OPEN ACCESS
Indo-JC
Indonesian Journal on Computing
ISSN 2460-9056
socj.telkomuniversity.ac.id/indojc

Ind. Journal on Computing
Vol. 8, Issue. 3, December 2023. pp. 29-42
doi:10.34818/indojc.2023.8.3.878

QUIDS: A Novel Edge-Based Botnet Detection with Quantization for IoT Device Pairing

Aji Gautama Putrada [#1], Nur Alamsyah [*2], Mohamad Nurkamal Fauzan [*3], Sidik Prabowo [*4], Ikke Dian Oktaviani [*5]

[#] *Advanced and Creative Networks Research Center, Telkom University Bandung, Indonesia*

[*] *School of Computing, Telkom University Bandung, Indonesia*

[1] ajigps@telkomuniversity.ac.id

[2] nuralamsyah@student.telkomuniversity.ac.id

[3] mnurkamalfauzan@student.telkomuniversity.ac.id

[4] prabowo@student.telkomuniversity.ac.id

[5] oktavianiid@telkomuniversity.ac.id

Abstract

Advanced machine learning has managed to detect IoT botnets. However, conflicts arise due to complex models and limited device resources. Our research aim is on a quantized intrusion detection system (QUIDS), an edge-based botnet detection for IoT device pairing. Using k-nearest neighbor (KNN) within QUIDS, we incorporate quantization, random sampling (RS), and feature selection (FS). Initially, we simulated a botnet attack, devised countermeasures via a sequence diagram, and then utilized a Kaggle botnet attack dataset. Our novel approach includes RS, FS, and 16-bit quantization, optimizing each step empirically. The test results show that employing a mean decrease in impurity (MDI) by FS reduces features from 115 to 30. Despite a slight accuracy drop in KNN due to RS, FS, and quantization sustain performance. Testing our model revealed 1200 RS samples as optimal, maintaining performance while reducing features. Quantization to 16-bit doesn't alter feature value distribution. Implementing QUIDS increased the compression ratio (CR) to 175×, surpassing RS+FS threefold and RS by 13 times. This novel method emerges as the most efficient in CR.

Keywords: Intrusion detection system, edge computing, botnet attack, quantization, IoT device pairing.

Abstrak

Pembelajaran mesin tingkat lanjut telah berhasil mendeteksi botnet IoT. Namun konflik muncul karena model yang kompleks dan sumber daya perangkat yang terbatas. Tujuan penelitian kami adalah pada *Quantized Intrusion Detection System* (QUIDS), deteksi botnet berbasis tepi untuk pemasangan perangkat IoT. Kami menggabungkan *quantization, random sampling* (RS), dan *feature selection* (FS). Awalnya kami merancang tindakan pencegahan melalui sequence diagram, dan kemudian menggunakan dataset serangan botnet Kaggle. Pendekatan baru kami mencakup RS, FS, dan 16-bit *quantization*. Hasil pengujian menunjukkan bahwa *mean decrease in impurity* (MDI) oleh FS mengurangi fitur dari 115 menjadi 30. Meskipun ada penurunan akurasi di KNN karena RS, FS, dan kuantisasi mempertahankan kinerja. Menguji model kami menunjukkan 1200 sampel RS sebagai sampel optimal. Kuantisasi ke 16-bit tidak mengubah distribusi nilai fitur. Penerapan QUIDS meningkatkan *compression ratio* (CR) menjadi 175×, melampaui RS+FS tiga kali lipat dan RS sebanyak 13 kali lipat. Metode baru ini muncul sebagai yang paling efisien dalam CR.

Kata Kunci: Intrusion detection system, edge computing, botnet attack, quantization, IoT device pairing.

Received on xxx, accepted on xxx, published on xxx

Received on September, 2023. Accepted on December, 2023

I. INTRODUCTION

TRADITIONAL Intrusion Detection Systems (IDS) are unable to withstand advanced cyber attacks such as malicious botnet attacks as more and more devices are connected to the Internet via the Internet of Things (IoT) [1] architecture. One of the vulnerabilities of IoT devices due to botnet attacks is during pairing, namely when the user connects to his IoT device via the application [2]. Advanced techniques such as machine learning can detect botnets in IoT [3]. However, the problem is that highly complex machine learning is contradictory to IoT devices, which usually have limited resources.

