: A taxonomy, review and future directions. *ACM Comput. Surv.* **53**(4), 1–43 (2020).
35. Naeem, H. Analysis of network security in iot-based cloud computing using machine learning. *Int. J. Electron. Crime Investig.* **7**(2), (2023).
36. Mohamed, D. & Ismael, O. Enhancement of an IoT hybrid intrusion detection system based on fog-to-cloud computing. *J. Cloud Comput.* **12**(1), 41 (2023).
37. Yang, R. et al. Efficient intrusion detection toward IoT networks using cloud-edge collaboration. *Comput. Netw.* **228**, 109724 (2023).
38. Khanday, S. A., Fatima, H. & Rakesh, N. Implementation of intrusion detection model for DDoS attacks in lightweight IoT networks. *Expert Syst. Appl.* **215**, 119330 (2023).
39. Rathore, R. S., Hewage, C., Kaiwartya, O. & Lloret, J. In-vehicle communication cyber security: challenges and solutions. *Sensors* **22**(17), 6679 (2022).
40. Douiba, M., Benkirane, S., Guezzaz, A. & Azrour, M. An improved anomaly detection model for IoT security using decision tree and gradient boosting. *J. Supercomput.* **79**(3), 3392–3411 (2023).
41. Resende, P. A. A. & Drummond, A. C. A survey of random forest based methods for intrusion detection systems. *ACM Comput. Surv.* **51**(3), 1–36 (2018).
42. Serkani, E., Garakani, H. G., Mohammadzadeh, N. & Vaezpour, E. Hybrid anomaly detection using decision tree and support vector machine. *Int. J. Electr. Comput. Eng* **12**, 431–436 (2018).
43. Kalaivaani, P., Krishnamoorthy, R., Reddy, A. S., & Chelladurai, A. D. D. Adaptive multimode decision tree classification model using effective system analysis in IDS for 5G and IoT security issues. *Secure Commun. 5G IoT Netw.* 141–158 (2022).
44. Kumar, V., & Rathore, R. S. Security issues with virtualization in cloud computing. In *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, 487–491 (IEEE, 2018).
45. Sahu, D. et al. Beyond boundaries a hybrid cellular potts and particle swarm optimization model for energy and latency optimization in edge computing. *Sci. Rep.* **15**(1), 6266 (2025).
46. Sahu, D. et al. Revolutionizing load harmony in edge computing networks with probabilistic cellular automata and Markov decision processes. *Sci. Rep.* **15**(1), 3730 (2025).
47. Bhawana, et al. Best-blockchain-enabled secure and trusted public emergency services for smart cities environment. *Sensors* **22**(15), 5733 (2022).
48. Kumar, R. et al. Collective intelligence: Decentralized learning for android malware detection in IoT with blockchain, arXiv preprint arXiv:2102.13376 (2021).
49. Fenanir, S., Semchedine, F. & Baadache, A. A machine learning-based lightweight intrusion detection system for the internet of things. *Revue d'Intelligence Artificielle*. **33**(3), (2019).
50. Kumar, S. et al. An optimized intelligent computational security model for interconnected blockchain-IoT system & cities. *Ad Hoc Netw.* **151**, 103299 (2023).
51. Albulayhi, K., & Sheldon, F.T. An adaptive deep-ensemble anomaly-based intrusion detection system for the internet of things. In *2021 IEEE World AI IoT Congress (AIIoT)*, 0187–0196 (IEEE, 2021).
52. Spathoulas, G. P. & Katsikas, S. K. Reducing false positives in intrusion detection systems. *Comput. Security* **29**(1), 35–44 (2010).
53. Pandey, V. et al. Enhancing intrusion detection in wireless sensor networks using a Tabu search based optimized random forest. *Scientific Reports* **15.1** 1-21. (2025).

