Hardware/Software Codesign of Real-Time Intrusion Detection System for Internet of Things Devices

Qingyu Zeng and Yuko Hara-Azumi, *Member, IEEE*

***Abstract*—The rapid expansion of the Internet of Things (IoT) has increased security concerns, thereby necessitating efficient intrusion detection systems (IDSs). In this article, we propose a real-time IoT IDS designed by combining a random forest (RF) classifier with an ensemble feature selection technique (EFST). The proposed IDS can be deployed on a small-scale field-programmable gate array (FPGA) board. The system utilizes a two-metric ensemble feature selection process to reduce computational complexity and enhance classification accuracy. In addition, the EFST aggressively extracts a limited number of features, thereby reducing the complexity of the RF model. Then, the tailored RF classifier is mapped onto an FPGA-based hardware accelerator to realize real-time detection. The proposed method was evaluated experimentally on the benchmark BoT-IoT data set. The results demonstrate that the proposed IDS realizes significant improvements in terms of resource utilization and processing time compared to several state-of-the-art FPGA-based IDS implementations while maintaining sufficient detection accuracy. In particular, our implementation on the Xilinx PYNQ Z2 achieved 10.2×, 135.7×, and 8.43× speed-up compared to state-of-the-art IDSs running on an Intel Core i7 CPU, an ARM Cortex-A9 microprocessor, and a neural network-based accelerator on the PYNQ, respectively. In addition, our approach exhibits the lowest resource utilization among FPGA-based IDS solutions. These results demonstrate that this work contributes to developing secure and sustainable IoT ecosystems by integrating EFST, RF classification, and FPGA-based acceleration.**

***Index Terms*—Field-programmable gate array (FPGA), hardware accelerator, Internet of Things (IoT), intrusion detection, random forest (RF).**

I. Introduction

With the proliferation of the Internet of Things (IoT), there has been a significant increase in the number of connected devices, which has resulted in a corresponding increase in security concerns [1]. Consequently, intrusion detection systems (IDSs) have become a critical component of network security, particularly to protect IoT devices against cyberattacks [2], e.g., distributed Denial of Service (DDoS) attacks [3], reconnaissance [4], and information theft [5]. Among the various anomaly intrusion detection algorithms, machine learning (ML) algorithms like random forest (RF) [6] and neural networks (NNs) [7] are effective [8], [9], [10]. These ML algorithms can realize good performance (i.e., sufficient inference accuracy) by increasing model complexity at the expense of increased processing time [11]. In other words, it is inherently difficult to construct both lightweight and accurate ML models. Thus, developing an ML-based IDS that can detect anomaly intrusions accurately in real time on resource-constrained IoT devices is an important but challenging issue.

Generally, the high dimensionality of input data is a bottleneck when reducing the complexity of ML models in terms of both computational costs and the amount of data. A simple and effective solution to this problem is to select and use a subset of important features for the ML model training and inference processes. With this approach, we can expect relaxed computational complexity, reduced memory costs, and improved inference accuracy [12]. Note that network traffic involves a high number of features and feature selection processes contribute to lightweight ML models for IDSs. In addition, ensemble feature selection techniques (EFSTs), which integrate the outcomes of various feature selection algorithms or data subsets, can be used to further identify the most important features from different perspectives [13]. Recently, existing IDS studies have exploited EFSTs to improve the detection accuracy of software-defined ML models [4], [5]. However, these studies utilized powerful off-the-shelf platforms [e.g., graphics processing units (GPUs)] to process the ML model; thus, they only filtered out ineffective features using the EFST and proceeded to use many features (e.g., 20 features in [5]). Considering that IDSs must be deployed on many small IoT devices, when they are hardware/software codesigned in manner similar to various embedded systems, the number of features should be reduced as much as possible to reduce both hardware costs and processing time. In other words, to develop a sufficiently lightweight ML model, the EFST should aggressively select a limited number of features.

In addition to the lightweight model design, the hardware accelerator architecture affects the performance of real-time processing. Previous studies have demonstrated that field-programmable gate arrays (FPGAs) are a promising platform to accelerate ML models in a number of emerging IoT applications (e.g., image classification [14], object detection [15], and image compression [16]). Note that ML

Manuscript received 15 May 2023; revised 7 December 2023 and 6 March 2024; accepted 19 March 2024. Date of publication 22 March 2024; date of current version 7 June 2024. This work was supported in part by JSPS KAKENHI under Grant JP21H03399 and Grant JP20H04154. *(Corresponding author: Qingyu Zeng.)*

The authors are with the School of Engineering, Tokyo Institute of Technology, Tokyo 1528550, Japan (e-mail: qyzeng@cad.ict.e.titech.ac.jp; hara@cad.ict.e.titech.ac.jp).

Digital Object Identifier 10.1109/JIOT.2024.3380822

© 2024 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

TABLE I
BoT-IoT Data Set

Category	Type of traffic	Instances #
Normal	Legitimate network traffic	9,543
DoS	DoS attack traffic	33,005,194
DDoS	DDoS attack traffic	38,532,480
Reconnaissance	Network traffic used to gather information	1,821,639
Information Theft	Network traffic used to steal sensitive data	1,587

algorithms have inherent parallelism; thus, accelerators that fully exploit this parallelism are expected to realize real-time intrusion detection. However, existing FPGA-based IDS acceleration methods must consider the difficulties associated with a compact accelerator design. Due to extensive use of digital signal processing units (DSPs) for matrix multiplication and accumulation operations in NN-based IDSs [9], [10] or memory-hungry implementations in RF-based network traffic classification [17], such accelerators cannot be deployed on small-scale FPGAs.

Thus, in this article, we propose a real-time IDS design that considers both hardware and software approaches. The goal of the proposed IDS is to provide accurate and efficient network intrusion detection while also addressing the challenges associated with real-time processing and the limited hardware resources of IoT devices. Among various ML algorithms, the proposed IDS utilized an RF algorithm because it can realize high classification accuracy and is robust against noise and outliers in the network traffic. In addition, in terms of hardware accelerator implementation, the RF algorithm is generally more lightweight than an NN solution, and it possesses inherently rich parallelism between decision trees. To comprehensively design a lightweight RF-based IDS, we first present an EFST that combines two feature selection metrics to evaluate the network traffic. Specifically, we utilize the Gini index [18] and the out-of-bag (OOB) score [19] because these metrics take distinct importance perspectives in terms of the RF algorithm. With our EFST, only a few important features are selected and used to develop and train our lightweight RF model. We then codesign the RF-based IDS on a heterogeneous Zynq SoC (PYNQ Z2) by utilizing both the CPU and the configurable fabrics of an FPGA. Here, a feature extractor is implemented to handle the network traffic as software running on the CPU, and the RF hardware accelerator is implemented on the FPGA fabrics in a flexibly programmable manner (i.e., parameterized and modulated). Compared to existing programmable accelerators [17], [20], our accelerator consumes much fewer resources and fits in a small-scale FPGA while achieving comparable latency. The proposed IDS was evaluated on a publicly available data set, i.e., the BoT-IoT data set [21] (Table I [22]), to quantitatively demonstrate its effectiveness against several state-of-the-art ML-based IDSs.

The primary contributions of this study are summarized as follows.

1) The proposed EFST utilizes the Gini Index and OOB score metrics to rank feature importance. Given the high dimensionality of network traffic features, the EFST selectively extracts a limited number of highly important features. This strategic extraction contributes to reducing the complexity of the RF model. In an experimental evaluation, among 15 standard and effective features [22], [23] for the BoT-IoT data set (used as a full feature set), the proposed EFST reduced the number of features to three while maintaining inference performance that is comparable with that of a model that used the full feature set.
2) The proposed RF accelerator is flexibly programmable in terms of the number of trees and the maximum depth of the RF model. These parameters are tunable during the design phase. The proposed EFST reduces the number of features, thereby reducing the resource utilization of the model. In addition, we address a potential issue regarding inefficient memory usage to store information about the structure of the model [17].
3) The proposed RF-based IDS is codesigned relative to both hardware and software to facilitate implementation on a heterogeneous Zynq SoC. Compared to existing FPGA-based IDS methods, the proposed method achieves the lowest resource utilization. In addition, the processing time of the proposed IDS on the Zynq SoC is up to two magnitudes faster than state-of-the-art IDSs. For instance, it is 10.2×, 135.7×, and 8.43× faster than systems running on an Intel Core i7 CPU, an ARM Cortex-A9 microprocessor, and an NN-based accelerator on the Zynq, respectively.