Some of our previous research has applied compression models concerned with using machine learning on devices with limited resources. Shuffle-split nearest neighbor editing (SSENN) makes the k-nearest neighbor (KNN) model small by carrying out random sampling (RS) and discarding the training data without reducing its performance [4]. In our research using an architecture called EdgeSL, we propose two new methods: quantized 8-bit k nearest neighbors (Q8KNN) and DistilKNN [5]. Q8KNN reduces the size of the KNN model with quantization, while DistilKNN utilizes distillation, a simple student model that uses soft labels generated by the teacher model. Utilizing RU and quantization to apply IDS to IoT edge devices to detect botnet attacks is a research opportunity.

Low-rank factorization, such as quantization and distillation, is also a method of model compression [6]. The feature selection (FS) method works by reducing the dimensions of the dataset while maintaining its global correlation and local geometry [7]. Zhu *et al.* [8] combines low-rank factorization and quantization with a method called low-rank representation vector quantization (LR2VQ) in model compression. Their implementation in ResNet-18 and ResNet-50 models results in a compression ratio (CR) of 43x and 31x, respectively. Combining FS with quantization and RU is a research opportunity.

Our research aim is to apply quantized IDS (QUIDS), a novel edge-based botnet detection for IoT device pairing. IDS in QUIDS uses KNN through quantization, RU, and FS. First, we design a botnet attack on device pairing and how to counter it with a sequence diagram. Then, we took the botnet attack detection dataset from Kaggle. Our novel method has three stages: RS, FS, and 16-bit quantization. We carry out an empirical optimization process at each point. This part then uses the mean decrease in impurity (MDI) at the FS stage. Finally, we tested the performance of our novel model with several parameters: accuracy, sensitivity, specificity, g-mean, model size, and CR.

To the best of our knowledge, no research has applied edge computing in IDS. Following are our research contributions:

- A botnet attack detection on IoT that applies edge computing.
- A sequence diagram explaining the threat of botnet attacks on IoT device pairing.
- A sequence diagram that explains how IDS can withstand botnet attacks on IoT device pairing.
- QUIDS, a novel IDS that applies quantization, RS, and FS with optimal CR.

The remainder of this paper has the following structure: Section II discusses research related to our study. Section III outlines our research methodology. Section IV shows test results and discusses them against state-of-the-art research. Finally, Section V highlights our judgment and answers the research aim.

II. LITERATURE REVIEW

Recent papers have discussed how to detect botnet attacks on IoT with machine learning. Catillo *et al.* [9] used autoencoder as anomaly detection in detecting botnet attacks on IoT. They tried it on nine IoT devices with sensitivity results of 0.99 to 1.00. Soe *et al.* [10] resisted Gafgyt and Mirai type botnet attacks by comparing artificial neural networks (ANN), decision trees, and naïve Bayes. ANN with sigmoid activation performs best with an accuracy value of 0.99. Creating edge-based machine learning detecting botnet attacks on IoT is a research opportunity.

Several studies have discussed security in IoT device pairing. Bruesch *et al.* [11] researched the threat of attacks on device pairing using the gait method. The results of this research state that observation via video can produce key sequences, which are a threat in gait-based device pairing. Farrukh *et al.* [12] stated that the weak secure key generation in homogeneous context-based device pairing is a single-point

Table I: Related Works on Botnet Attack, IoT, Device Pairing, and Edge Computing

Cite	Botnet Attack on IoT	Device Pairing	Sequence Diagram	Model Compression and Edge Computing
Catillo *et al.* [9]	✓	✗	✗	✗
Soe *et al.* [10]	✓	✗	✗	✗
Bruesch *et al.* [11]	✗	✓	✗	✗
Farrukh *et al.* [12]	✗	✓	✗	✗
Putrada *et al.* [4]	✗	✗	✗	✓
Putrada *et al.* [5]	✗	✗	✗	✓
Proposed Method	✓	✓	✓	✓

failure. Their solution is three stages: event detection using the window method, grouping events with fuzzy clustering, and group key distribution. There is a two-fold research opportunity. The first is to design a sequence diagram for botnet attacks on device pairing. The second is to create a sequence diagram of how the IDS can withstand these attacks.

Regarding applying compression models to KNN, our previous research has offered several novelties. Our first research carried out a compression model called SSENN on KNN by combining the concepts of RU and ENN [4]. By running the algorithm on Arduino, we show that SSENN can create a KNN model smaller than ENN but accurately approximates the original KNN. Our second contribution is Q8KNN, an 8-bit quantization method on KNN [5]. With this method, we can enter five times more data into the NodeMCU because it reduces the data size based on its type, from 64 bits to 8 bits. Next, we also created a novel method called DistilKNN, which uses DNN as a teacher model and KNN as a student model. This method has been proven more efficient than SSENN and Q8KNN at a model size of 45 kB. The research opportunity is to create a model with more optimal CR utilizing quantization, RU, and FS. Table I summarizes our explanation in a table and highlights our research contributions.