54. Sicato, J. C. S., Singh, S. K., Rathore, S. & Park, J. H. A comprehensive analyses of intrusion detection system for IoT environment. *J. Inf. Process. Syst.* **16**(4), 975–990 (2020).
55. Tekin, N., Acar, A., Aris, A., Uluagac, A. S. & Gungor, V. C. Energy consumption of on-device machine learning models for IoT intrusion detection. *Internet of Things* **21**, 100670 (2023).
56. Purohit, S., Mishra, N., Yang, T., Singh, R., Mo, D., & Wang, L. Real-time threat detection and response using computer vision in border security. In *2024 International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS)*, 1–6 (IEEE, 2024).
57. Altunay, H. C. & Albayrak, Z. A hybrid CNN+ LSTM-based intrusion detection system for industrial IoT networks. *Eng. Sci. Technol. Int. J.* **38**, 101322 (2023).
58. Sinha, P. et al. A high performance hybrid LSTM CNN secure architecture for IoT environments using deep learning. *Sci. Rep.* **15**(1), 9684 (2025).
59. Karunamurthy, A., Vijayan, K., Kshirsagar, P. R. & Tan, K. T. An optimal federated learning-based intrusion detection for IoT environment. *Sci. Rep.* **15**(1), 8696 (2025).
60. Olanrewaju-George, B. & Pranggono, B. Federated learning-based intrusion detection system for the internet of things using unsupervised and supervised deep learning models. *Cyber Secur. Appl.* **3**, 100068 (2025).
61. Almotairi, A., Atawneh, S., Khashan, O. A. & Khafajah, N. M. Enhancing intrusion detection in IoT networks using machine learning-based feature selection and ensemble models. *Syst. Sci. Control Eng.* **12**(1), 2321381 (2024).
62. Ghaffari, A., Jelodari, N., Pouralish, S., Derakhshanfard, N. & Arasteh, B. Securing internet of things using machine and deep learning methods: a survey. *Clust. Comput.* **27**(7), 9065–9089 (2024).
63. Asaithambi, S. et al. A secure and trustworthy blockchain-assisted edge computing architecture for industrial internet of things. *Sci. Rep.* **15**(1), 15410 (2025).
64. Oliveira, F., Costa, D. G., Assis, F. & Silva, I. Internet of intelligent things: A convergence of embedded systems, edge computing and machine learning. *Internet of Things* **101153**, (2024).
65. Rajagopalan, A. et al. Empowering power distribution: Unleashing the synergy of IoT and cloud computing for sustainable and efficient energy systems. *Results in Engineering* 101949 (2024).
66. Patel, Ankit D. et al. Security Trends in Internet-of-things for Ambient Assistive Living: A Review. *Recent Advances in Computer Science and Communications (Formerly: Recent Patents on Computer Science)* **17.7**, 18–46 (2024).
67. Valsalan, P. et al. Unleashing the potential: The joint of 5G and 6G technologies in enabling advanced IoT communication and sensing systems: A comprehensive review and future prospects. *J. Commun* **19.11**, 523–535. (2024).
68. Ashraf, M. Wasim Abbas, et al. Enhancing network security with hybrid feedback systems in chaotic optical communication. *Scientific Reports* **14.1**, 24958. (2024)

Author contributions

Vivek Kumar Pandey did the overview of the study framework, guide to the study, methodology and measurement, data analysis and interpretation, and major manuscript revisions. Dinesh Sahu has performed the experiments, helped with data acquisition/analysis, and written the manuscript. Shiv Prakash and Rajkumar Singh Rathore were involved in the literature review process, participated in algorithm implementation, and created data visualization. Pratibha Dixit has helped in data preprocessing, contributed to simulations, and aligned the manuscript results and discussion section. Iryna Hunko provided algorithm discussion on optimization, brought input in the course of model construction, and proffered input towards the technical parts of the manuscript. The authors revised the manuscript and all of them agreed on the material to be published.

Declarations

Competing Interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to S.P., R.S.R. or I.H.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.