The remainder of this article is organized as follows. Section II briefly describes the basic technologies utilized in this study. Section III explains the proposed hardware/software codesigned IDS for IoT devices. Section IV holistically evaluates the proposed method compared to state-of-the-art methods. Finally, this article is concluded in Section V.

II. Preliminaries

In this section, we briefly review the fundamental technologies utilized in the proposed IDS, i.e., the RF algorithm and feature selection.

A. Random Forest

The RF algorithm is an ensemble learning method comprising multiple decision trees, each of which is constructed independently using bootstrapped samples from the original data set. There are two important hyperparameters in an RF model, i.e., the number of trees (N) and the maximum depth of each decision tree (M). The former balances a tradeoff between the complexity of the model and the inference performance, and the latter determines how many splits are permitted in the tree. Fig. 1 shows an example RF model comprising three decision trees, each of which has a depth of three (i.e., $N = 3$ and $M = 3$). Here, each split (or node) has two child nodes (the left and right nodes), either of which is taken according to the features of the input data and the corresponding threshold values.

In inference, computational complexity stems from the number of trees (N), maximum tree depth (M), and considered features. Storage requirements depend on the number of trees

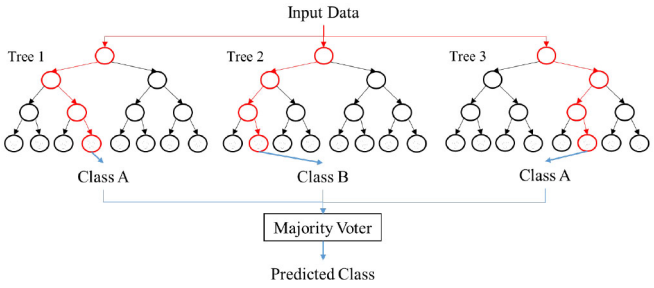

Fig. 1. Example RF model comprising three decision trees.

(N), target data set size, and individual tree size. The tree size is dictated by the number of nodes, features, and corresponding threshold values used in decision-making at each node. A layer, also referred to as a level or depth, refers to a group of nodes that are at the same distance from the root node in the tree. Here, the root node, i.e., the node where the tree begins, is considered the first layer. In the classification context, each tree in an RF model casts a vote for a specific class label when presented with an unseen data point. Here, the model allocates the data point to the class that accumulates the majority of votes. This aggregation of votes from multiple trees culminates in a more robust and accurate classification than a single decision tree [24].

The RF method, as described above, can be computationally expensive when running on resource-constrained IoT devices, which will affect both the inference latency and hardware accelerator footprint. Binarizing the operations in the RF model is effective in addressing the issue. To realize effective binarization, the feature values and decision thresholds must be quantized into discrete binary values. This quantization process can be performed using several techniques, e.g., uniform or nonuniform quantization [25], considering the data distribution and the desired tradeoff between computational efficiency and model accuracy. Once the quantization is complete, the resulting discrete values are encoded as binary representations, which facilitates the use of bitwise operations, e.g., the XOR and bit counting operations. When classifying an unseen data point, the binarized RF model traverses the decision trees using these bitwise operations to compare the binarized feature values with the binarized decision thresholds, and the outcome of these comparisons determines the path taken through each tree.

The binarized RF model can be implemented effectively on an FPGA as a dedicated accelerator. The inherent independence between the trees enables streamlined parallel inference for multiple decision trees, which improves the overall computational efficiency. In existing methods (e.g., [17]), within each tree, storage and computations are allocated for each layer to facilitate effective pipelining between layers. However, this method incurs considerable consumption of storage resources and necessitates time-intensive memory configuration for each layer, thereby hindering the deployment of RF models on resource-constrained devices.

B. Feature Selection

Feature selection is a critical process in ML classification tasks involving high-dimensional data By identifying and extracting highly important features, we can reduce computational costs, storage requirements, and training time and improve classification accuracy [26]. There are several common feature selection metrics for the RF method, e.g., information gain [27], the Gini index [18], and the OOB score [19]. The information gain metric quantifies the reduction in entropy or uncertainty after splitting a data set based on a specific feature. Here, for each feature, the information gain is determined according to the difference between the parent entropy and the weighted average of the children entropy. The Gini index, which is derived from the Gini impurity, measures the impurity of a node in a decision tree. It quantifies the likelihood of misclassifying a randomly selected instance from the data set if it were labeled according to the class distribution at a node. The OOB score leverages the unique structure of the ensemble learning method. During the construction of each decision tree, OOB instances (i.e., data instances that are left out of the bootstrapped samples[1]) can be utilized to estimate the generalization error and feature importance of the RF model without requiring a separate validation set.

Various of feature selection techniques (FSTs) based on these metrics have been studied previously. Among RF-based works, a representative study [29] selected important features according to the mean decrease Gini (MDG) and mean decrease accuracy (MDA) metrics, which can be obtained by calculating the average reduction in the Gini impurity and accuracy, respectively. This work demonstrated the effectiveness of leveraging both MDG and MDA, resulting in improvements to robustness and performance, as well as a reduction of inference time for high-dimensional data sets.

Later, FSTs were extended to EFSTs by integrating various metrics and methods. EFSTs can capture complex interactions among features from distinct importance perspectives to provide a more comprehensive and robust representation of feature importance. Existing EFSTs for IoT IDSs employ multiple metrics. For example, three filter-based feature ranking techniques and four supervised learning-based feature ranking techniques were combined in previous studies [4], [5] to obtain a reduced subset of 20 features. However, these approaches do not consider the resource-constrained requirements of IoT devices, which would demand fewer features to realize efficient processing and real-time detection.

Feature selection affects both the inference accuracy and computational/data requirements (i.e., the model's complexity). In terms of realizing a real-time IDS, striking an optimal balance between them is crucial to ensure practical application. Thus, to deploy an RF-based IDS on resource-constrained devices, an effective FST or EFST that identifies and retains the most relevant features must be utilized to reduce the model's complexity.

III. Proposed Codesigned RF-Based IDS on FPGA

In this section, we present the proposed real-time IDS by codesigning hardware/software for IoT devices, which must handle massive amounts of network traffic. As described in Fig. 2, the cyber intrusion targets IoT devices connected to

[1]Approximately one-third of the data instances tend to be categorized as OOB instances [28].

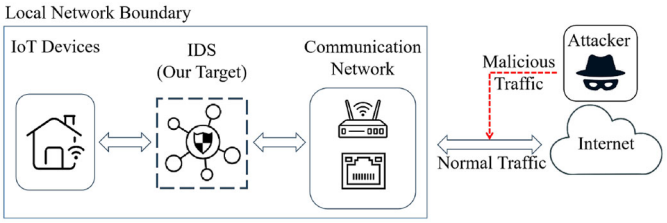

Fig. 2. Deployment of IDS in IoT environment.

the public Internet through the communication network. The IDS should be deployed on gateways or integrated into IoT devices to identify and address malicious traffic threats. Both gateways and IoT devices have limited resources to directly operate IDS with optimal tradeoffs.

In detail, balancing real-time and high-accuracy detection simultaneously is a major challenge for these resource-constrained devices. As mentioned previously, to date, ML algorithms (including the RF method) have been proven effective for malicious and normal traffic classification in the IoT context. Thus, a promising and realistic solution is to design a dedicated hardware accelerator for a lightweight ML classifier with reduced computational burden through model optimizations.

According to these motivations, the goal of this study is to hardware/software codesign a lightweight RF-based IDS that utilizes an effective EFST to reduce hardware resource utilization by reducing the dimensionality of the input data. For embedded systems applications, hardware/software codesign can simultaneously realize flexibility via software solutions and accelerate processing using hardware. Our software implementation involves two main components: 1) an EFST that incorporates the Gini index [18] and the OOB score [19] to select a limited number of useful features from the IoT network traffic and 2) a software feature extractor to extract the target features from the network traffic in real-time. The former is used offline (i.e., during RF training), and the latter is used online (i.e., it is deployed on the CPU of an FPGA SoC). For the hardware implementation, we present a lightweight and programmable accelerator architecture to flexibly implement the RF model in a memory-efficient manner. Finally, to realize a real-time IDS, we codesign the software feature extractor and the hardware accelerator on a heterogeneous Zynq SoC. Here, the software feature extractor and hardware accelerator exploit the processing system (PS) and programmable logic (PL) of the Zynq SoC, respectively.