III. RESEARCH METHOD

We use a research methodology to achieve our research aim. First, we design a botnet attack on device pairing and how to counter it with a sequence diagram. Then, we took the botnet attack detection dataset from Kaggle. Our novel method has three stages: RS, FS, and 16-bit quantization. We carry out an empirical optimization process at each point. This part then uses MDI at the FS stage. Finally, we tested the performance of our novel model with several parameters: accuracy, sensitivity, specificity, g-mean, model size, and CR. Figure 1 summarizes our explanation.

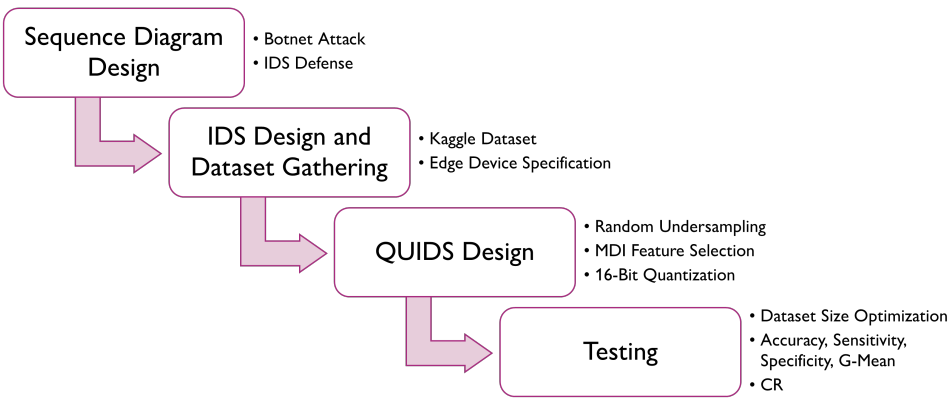


Figure 1: Our Proposed Methodology

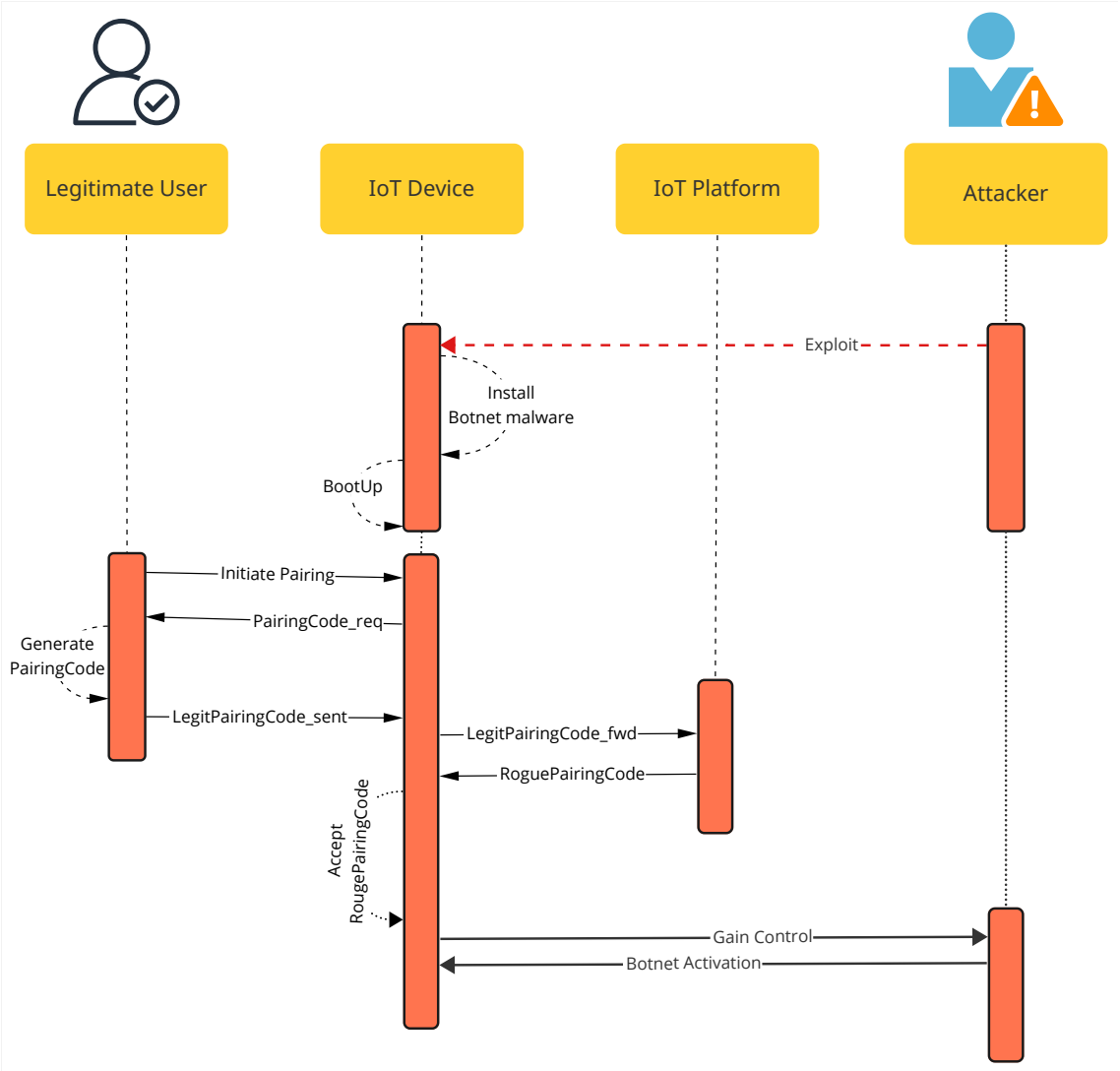


Figure 2: The botnet attack on IoT device pairing sequence diagram.

A. *Botnet Attack on IoT Device Pairing*

Someone who has just purchased a commercial IoT product must connect their smartphone to the product via an application, where device pairing comes into play [13]. IoT device pairing is when two devices try to connect to the same IoT network [14]. By joining the two devices, control, monitoring, or other communication processes can occur between the two devices [15]. On the other hand, a botnet attack is when an adversary controls a device in a network and carries out malicious activity [16]. The effect of botnet attacks in IoT device pairing is that when a legitimate user tries to connect to an IoT device that has been affected, the legitimate user also becomes affected.

Botnet attacks in IoT device pairing begin with initiation from a legitimate user. The IoT device will ask the legitimate user for a pairing code, and they will answer it with a valid and secure one. However, instead of using a legit pairing code, the attacker changes the pairing code through illicit communications. So, the answer from the IoT platform is not the legitimate user but the attacker. In other words, the legitimate user presents the rouge pairing code to the attacker. The legitimate user is now part of the attacker's network. Figure 2 explains this process as a sequence diagram.

B. Edge-Based IDS

Edge computing in the context of IoT means bringing processing closer to the end-device [17]. This is important because processing, which uses machine learning, becomes a double-edged sword in IoT. On the one hand, it provides more effective processing in an IoT environment, and on the contrary, it has heavy processing, whose characteristics are the opposite of IoT's nature. real-time [18]. Processing becomes more responsive as it gets closer to the end device via edge computing due to reduced latency due to network communications. On the other hand, IDS is a technology that monitors the network and provides alerts if there is suspicious network activity, usually with anomaly detection or machine learning [19]. Applying edge computing to an IDS means that the IDS, which was previously placed in the cloud, is moved to the end device to speed up the detection of attacks, especially botnet attacks.

The IDS placed on the end device monitors the network continuously. The IDS's task in preventing the attacks described in the previous sequence diagram is to detect suspicious communications between the IoT device and the IoT platform when the IoT device requests a pairing code. If there is suspicious communication, IDS will report it to the IoT device, which then thwarts the sending of the rogue code. That way, legitimate users will not become part of the attacker's botnet. IDS is a crucial layer in IoT in protecting the IoT system from cyber crimes. Figure 3 shows these steps as a sequence diagram.

C. QUIDS

Our case study uses an IoT doorbell as an IoT device. The main processing in IoT doorbells is usually represented by a single-board computer (SBC) [20]. The SBC generally has an ARM cortex 1.0 to 1.5 GHz processor. Then, the RAM has a size of 512 MB to 1 GB. The flash memory has a capacity of 4 GB to 8 GB. The operating system is Linux, which is specifically for embedded systems. The SBC input-output (I/O) is a USB port, GPIO for sensors, and other I/O such as HDMI. Even though it has a GPIO, it usually cannot be installed with complex outputs due to the absence of an analog-to-digital converter (ADC). These specifications contrast with computer servers that run IDS in the cloud, usually with processor speeds of up to more than 2.5 GHz, RAM up to 16 GB, solid state disk (SSD) with a capacity of 256 GB, OS for servers such as Linux Ubuntu Server, and I/ O which is more advanced and resourceful. With a rough ratio of 32 times the capacity of secondary memory, there must be a method that can model compression with the same ratio. Don't forget, apart from running an IDS, an IoT doorbell has other tasks that are not easy, such as user interface, running the camera, wireless communication, data storage, motion detection and sensor reading, and other charges. Table II compares specifications between edge computers and cloud computers for IDS.