In the following sections, we describe the EFST, the memory-efficient RF accelerator, and the IDS system codesigned on the Zynq SoC in detail.

A. Ensemble Feature Selection Technique

Network traffic is a representative example of high-dimensional data that involves a number of features (refer to Table IV in Section IV). For such data, feature selection metrics can effectively quantify the contribution of each feature to RF algorithms, compared to that of many features.

Algorithm 1 Proposed EFST Algorithm

Input: X: Feature matrix, y: Target vector, K: Number of features to select, T_O: OOB score threshold, T_G: Gini score threshold
Output: S_E: Selected feature set

1: Train an RF classifier on X and y
2: **for** each feature subset f_i in the RF classifier **do**
3: Calculate the Gini score $scoreGini(f_i)$ for the feature subset f_i by the formulae (1) and (2)
4: **end for**
5: **for** each feature subset f_i in X **do**
6: Train a new RF classifier on f_i and y
7: Calculate the OOB score $scoreOOB(f_i)$ for the feature subset f_i by the formula (3)
8: **end for**
9: $S_E \leftarrow \emptyset$
10: $count \leftarrow 0$
11: **for** each feature j in X **do**
12: **if** $scoreOOB(f_i) > T_O$ and $scoreGini(f_i) > T_G$ **then**
13: $S_E \leftarrow S_E \cup f_i$
14: $count \leftarrow count + 1$
15: **end if**
16: **if** $count == K$ **then**
17: break
18: **end if**
19: **end for**
20: **return** S_E

In this study, we focused on the Gini index and OOB score as the feature selection metrics to integrate into the EFST. As discussed in Section II-B, these metrics are based on distinct perspectives to quantify the feature importance, i.e., the misclassification likelihood by the Gini index and the generalization error by the OOB score. Thus, the proposed EFST can extract useful and robust features to construct and train a lightweight but sufficiently accurate RF model (Section III-B). Existing EFSTs (e.g., [4] and [5]) did not reduce the number of features significantly (e.g., 20 features) because they only focused on the model's classification performance. In other words, the runtime was not considered. In addition, they targeted a specific intrusion threat (e.g., reconnaissance detection in [4] and information theft in [5]). In contrast, the proposed EFST aggressively reduces the number of selected features and covers a wider variety of IoT threats (Table I).

The Proposed EFST: The proposed EFST algorithm is given in Algorithm 1. The goal of this algorithm is to assess the impact of each feature on the performance of the model. This algorithm first trains the RF model on the training data set X and labels y. Then, we obtain the feature importance according to the Gini index (lines 1–4). We then train a new RF model with each feature subset and target label y, and we obtain the feature importance according to the OOB score (lines 5–8). In RF, the Gini importance index value (hereafter referred to as the Gini score) for a feature is obtained by the sum of the Gini index reduction over all nodes where the specific feature

is used to split. Specifically, given the number of features n and the ith feature f_i ($1 \leq i \leq n$), the Gini score for feature f_i (scoreGini(f_i)) is calculated as follows:

$$\text{scoreGini}(f_i) = \sum_{j=1}^{k} \text{Imp}(f_i, C_j) \tag{1}$$

where k is the number of classes, C_j is the jth class, and $\text{Imp}(f_i, C_j)$ is the impurity of feature f_i for class C_j. The impurity is computed as follows using the Gini impurity formula:

$$\text{Imp}(f_i, C_j) = 1 - \sum_{m=1}^{k} P_{m|C_j}^2 \tag{2}$$

where $P_{m|C_j}$ is the proportion of samples from class C_j that are classified as class m using feature f_i.

Note that no single decision tree generation process uses all of the samples. Here, we refer to the unused samples as OOB samples, which can be utilized to evaluate the accuracy of the tree. The OOB score for each feature is evaluated iteratively, where a higher score indicates higher feature importance. The OOB score for feature f_i (scoreOOB(f_i)) can be obtained as follows:

$$\text{scoreOOB}(f_i) = \frac{1}{|\text{OOB}_{f_i}|} \sum_{t \in \text{OOB}_{f_i}} I\left[y_t = \hat{y}_t(f_i)\right] \tag{3}$$

where y_t is the true label of the tth sample, $\hat{y}_t(f_i)$ is the predicted label of the tth sample by the RF model trained using feature f_i, function $I[]$ counts the number of times the equation in square brackets becomes true, and $|\text{OOB}_{f_i}|$ denotes the number of samples not used to construct the trees referring to feature f_i to split.

By setting different thresholds for these scores, we can flexibly identify a final set of features that are consistently important according to both evaluation metrics (lines 11–19).

The Feature Selection Process: When implementing the EFST to reduce the number of network traffic features, we initially employ the RF model for training on the network traffic data set. In this process, the RF model assesses the impurity change of each feature during node partitioning using the Gini index to gauge its importance. The Gini index quantifies a feature's role in reducing the node impurity, thereby measuring its contribution to the overall model. It primarily focuses on the misclassification risk. Subsequently, we train a new RF model using each feature subset and target label, measuring feature importance through the OOB score. The essence of this process lies in iteratively evaluating the contribution of each feature, ultimately resulting in a set of consistently important features.

The reduction of features yields dual advantages. First, it diminishes data dimensionality, thereby accelerating model training and rendering it better suited for real-time applications. Second, EFST guarantees that the selected features are both useful and robust, leading to effective simplification of the model structure and an enhancement of its generalization capacity to unobserved data.

B. Hardware Accelerator for RF Classifier

In RF, individual decision trees can be evaluated independently in their reasoning process, thereby making them well-suited for parallelized acceleration on an FPGA [17], [20], [30], [31]. In most previous studies, the decision tree data comprising tree structures, decision nodes, and leaf nodes is stored in on-chip memory to allow multiple processing elements (PEs) perform the inference task in parallel. Although this memory-oriented approach enables efficient use of other types of available hardware resources, without a careful decision regarding the parallelized granularity, memory consumption can easily become a dominant factor when selecting a deployable FPGA board. In fact, existing memory-hungry architectures, e.g., layer-wise parallelization in RF-RISA [17], may limit their applicability to resource-constrained environments.

The proposed RF accelerator resolves this memory bottleneck issue by allocating block RAM (BRAM) resources at a coarser granularity to achieve effective resource utilization. Specifically, we allocate BRAMs to individual decision trees rather than individual layers, which results in a significant reduction in the total number of BRAMs required for the entire accelerator. Here, we consider an RF model with N trees and maximum depth M. While layer-wise parallelization approaches (e.g., RF-RISA) require $N \times M$ BRAMs to store this model, the proposed approach only requires $N \times X$ BRAMs to store the same model, where X is the number of BRAMs needed to store a single decision tree. Although X is correlated with the total number of nodes in a single tree and thus can be up to $2^{(M+1)} - 1$, we can search data efficiently without exploring all nodes. Specifically, two 36 kB BRAMs can store sequence data with the size of depth 2,048 and a width of 36 bits. Note that this capacity is sufficient to accommodate the information of all decision nodes in decision trees with a depth of 10 or less. Thus, in practice, the number of decision nodes is significantly less than the theoretical upper bound ($2^{(M+1)} - 1$), which means that X can be two or three in most cases, thereby significantly reducing BRAMs. This optimized BRAM allocation scheme enhances the efficiency of the accelerator and helps overcome the resource constraints associated with RF models implemented on low-end FPGAs.

In the following, we elaborate on the design of our accelerator. Fig. 3(a) shows an example of a single decision tree with a maximum depth of two. It contains one root node, two internal nodes, and four leaf nodes. Fig. 3(b) shows the corresponding memory layout. Here, all the nodes are stored in the instruction format defined in Fig. 3(c). Fig. 3(d) describes the accelerator architecture of the corresponding decision tree.