Botnet attacks have several types. Gafgyt is a botnet attack that targets IoT devices such as routers, cameras, and smart devices. After mastering the IoT device, Gafgyt will exploit [21]. Mirai also attacks IoT devices. In contrast to Gafgyt, Mirai carries out distributed denial of service (DDoS) [22]. Reaper is a botnet attack that also attacks IoT. Reaper can carry out DDoS and confidentiality threats [23]. Unlike previous episodes, Hajime is a botnet attack that targets IoT devices to improve their security system [24]. Then, like Mirai, Echobot is also a botnet attack that carries out DDoS by targeting IoT devices [25].

We use the botnet attack dataset from Kaggle. The original dataset comes from research by Meidan *et al.* [26], which contains a Gafgyt-type botnet attack on IoT doorbells. where this dataset has 15,648

Table II: Specification Comparison of Edge Computer and Cloud Computer for IDS

Specification	End Device	Cloud Server
CPU Speed	1.0 to 1.5 GHz	2.5 GHz
RAM Capacity	512 MB to 1 GB	16 GB
Secondary Memory Type	Flash Memory	SSD
Secondary Memory Capacity	4 to 8 GB	256 GB
OS	Linux For Embedded System	Linux Ubuntu Server
I/O	Simple	Complex
Tasks	User interface, camera function, wireless communication, data storage, motion detection, and sensor reading	Communication, data analytics, user authentication, security, and logging