Memory Layout: In our accelerator, one MEM is used to store all of the instruction sets of a single decision tree (e.g., the node information, node addresses, and tree structure) and is read by the PEs to perform the reasoning tasks. The memory layout is designed based on the depth-first traversal order of the decision tree during the actual inference process. All decision nodes are stored in a 1-D array format. Fig. 3(b) shows the memory layout for a balanced decision tree with a depth of two. In practice, decision trees are rarely balanced

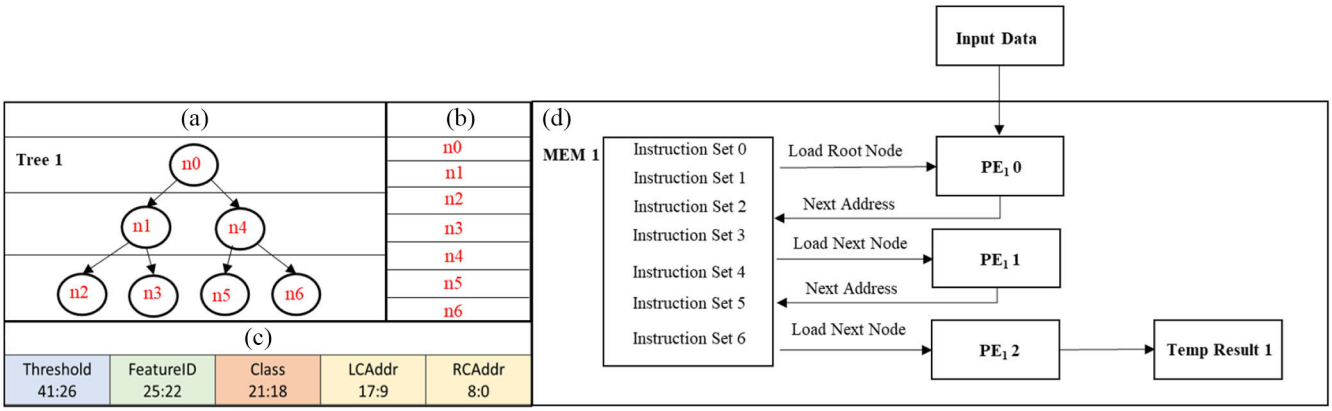

Fig. 3. Decision tree model and overview of the corresponding accelerator. (a) Decision tree model. (b) Memory layout. (c) Node instruction structure. (d) Hardware accelerator.

TABLE II
VALUES AND LABELS OF "CLASS" AND "FEATUREID"

Component	Value	Label
Class	0000	DDoS/DoS
	0001	Normal
	0010	Reconnaissance
	0011	Information Theft
FeatureID	0000	Bytes
	0001	Dbytes
	0010	Dpkts
	0011	Drate
	.....	.....

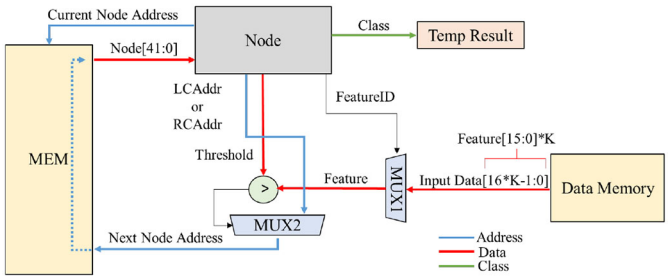

Fig. 4. Internal PE structure.

perfectly. In addition, if a substantial number of decision nodes are pruned, it is unnecessary to allocate storage space for the pruned nodes.

Instruction Format: The width of the data stored in BRAMs is defined by the format shown in Fig. 3(c). As can be seen, the instruction comprises five components. Here, "LCAddr" and "RCAddr" represent the addresses of the left child and right child, respectively, which guide the decision tree nodes to the next node. For the leaf nodes [$n2, n3, n5$, and $n6$ in Fig. 3(a)], "`9'b00000000`" is set to LCAddr and RCAddr. "Class" represents the reasoning category of the current node, and "FeatureID" is the label of the specific feature used for comparison. The values and labels of "Class" and "FeatureID" are shown in Table II. "Threshold" holds the threshold value of the decision tree node used for comparison with the input feature. Among these five components, only the threshold values can be set as a floating-point number. As briefly mentioned in Section II-A, we performed fixed-point quantization on the threshold value. The quantization formula is defined as follows. Here, x is the threshold value, and round() is a rounding function that maps the result to the nearest integer value. The range of threshold values $[a, b]$ is mapped to the range of possible integer values $[0, 2^k - 1]$, where k is the number of bits used to represent the integer value ($k = 16$ in our case)

$$q = \text{round}\left(\frac{(x-a) \times \left(2^k - 1\right)}{b-a}\right). \tag{4}$$

Hardware Accelerator: The architectural overview of a single decision tree is shown in Fig. 3(d). From a high-level perspective, the PEs are responsible for evaluating the decision criteria at each node and directing the data flow accordingly. The processing in the PEs is described in the following paragraph. These PEs are performed in a pipelined manner to produce N results in parallel, which are then input to a majority voter to output an inference result. Here, N represents the number of decision trees; thus, the decision tree node data (described as "Instruction Set") is stored in N memory blocks (i.e., MEMs). The input data are fed to the accelerator in a streaming manner, which allows for efficient pipelining of the PEs. By continuously supplying input data to the PEs, their processing capabilities can be fully utilized, further enhancing performance.

The internal PE structure is shown in Fig. 4. In the PE, the decision tree node first loads node information from the MEM through the address of the root node. Here, the FeatureID is used to determine the specific feature from the input data. Since the input data are composed of K features, each represented by a 16-bit value, the total bit-width is $16 \times K$ [the number of input features K is determined using the proposed EFST (Section III-A)]. The corresponding 16-bit feature width is extracted from the input data using a multiplexer (MUX1) based on the FeatureID. Then, this feature and the corresponding threshold value (stored in the node information) are input to the comparator for comparison. Beginning with the most significant bit, the XOR operation is performed to compare the corresponding bits of the feature and threshold. If XOR

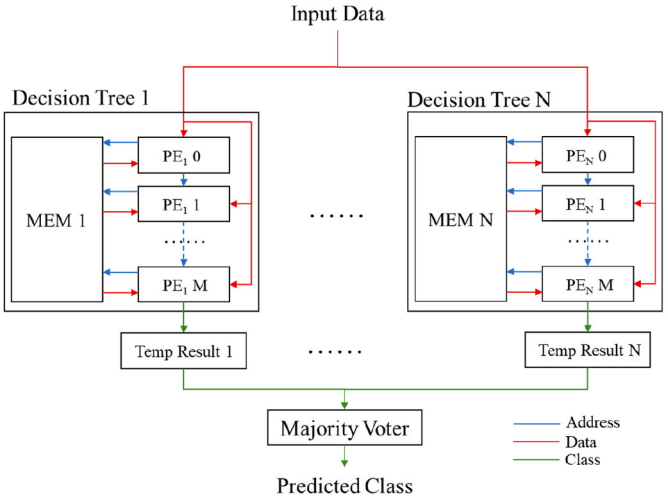

Fig. 5. Overview of RF accelerator architecture.

yields "1", the number with "1" at that position is larger. The comparison continues along subsequent bits until a difference is found, ultimately determining whether or not the feature is greater than the threshold value. Then, according to the comparison result, the next address to be accessed (i.e., the left child address or the right child address) is determined to read the next node information from the MEM. When all decision nodes finish the comparisons, the Class stored in the leaf node information represents the classification result of the tree [i.e., Temp Result 1 in Fig. 3(d)].

Fig. 5 shows an overview of our RF accelerator. Note that all of the Temp Results are input to a majority voter to output the final classification result. Here, each bit position is assessed independently. The most frequent bit value at each position is selected for the output, resulting in a majority-voted binary representation. For instance, if three decision trees predict the classes 0011, 0010, and 0011 for a particular input, the majority vote would yield the final prediction as class 0011. For an RF model comprising N trees with a maximum depth of M, the accelerator has a total of $N \times M$ PEs, which are configured using slices (in Xilinx FPGAs). In addition, the accelerator has N MEMs (with a data width of 42 bits) and N data memories (with a data width of 16 bits for each of the K features).

In summary, our accelerator is lightweight in memory. It requires $N \times X$ BRAMs for the MEMs and $N \times K \times 2$ bytes for the data memories (i.e., the data width is 16 bits), where X and K are effectively reduced by the tree-wise memory allocation and our EFST (Section III-A), respectively. The accelerator is also lightweight in terms of computations because the RF model is binarized to reduce computational complexity.

C. FPGA-Based IDS

As shown in Fig. 6, we implemented the proposed IDS on a heterogeneous Zynq SoC. We designed the RF hardware accelerator in Verilog HDL and then mapped it on the PL part of the Zynq. The software-implemented feature extractor was processed on the PS part by an ARM Cortex-A9 with

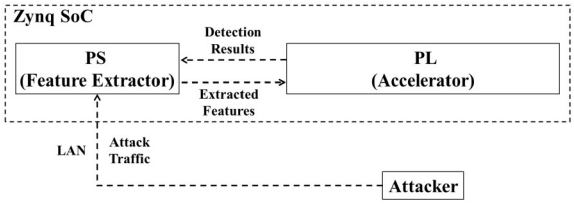

Fig. 6. Proposed IDS deployed on the Zynq platform.

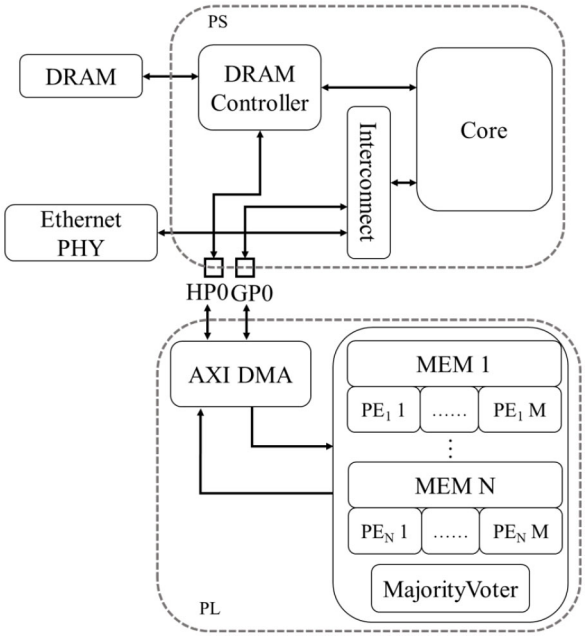

Fig. 7. Overview of the proposed codesigned IDS.

a clock frequency of 667 MHz. Here, the feature extractor communicates with external devices to receive the network traffic through a PHY Ethernet LAN. It also proceeds the received network traffic to extract and preprocess the required features (obtained by our EFST) for input to the accelerator. Then, the accelerator processes the extracted input features and produces the detection results, which are returned to the PS side. The entire system was designed using Xilinx Vivado 2021.1.

Fig. 7 shows a more detailed view of the proposed codesigned IDS. The PS and PL parts are interfaced through the advanced extensible interface direct memory access (AXI DMA) module. The AXI DMA supports the AXI stream protocol for high-speed data transfer, and connects the CPU core on the PS part and the accelerator on the PL part through the AXI dedicated ports ("HP0" as a slave port and "GP0" as a general-purpose port). The DRAM controller on the PS part is used to read the required data from the DRAM that stores the extracted features.

IV. Evaluation

In this section, we first explain our experimental setup and the metrics used to evaluate the proposed IDS. We then evaluate the results quantitatively.

TABLE III
EXTRACTED BOT-IOT DATA SET

Category	Original size	Extracted size
DDoS/DoS	71,537,674	20,000
Reconnaissance	1,821,639	20,000
Information Theft	1,587	1,587
Normal	9,543	9,543

TABLE IV
EXTRACTED 15 FEATURES FROM THE PCAP DATA IN THE BOT-IOT DATA SET (THE GINI INDEX IN RATIO % AND THE OOB SCORE IN ACCURACY %)

Feature	Description	Gini	OOB
Bytes	Total number of bytes in transaction	18.79	98.41
Dbytes	Destination-to-source byte count	1.56	65.62
Dpkts	Destination-to-source packet count	0.64	59.19
Drate	Destination-to-source packets per second	0.48	44.65
Dur	Record total duration	14.10	93.71
Max	Maximum duration of aggregated records	3.56	83.69
Mean	Average duration of aggregated records	2.56	83.37
Min	Minimum duration of aggregated records	0.89	78.46
Pkts	Total count of packets in transaction	8.44	82.26
Rate	Total packets per second in transaction	5.43	93.64
Sbytes	Source-to-destination byte count	20.30	98.23
Spkts	Source-to-destination packet count	9.10	87.26
Srate	Source-to-destination packets per second	7.55	88.76
Stddev	Standard deviation of aggregated records	3.20	76.65
Sum	Total duration of aggregated records	3.40	84.43

A. Setup

To demonstrate the effectiveness of the proposed methods in realistic IoT network environments, we conducted two evaluations. We first evaluated the proposed EFST (Section III-A), and then we evaluated the proposed codesigned IDS with the RF accelerator (Sections III-B and III-C). Note that we utilized the BoT-IoT data set in both of these evaluations. The BoT-IoT data set, which was released by the University of New South Wales in 2019, is a specialized intrusion detection data set that serves as a realistic representation of BoT network attacks on IoT devices [21]. It includes instances of normal traffic and four categories of attacks, i.e., DDoS, DoS, reconnaissance, and information theft. When the traffic of a DoS attack comes from multiple sources, it is referred to as a DDoS attack. Due to their similarity in attack mechanisms, DoS and DDoS are frequently studied together in terms of detection and defense methods [21], [32], [33], [34], [35]. Thus, we handle them together in our evaluations.

The full BoT-IoT data set processed by the Argus network security tool[2] comprises 73 million instances with 29 features. As shown in Table III, the data for the categories are heavily imbalanced; thus, we downsampled the number of instances in each category to 20 000, except for the normal and information theft categories [23]. Among the 29 features in the data set, 15 network flow features (Table IV) can be extracted from packet capture (PCAP) data through network flow analysis and thus can be utilized for real-time inference by IDSs. The extracted features of each record are then transformed into floating-point values (ranging between 0 and 1) using min-max normalization. Hereafter, these 15 features are considered a full set of features.

TABLE V
CONFUSION MATRIX

	Attack	Normal
Attack	True positives (TP) (i.e., correctly predicted attack samples)	False negatives (FN) (i.e., incorrectly predicted normal samples)
Normal	False positives (FP) (i.e., incorrectly predicted attack samples)	True negatives (TN) (i.e., correctly predicted normal samples)

For the first evaluation (Section IV-B), we implemented our RF model using scikit-learn (sklearn)[3] to evaluate the detection performance. By varying the threshold parameters T_G and T_O to extract important features from the 15 features in Table IV, we identified a set of parameters to develop a lightweight (i.e., as few trees as possible) RF hardware accelerator while achieving good detection performance. Here, the number of trees N and the maximum depth of trees M were set to $N = 6$ and $M = 11$. In addition, the accuracy, precision, recall, and F1 score metrics were used to evaluate the RF model, and the confusion matrix in Table V defines these evaluation metrics.

1) *Accuracy:* The ratio of correctly classified normal and attack samples to the total number of samples.

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

2) *Precision:* The percentage of successful predictions.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}.$$

3) *Recall:* The ratio of correctly classified attack samples.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

4) *F1 Score:* A composite metric that considers both precision and recall metrics to evaluate the extent to which the model detects attacks successfully.

$$F1 = 2 \cdot \frac{\text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}}.$$

In the second evaluation (Section IV-C), according to the results of the first evaluation, we selected the appropriate features and developed the corresponding RF hardware accelerator in Verilog HDL. Here, the online inference time and circuit area (i.e., the FPGA resource utilization) of the IDS (Section III-C) were evaluated for the PYNQ Z2 (Xc7z020clg400) using Xilinx Vivado 2021.1.

B. Results of Our Proposed EFST

Fig. 8 shows the Gini ratio and OOB score for each feature in the red and blue bars, respectively. With the cumulative Gini ratios equating to 100%, a more substantial ratio underscores the increased significance of the feature. A higher OOB score reflects the augmented importance of the given feature.

[2]https://openargus.org

[3]https://scikit-learn.org

TABLE VI
THRESHOLD SETTINGS FOR RF-EFST-#

	T_G	T_O	Selected features
RF-EFST-2	0.15	0.95	Sbytes, Bytes
RF-EFST-3	0.10	0.90	Sbytes, Bytes, Dur
RF-EFST-6	0.05	0.85	Sbytes, Bytes, Dur, Spkts Srate, Rate

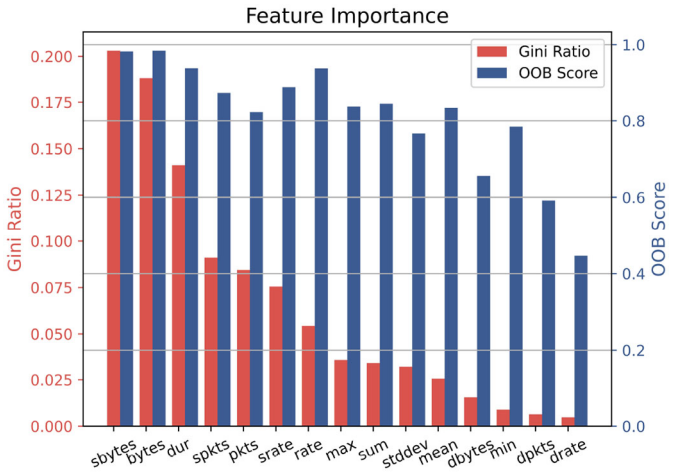

Fig. 8. Feature importance ranking.