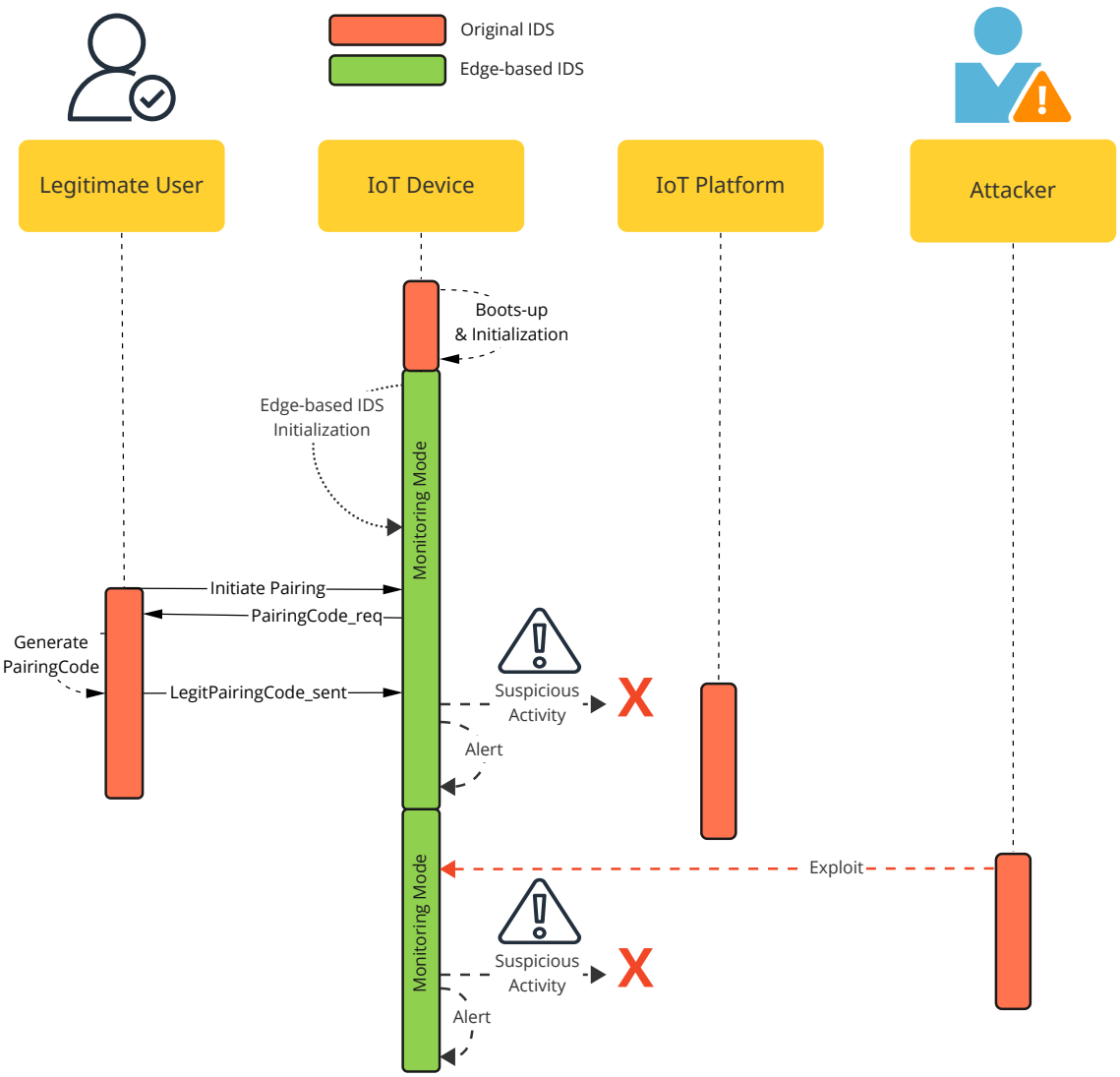


Figure 3: The sequence diagram of how the IDS protects the device pairing from botnet attack.

data items, 115 features, and two labels. Label 0 indicates that the data item is not a botnet attack; conversely, label 1 indicates that the data item is a botnet attack. The number of features listed results from multiplication between the main features, dimension reduction for the time-series data, and feature extraction. Its main features consist of the following items:

- Host traffic, based on internet protocol (IP) address
- Host to destination traffic, based on IP address
- Host process to destination process traffic, based on IP address, with port address
- Host to destination traffic jitters, based on IP address

The data then goes through dimension reduction [9]. Each dimension reduction process produces different dimensions. From each new feature, feature extraction is carried out based on a window value with the following statistical functions:

- The weight, which is the amount of observed time series data
- The average of data in one window
- The standard deviation of data in one window
- The radius, which is the root of the sum of the squared value of two stream variances

- The magnitude, which is the root of the sum of the squared value of two streams' means
- The covariance of two streams
- The Pearson correlation coefficient (PCC) of two streams

We use KNN for botnet attack detection, where the method performs classification based on the largest class in k data items that has the closest distance to the data to be classified in the feature space [27]. Several methods can calculate distances in KNN, using Euclidean distance [28]. KNN is a non-parametric method, meaning that the dataset's distribution is not considered in the algorithm [29]. Here, we use $k = 3$, which means that the largest label in the three nearest neighbors is the one that determines the label of the predicted data [30]. KNN is suitable for data spread out and separated not linearly and whose distribution is not well-defined.

Our novel method, QUIDS, has the function of performing model compression on KNN. The main goal of model compression is to make machine learning smaller without reducing its performance [17]. This method goes through three stages. The first stage is RS, where the sampling fraction is an important factor. This value is determined without reducing model performance from the original dataset. The next stage is FS, where several studies have shown that FS can compress a [31] model. We use MDI type FS, the method the random forest model originally used to rank the features [32]. This ranking becomes a filter for selecting features for training the detection model, usually utilizing a threshold whose value is the average of all scales. The third stage is performing quantization, which generally consists of two steps. The first step is to normalize so that no values are lost due to being outside the range of the new data type. The second step is converting the data type to a more compact representation, changing the data type from 64-bit floating point to 16-bit integer [33].

Algorithm 1 shows the algorithm from QUIDS, where the variable X is the original dataset. Then, in the RS step, P is the probability of selection, n is the number of samples, and N is the population. The output of this step is X', which is the RS result. In the next step, namely FS, the variable s is the MDI score of each feature, S is the data structure containing all s, B is the number of trees in the random forest model, T_b is the number of nodes in each tree, $p(t)$ is the proportion of each feature in the node, $Gini(t)$ and is the impurity of each node. The output of the FS step is X'', which is the result of the FS transformation of X'. In the first quantization step, X''_{min} is the smallest value in X'', and X''_{max} is its largest value. The output of this step is X', which is the result of the quantization transformation of X''. The algorithm output is a model which is an IDS model resulting from KNN training on X'''.

We test our performance with several metrics. First, we use accuracy because this metric compares all the correctly predicted data with the amount of data. We use sensitivity and specificity for two reasons: Our detection is a binary classification, 0 is no botnet attack, and 1 is a botnet attack detected. The second reason is that sensitivity and specificity are two metrics that can be used if there is an imbalance in the [34] dataset. G-mean is a metric that aggregates sensitivity and specificity. CR shows the ratio between the size of the model that has gone through compression and the size of the original model [35]. Here is the formula for CR:

$$\text{CR} = \frac{\text{Original Size}}{\text{Compressed Size}} \tag{4}$$

The greater the CR value, the more effective the performance of the compression algorithm model.

We search for the optimal value of n in the RS step by applying repeated k-fold cross-validation [**?**]. Several studies have shown that repeating k-fold cross-validation testing can reduce dependency on random splits between training and testing data [36].

One ramification of quantization is information loss [37]. This can happen because the normalization process in quantization can eliminate useful details in the data. Analysis of kernel density estimation (KDE) from each dataset can detect whether information loss has occurred or not. Information loss can be observed from the difference in function form between the original and quantized datasets.

Algorithm 1: The QUIDS algorithm

Data: n, N, X, B
Result: model
Get the optimum n for the RS stage;
/* RS */
Calculate the P with the following equation:

$$P = \frac{n}{N} \tag{1}$$

$X' \leftarrow$ Select each data item in the dataset with a probability of P;
/* FS */
Calculate the MDI score of each feature s with the following equation:

$$s = \frac{1}{B} \sum_{b=1}^{B} \sum_{t=1}^{T_b} p(t) \cdot Gini(t), s \in S \tag{2}$$

$X'' \leftarrow \text{FS}(X', S)$;
/* Quantization */
Normalize each member of the transformed dataset with the following equation:

$$X''' = \frac{X'' - X''_{min}}{X''_{max} - X''_{min}} \times 2^{16} \tag{3}$$

$X''' \leftarrow$ cast X''' datatype from 64-bit floating point to 16-bit unsigned integer;
/* IDS Model Training */
$model \leftarrow$ Train KNN model with X''';

IV. RESULTS AND DISCUSSION

A. Results

The first step in testing is to train the IDS model for botnet attacks using the KNN model. At this stage, we are still using 115 features and two labels. The number of original datasets is 15,647 data items. The number of labels with a value of "0" is 14,865, while the number with a "1" is 782. Under these circumstances, the imbalance ratio (IR) value is 19.01. The IR value is included in the significant imbalance category. Metrics such as precision and f1-score do not describe the model performance. Next, we divided the dataset with a train-by-test composition of 66.7%-by-33.3%. We use KNN with $k = 5$. The model's accuracy is 0.9998, while the model size is 9.7 MB.

The first step of our QUIDS is to perform RS and determine the optimum n. At this stage, we are still using 115 features and two labels. We look for the optimal n range $200 \leq n \leq 1200$. We use k-fold cross-validation with $k = 3$ and metric accuracy to test the model performance with each RS. We iterate over each n value ten times. Figure 4 compares each model's version using a line plot of the average of each n and an error bar showing the standard deviation. Our optimization problem is as follows: The larger the value of n, the larger the model size.

Conversely, the lower the value of n, the greater the standard deviation of accuracy and the lower the mean accuracy. We choose $n = 1200$ as the optimum n value for RS. At this stage, the accuracy of KNN+RS is 0.9899, while the model size is reduced to 746.8 kB.

In the second stage of QUIDS, we perform FS with MDI. The score results for each feature by MDI range from 0.0 to 0.09. The FS process with MDI retains features with an MDI score above the average and excludes features with an MDI score below the average MDI score. The average MDI score is 0.0087. The FS process reduces the number of features from 115 to 30. Figure 5 shows a bar plot of the scores of some of the best features. In this figure, the threshold is shown in the form of a dotted line. Features with values above the threshold are retained. At this stage, the accuracy of KNN+RS+FS is 0.9899, with