According to the results shown in Fig. 8, we conducted the first evaluation for the following three methods. For all methods, we performed four-class training and testing on the extracted BoT-IoT data set (Table III).

1) *RF- Full (Baseline):* An RF model using all 15 features in Table IV (i.e., feature selection was not performed). This model is considered the baseline model.
2) *RF-EFST-#:* An RF model with the proposed EFST. Here, the suffix "#" indicates the number of features selected by different threshold settings T_G and T_O. In our evaluation, we varied these thresholds as summarized in Table VI to generate three sets of selected features.
3) *RF-EFST-#-q:* A 16-bit quantized version of the best RF-EFST to confirm how the quantization process affects the detection performance.

First, we compared the features selected according to different threshold values as shown in the fourth column of Table VI. Considering that anomaly intrusion is abnormal communication from the source-to-destination, intuitively, most of the selected features are relevant to source-to-destination information. In contrast, the destination-to-source features were ranked lower by both Gini ratio and OOB score metrics. In this evaluation, only three sets of thresholds were given to the proposed EFST because selecting additional low-importance features (as determined by the two metrics) would not benefit a lightweight RF model.

We then investigated the classification performance of the compared RF methods on the BoT-IoT data set as described in "Test set" columns of Table VII. By using all 15 features, the RF-full achieved the best classification performance with 99.14% accuracy. Interestingly, even by reducing the features from 15 to three or six, we found that the performance did not change. However, further reducing the features down to two resulted in a performance degradation of 0.2% on average.

Furthermore, to deepen our insights into the model's generalization capability, we also conducted cross-validation as described in the right half of Table VII. Across fivefold cross-validation, the model's performance remains at a high level. The uniformity observed across various evaluation metrics indicates that the model demonstrates robustness in classification across diverse categories. The alignment between the outcomes of the test set and those of cross-validation is pivotal in mitigating the risk of overfitting. Models are evaluated across a broader range of data subsets, reducing over-reliance on specific features of the training set and enhancing generalization performance. This indicates that the model can consistently perform well when presented with previously unseen data.

Recall that selecting features more aggressively (i.e., selecting fewer features) enables more efficient real-time IDSs, the RF-EFST-3 is considered the best method. We then applied quantization to the RF-EFST-3 method to realize the RF-EFST-3-q in the table, which demonstrated a slight decline in accuracy and F1-score (0.46% and 0.45%, respectively). One may think that the RF-EFST-2 is better than the RF-EFST-3-q; however, applying quantization to the RF-EFST-2 would lead to a further large decline in performance. Thus, from the first evaluation, we conclude that the RF-EFST-3-q is a suitable model upon which our hardware/software codesigned real-time IDS should be implemented on an FPGA.

C. Results of Hardware Implementation

To demonstrate the effectiveness of the proposed codesigned RF-based IDS, we first compared the following five accelerators designed for the same FPGA board in terms of resource utilization.

1) *NN2019 [9]:* An NN accelerator working with a clock frequency of 76 MHz. The design of this accelerator is flexible and can be updated to adapt to emerging attacks.
2) *NN2021 [10]:* An NN accelerator working with a clock frequency of 100 MHz. Here, a hierarchical decision-making approach is taken for real-time IoT network intrusion detection.
3) *NN2023 [36]:* An NN accelerator working a clock frequency of 100 MHz. This accelerator was extended from NN2021 for a heterogeneous hardware-based network intrusion detection framework.
4) *RF-Full-q:* A 16-bit quantized RF accelerator using all 15 features in Table IV (i.e., the quantized version of RF-full).
5) *RF-EFST-q:* A 16-bit quantized version of the proposed RF model using only three features selected by the EFST (i.e., RF-EFST-3 in the first evaluation. This accelerator works with a clock frequency of 100 MHz).

The results are shown in Table VIII, where the numbers in the parentheses in the first row represent the available number of resources of each type on the chip Xc7z020clg400 (containing a total of 53 200 LUTs, 106 400 FFs, 140 BRAMs, 220 DSP slices, and 125 Bonded IOBs). Here, "N/A" indicates

TABLE VII
COMPARISONS OF CLASSIFICATION RESULTS (IN %) ON THE BOT-IOT DATA SET

Model	Test set				Cross validation			
	Accuracy	Precision	Recall	F1 score	Accuracy	Precision	Recall	F1 score
RF-full	99.14	99.14	99.14	99.13	99.10	99.11	99.10	99.10
RF-EFST-2	98.95	98.94	98.94	98.94	98.98	99.00	98.98	98.98
RF-EFST-3	99.14	99.15	99.14	99.13	99.09	99.10	99.09	99.09
RF-EFST-6	99.09	99.10	99.09	99.09	99.08	99.09	99.08	99.08
RF-EFST-3-q	98.68	98.68	98.68	98.68	98.99	99.00	98.99	98.99

TABLE VIII
COMPARISON OF RESOURCE UTILIZATION ON THE XILINX PYNQ Z2 (XC7Z020CLG400)

Accelerator	LUTs (53,200)	FFs (106,400)	DSPs (220)	BRAMs (140)	LUTRAMs (17,400)	Bonded IOBs (125)
NN2019	26,463	56,478	111	44	N/A	N/A
NN2021	11,603	16,461	184	61	242	N/A
NN2023	28,004	33,974	219	20	1,412	N/A
RF-full-q	6,993	3,436	0	30	0	245
RF-EFST-q	3,653	2,811	0	15	0	53

TABLE IX
CLASS-BASED F1 SCORE (IN %) COMPARISON TO STATE-OF-THE-ART SYSTEMS ON THE BOT-IOT DATA SET

System	No. of features	DDoS/DoS	Normal	Reconnaissance	Information Theft
ProtÉdge [23]	16	94.50	N/A	72.50	35.00
Lawal et al. [40]	10	100.00	96.00	96.41	92.00
Ullah at al. [33]	64	99.92	99.73	99.76	99.75
Our system	3	100.00	96.55	98.46	97.60

that the corresponding result was not available from the publication. By comparing the existing NN-based accelerators and RF-full-q, we found that the latter reduced resource utilization significantly. Note that no DSP was used due to the binarized operations. In contrast, the required IOBs exceeded the number of available IOBs on the PYNQ-Z2. This was caused by taking 15 features as input, meaning that a larger FPGA would be required to deploy RF-full-q. Among the compared accelerators, the proposed accelerator (i.e., RF-EFST-q) was the most lightweight for all resource types. For the logic configuration, the LUTs were the most used on-chip resource; however, this represented only 6.97% utilization. For the on-chip memory configuration, the proposed accelerator (i.e., RF-EFST-q) consumed 10.71% BRAMs. Compared to the RF-full-q, the RF-EFST-q reduced the complexity of the logic design by selecting a smaller subset of features at each decision node. This reduction in complexity can lead to more efficient hardware implementations. In addition, the RF-EFST-q requires fewer BRAMs to store all decision trees because reducing the number of features can lead to simpler trees with fewer decision nodes [37]. Considering that the PYNQ-Z2 is one of the smallest FPGA boards used for IoT devices, the proposed accelerator can potentially be deployed on even smaller boards. This demonstrates the usefulness of our work.

Next, we compared our implemented system against with state-of-the-art systems in terms of classification performance on the BoT-IoT data set. It is crucial to highlight that our hardware implementation exhibited identical performance to the quantized RF model (i.e., RF-EFST-3-q). Converting the quantified RF model into our hardware accelerator on the FPGA development board does not cause any loss in classification performance. Table IX elaborates the class-based F1 score for four distinct classes, including DDoS/DoS, Normal, Reconnaissance, and Theft. Our system achieved 100.00% in identifying instances related to DDoS/DoS attacks, underscoring its robust accuracy. In the Normal category, the system demonstrated a strong score (96.55%), indicating its effective discrimination of normal network behavior. The Reconnaissance and Theft categories also exhibited balanced and robust performance. Compared with state-of-the-art systems, our system achieves a comparable performance in multicategory network traffic threat detection, consolidating its effectiveness in complex network environments. Furthermore, another highlight is that to achieve this performance, our system utilizes only three features, which is much fewer than the features used by the other systems. The reduced number of features reflects our concise and efficient design in feature engineering, which helps improve system performance and mitigate the risk of overfitting.

Delving further into these results, our system exhibits performance differences in multicategory network intrusion detection. The consistent high detection performance of DDoS/DoS attacks is attributed to their distinct and regular features. Contrary, our system falls slightly short behind state-of-the-art systems in the Normal, Reconnaissance, and Information Theft categories due to different reasons; Recognizing the diversity of normal network behavior is challenging since this diversity results in differences between features. Also, the uneven distribution of normal samples contributes to the performance decrease; Reconnaissance attacks pose difficulty to detect due to their covert nature [4].

TABLE X
COMPARISON OF INFERENCE TIME

Model	Platform	Clock (MHz)	Time (ms)
RF-EFST-3-a9	ARM Cortex-A9	667	145.24
RF-EFST-3-i7	Intel Core i7	2,800	10.97
NN2019	FPGA (Zynq Z-7020)	76	9.02
NN2021	FPGA (PYNQ Z2)	100	0.44
NN2023	FPGA (PYNQ Z2)	100	2.24
RF-EFST-q	FPGA (PYNQ Z2)	100	1.07

Our system may not have ideally captured the covert signatures of the Reconnaissance attack; The lack of sufficient Information Theft samples [23] impacts our system's ability to adequately learn and identify these attacks, resulting in performance degradation. These findings underscore the complexities of accurately identifying various types of attacks in multicategory network intrusion detection. To address these challenges, exploring data augmentation [38] and ensemble learning [39] could be promising. These approaches can enhance the robustness of the training data set and leverage the strengths of multiple models to achieve more accurate detection outcomes.

Lastly, we evaluated the inference time of the proposed IDS when deployed on the PYNQ-Z2. Here, to facilitate a fair comparison, we performed the inference task on 22 544 records (similar to previous studies [9], [10], [36]). In this evaluation, in addition to the above four accelerators, we also compared a few software implementations of IDSs on an embedded microprocessor (i.e., an ARM Cortex-A9, which is deployed on the PYNQ-Z2) and a desktop PC (with an Intel Core i7 CPU). Specifically, we evaluated a software implementation of the RF-EFST-3 for the ARM Cortex-A9 and Core i7 (hereafter RF-EFST-3-a9 and RF-EFST-3-i7, respectively). Through this evaluation, we verified the effectiveness of parallelization in our hardware accelerator.

The results are shown in Table X. As can be seen, the software implementations on both the Cortex-A9 and Core i7 struggled with real-time intrusion detection even with the benefit of a high clock frequency and the feature reduction by the proposed EFST. The inference times of RF-EFST-3-a9 and RF-EFST-3-i7 were 145.24 ms and 10.97 ms, respectively. In contrast, despite the lower clock frequency on the FPGA (i.e., even 28× lower than the Core i7), FPGA accelerators can detect intrusions more quickly because they all enjoy rich parallelism inherent in the ML algorithms (both NN and RF). Compared to the existing FPGA accelerators, the proposed method (i.e., RF-EFST-3-q) achieved a shorter inference time than the NN2019 and NN2023 methods. Specifically, the inference time of RF-EFST-3-q was 8.43× and 2.09× (1.07 ms compared to 9.02 ms and 2.24 ms) faster than NN2019 and NN2023 methods, respectively. In addition, the inference time of the proposed method was comparable to that of the NN2021 method. Note that although RF-EFST-3-q was slightly slower, its inference time is sufficient for practical application in industrial IoT (IIoT) contexts. In a typical IIoT setting, sensors may be connected via low power wide area networks, e.g., LoRaWAN or Narrowband IoT [41]. These networks were designed for low power consumption and long-range communication at the cost of limited bandwidth and higher latency compared to traditional networks.

For example, given that the average latency for data transmission in an IIoT network is 100 ms and the available data rate is 50 Kb/s [42], we can calculate the number of records that can be transmitted within the latency constraint as follows. First, for RF-EFST-3-q, we set the size of each record to 48 bits because we have 16 bits for each of the three features

$$\text{Record size (bits)} = 48.$$

We then calculate how many records can be transmitted during the data transmission latency

$$\begin{aligned}\text{Records transmitted in 100 ms} &= \frac{(\text{Data rate} \times 0.1\ \text{s})}{\text{Record size}} \\ &\approx \frac{(50\,000 \times 0.1)}{48} \\ &\approx 104\ \text{records}.\end{aligned}$$

Recall that the proposed IDS can process 22 544 records in 1.07 ms, this result satisfies the constraints of the network's latency and bandwidth constraints and ensures real-time detection.

D. Discussion of Limitations and Future Direction

While our system has demonstrated remarkable performance against current cyber threats, it has limitations in handling ever-evolving attacks. For example, since our system would confront the issue of data timeliness, regular updates to the IoT network traffic data set will need to be considered. Through such updates, we can effectively adapt to the dynamic threat landscape and enhance the system's performance in addressing emerging attacks. Future research can concentrate on advanced deep learning techniques customized for resource-constrained IoT environments, aiming to better capture latent features and enhance detection performance for emerging threats.

Our work has made progress in deploying IoT IDS on FPGA without due consideration for data privacy protection. In sensitive industrial or medical applications, it is essential to carefully consider data privacy. To address this concern, integrating privacy-aware ML techniques (e.g., federated learning [43], [44]) into our system will provide a comprehensive approach toward privacy protection. Federated learning allows model training between distributed data sources without directly sharing the original data, thus protecting the privacy of the data. By incorporating privacy protection techniques, we can make our system better suited for handling sensitive data and enhance its practical applicability.

V. CONCLUSION

In this article, we have proposed a hardware/software codesigned IDS system based on a lightweight RF classifier. By applying our designed EFST, which aggressively reduces the number of selected network traffic features, the proposed method obtains high detection accuracy while reducing computational complexity (i.e., both hardware resources and detection

time). In an experimental evaluation, with only three out of 15 features extracted from the PCAP data, we achieved 99.14% accuracy on the BoT-IoT data set, which is equivalent to the accuracy obtained using all 15 features. The corresponding IDS demonstrated sufficiently low resource utilization on the PYNQ-Z2 to address the issue of substantial memory usage found in state-of-the-art IDS accelerators. By successfully addressing the unique challenges associated with high-dimensional data and constrained processing capabilities, these results demonstrate that the proposed method is suitable for real-time intrusion detection on resource-limited IoT devices.

REFERENCES

[1] W. Iqbal, H. Abbas, M. Daneshmand, B. Rauf, and Y. A. Bangash, "An in-depth analysis of IoT security requirements, challenges, and their countermeasures via software-defined security," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10250–10276, Oct. 2020.

[2] M. Zeeshan et al., "Protocol-based deep intrusion detection for DoS and DDoS attacks using UNSW-NB15 and Bot-IoT data-sets," *IEEE Access*, vol. 10, pp. 2269–2283, 2022.

[3] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other Botnets," *Computer*, vol. 50, no. 7, pp. 80–84, Jul. 2017.

[4] J. L. Leevy, J. Hancock, T. M. Khoshgoftaar, and N. Seliya, "IoT reconnaissance attack classification with random undersampling and ensemble feature selection," in *Proc. 7th Int. Conf. Collab. Internet Comput.*, 2021, pp. 41–49.

[5] J. L. Leevy, J. Hancock, T. M. Khoshgoftaar, and J. M. Peterson, "IoT information theft prediction using ensemble feature selection," *J. Big Data*, vol. 9, no. 1, pp. 1–48, 2022.

[6] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, pp. 5–32, Oct. 2001.

[7] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[8] P. A. A. Resende and A. C. Drummond, "A survey of random forest based methods for intrusion detection systems," *ACM Comput. Surveys*, vol. 51, no. 3, pp. 1–36, 2018.

[9] L. Ioannou and S. A. Fahmy, "Network intrusion detection using neural networks on FPGA SoCs," in *Proc. 29th Int. Conf. Field Program. Logic Appl.*, 2019, pp. 232–238.

[10] D.-M. Ngo, A. Temko, C. C. Murphy, and E. Popovici, "FPGA hardware acceleration framework for anomaly-based intrusion detection system in IoT," in *Proc. 31st Int. Conf. Field-Program. Logic Appl.*, 2021, pp. 69–75.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[12] M. Shafiq, Z. Tian, Y. Sun, X. Du, and M. Guizani, "Selection of effective machine learning algorithm and Bot-IoT attacks traffic identification for Internet of Things in smart city," *Future Gener. Comput. Syst.*, vol. 107, pp. 433–442, Jun. 2020.

[13] Y. Saeys, T. Abeel, and Y. Van de Peer, "Robust feature selection using ensemble feature selection techniques," in *Proc. Mach. Learn. Knowl. Discov. Databases Eur. Conf.*, 2008, pp. 313–325.

[14] D. Danopoulos, C. Kachris, and D. Soudris, "Acceleration of image classification with Caffe framework using FPGA," in *Proc. 7th Int. Conf. Modern Circuits Syst. Technol.*, 2018, pp. 1–4.

[15] X. Zhang et al., "SkyNet: A champion model for DAC-SDC on low power object detection," 2019, *arXiv:1906.10327*.

[16] S. Fujimaki, Y. Inoue, D. Hisano, K. Maruta, Y. Nakayama, and Y. Hara-Azumi, "A self-attention network for deep JSCCM: The design and FPGA implementation," in *Proc. IEEE Glob. Commun. Conf.*, 2022, pp. 6390–6395.

[17] S. Zhao, S. Chen, H. Yang, F. Wang, and Z. Wei, "RF-RISA: A novel flexible random forest accelerator based on FPGA," *J. Parallel Distrib. Comput.*, vol. 157, pp. 220–232, Nov. 2021.

[18] M. Pal, "Random forest classifier for remote sensing classification," *Int. J. Remote Sens.*, vol. 26, no. 1, pp. 217–222, 2005.

[19] L. Breiman, "Out-of-bag estimation," Statist. Dept., Univ. California, Berkeley, CA, USA, 1996.

[20] M. Owaida, A. Kulkarni, and G. Alonso, "Distributed inference over decision tree ensembles on clusters of FPGAs," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 12, no. 4, pp. 1–27, 2019.

[21] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-iot dataset," *Future Gener. Comput. Syst.*, vol. 100, pp. 779–796, Nov. 2019.

[22] J. M. Peterson, J. L. Leevy, and T. M. Khoshgoftaar, "A review and analysis of the Bot-IoT dataset," in *Proc. Int. Conf. Service Oriented Syst. Eng.*, 2021, pp. 20–27.

[23] A. Demirpolat, A. K. Sarica, and P. Angin, "ProtÉdge: A few-shot ensemble learning approach to software-defined networking-assisted edge security," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 6, 2021, Art. no e4138.

[24] J. Ali, R. Khan, N. Ahmad, and I. Maqsood, "Random forests and decision trees," *Int. J. Comput. Sci. Issues*, vol. 9, no. 5, p. 272, 2012.

[25] C. Baskin et al., "Uniq: Uniform noise injection for non-uniform Quantization of neural networks," *ACM Trans. Comput. Syst.*, vol. 37, nos. 1–4, pp. 1–15, 2021.

[26] R.-C. Chen, C. Dewi, S.-W. Huang, and R. E. Caraka, "Selecting critical features for data classification based on machine learning methods," *J. Big Data*, vol. 7, no. 1, p. 52, 2020.

[27] B. Azhagusundari and A. S. Thanamani, "Feature selection based on information gain," *Int. J. Innov. Technol. Explor. Eng.*, vol. 2, no. 2, pp. 18–21, 2013.

[28] J. Zhang and M. Zulkernine, "Anomaly based network intrusion detection with unsupervised outlier detection," in *Proc. Int. Conf. Commun.*, 2006, pp. 2388–2393.

[29] H. Han, X. Guo, and H. Yu, "Variable selection using mean decrease accuracy and mean decrease Gini based on random forest," in *Proc. 7th Int. Conf. Softw. Eng. Service Sci.*, 2016, pp. 219–224.

[30] B. Van Essen, C. Macaraeg, M. Gokhale, and R. Prenger, "Accelerating a random forest classifier: Multi-core, GP-GPU, or FPGA?" in *Proc. 20th Int. Symp. Field-Program. Custom Comput. Mach.*, 2012, pp. 232–239.

[31] T. Van Chu, R. Kitajima, K. Kawamura, J. Yu, and M. Motomura, "A high-performance and flexible FPGA inference accelerator for decision forests based on prior feature space partitioning," in *Proc. Int. Conf. Field-Program. Technol.*, 2021, pp. 1–10.

[32] T. Peng, C. Leckie, and K. Ramamohanarao, "Survey of network-based defense mechanisms countering the DoS and DDoS problems," *ACM Comput. Surv.*, vol. 39, no. 1, p. 3, 2007.

[33] I. Ullah and Q. H. Mahmoud, "Design and development of RNN anomaly detection model for IoT networks," *IEEE Access*, vol. 10, pp. 62722–62750, 2022.

[34] M. Ge, X. Fu, N. Syed, Z. Baig, G. Teo, and A. Robles-Kelly, "Deep learning-based intrusion detection for IoT networks," in *Proc. 24th Pac. Rim Int. Symp. Dependable Comput.*, 2019, pp. 256–25609.

[35] M. A. Ferrag and L. Maglaras, "DeepCoin: A novel deep learning and blockchain-based energy exchange framework for smart grids," *IEEE Trans. Eng. Manag.*, vol. 67, no. 4, pp. 1285–1297, Nov. 2020.

[36] D.-M. Ngo et al., "HH-NIDS: Heterogeneous hardware-based network intrusion detection framework for IoT security," *Future Internet*, vol. 15, no. 1, p. 9, 2023.

[37] G. Louppe, "Understanding random forests: From theory to practice," 2015, *arXiv:1407.7502*, 2014.

[38] C. Liu, R. Antypenko, I. Sushko, and O. Zakharchenko, "Intrusion detection system after data augmentation schemes based on the VAE and CVAE," *IEEE Trans. Rel.*, vol. 71, no. 2, pp. 1000–1010, Jun. 2022.

[39] N. Moustafa, B. Turnbull, and K.-K. R. Choo, "An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4815–4830, Jun. 2019.

[40] M. A. Lawal, R. A. Shaikh, and S. R. Hassan, "An anomaly mitigation framework for IoT using fog computing," *Electronics*, vol. 9, no. 10, p. 1565, 2020.

[41] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, "A comparative study of LPWAN technologies for large-scale IoT deployment," *ICT Exp.*, vol. 5, no. 1, pp. 1–7, 2019.

[42] A. Augustin, J. Yi, T. Clausen, and W. M. Townsley, "A study of LoRa: Long range & low power networks for the Internet of Things," *Sensors*, vol. 16, no. 9, p. 1466, 2016.

[43] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from Decentralized data," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[44] V. Mothukuri, P. Khare, R. M. Parizi, S. Pouriyeh, A. Dehghantanha, and G. Srivastava, "Federated-learning-based anomaly detection for IoT security attacks," *IEEE Internet Things J.*, vol. 9, no. 4, pp. 2545–2554, Feb. 2022.

Qingyu Zeng received the B.E. degree in VLSI design and system integration from Nanjing University, Nanjing, China, in 2021. He is currently pursuing the master's degree with the School of Engineering, Tokyo Institute of Technology, Tokyo, Japan.

He worked on binarized neural networks and optimization on FPGAs during his internship with Polytechnique Montreal, Montreal, QC, Canada, in 2023. His research interests include IoT-oriented security and efficient machine learning.

Yuko Hara-Azumi (Member, IEEE) received the Ph.D. degree in information science from Nagoya University, Nagoya, Japan, in 2010.

She was a JSPS Postdoctoral Research Fellow with Ritsumeikan University, Kyoto, from 2010 to 2012; during which she was also a Visiting Scholar with University of California, Irvine, CA, USA; and Karlsruhe Institute of Technology, Karlsruhe, Germany. In 2012, she joined Nara Institute of Science and Technology, Ikoma, Japan, as an Assistant Professor. Since 2014, she has been with the Department of Information and Communications Engineering, School of Engineering, Tokyo Institute of Technology, Tokyo, Japan, where she is currently an Associate Professor. Her research interests include system-level design automation, especially on high-level and logic synthesis, microprocessor architecture, and hardware/software co-design for embedded/IoT systems.

Dr. Hara-Azumi has served as an Organizing and Program Committee Member for several premier conferences, including DAC, ICCAD, DATE, CASES, ASP-DAC, and FPL. She is a member of ACM.