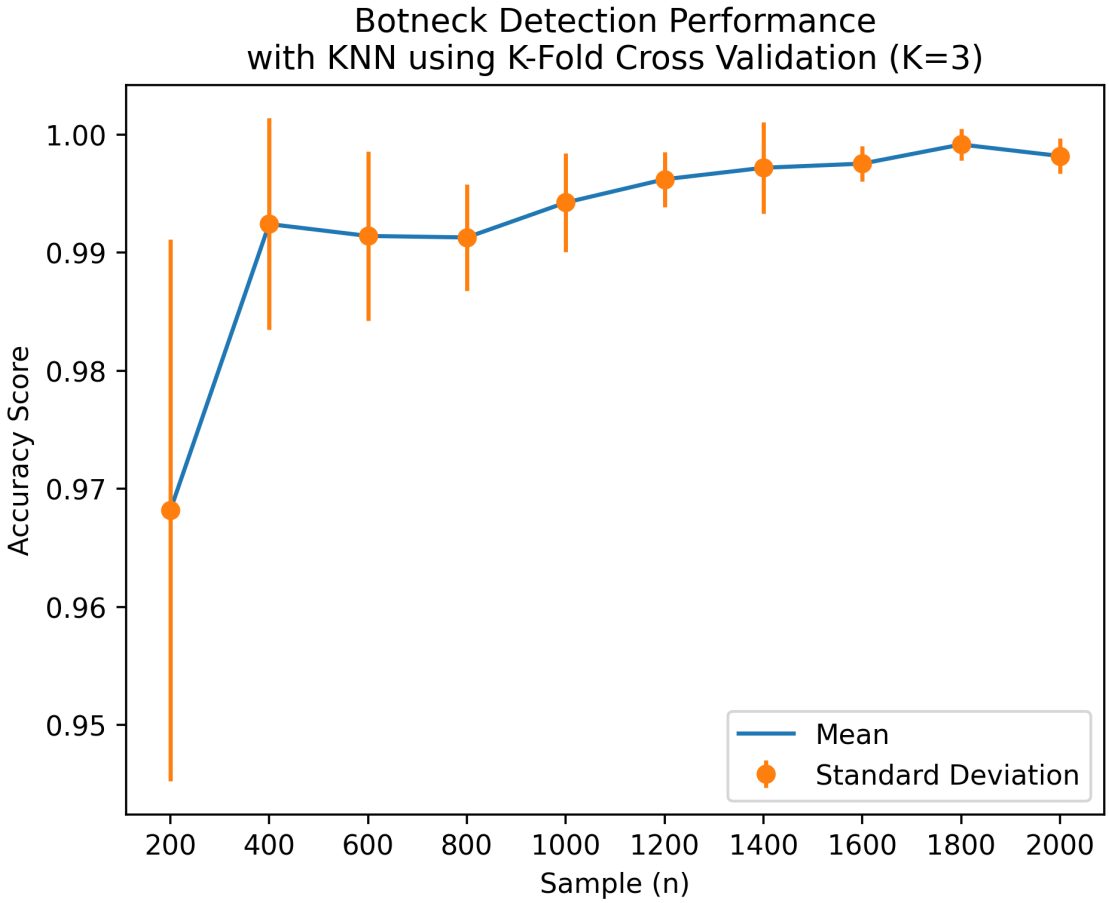


Figure 4: Performance comparison of IDS detection with different sample (n) values using k-fold cross-validation with $k = 3$ and ten times repetition, shown with a line plot for the mean of the accuracy and an error bar showing the standard deviation.

the model size reduced to 194 kB.

The final step of QUIDS is to apply quantization. Quantization also requires optimization: The smaller the data type used, the greater the CR value, but conversely, the information loss also becomes greater. We do optimization with KDE. Figure 6 shows the KDE plot of the original dataset and the dataset after quantization into three different data types: uint32, uint16, and uint8. Changes only occur in the data scale at uint32 and uint 16, but there is no change in the KDE form. However, in uint8, the shape of KDE has changed. This indicates that there is information loss due to loss of detail. The uint16 datatype is more optimal than uint32 because its bit size is smaller.

In the final stage of QUIDS, we need to check whether there is a decrease in accuracy and how our compression method performs. After going through the last step, our QUIDS method has an accuracy of 0.9899, while the model size is 58 kB. Figure 7 shows a complete performance comparison comparing botnet attack detection performance with KNN, KNN+RS, KNN+RS+FS, and QUIDS (KNN+RS+FS+quantization). There was a decrease when applying the compression model to KNN. However, using FS and quantization does not reduce model performance. Sensitivity is the metric with the lowest value among the other metrics. Sensitivity measures the model's ability to predict a value of 1. This low value can occur due to imbalance problems in the dataset. In real terms, there was a prediction error once in 23 datasets with label 1. This number could increase if there were more data with label 1.

We now analyze how our novel compression method performs. Figure 8 shows the comparison bar chart. Figure 8a shows the model's size after each stage, from RS to QUIDS. Meanwhile, for analysis,

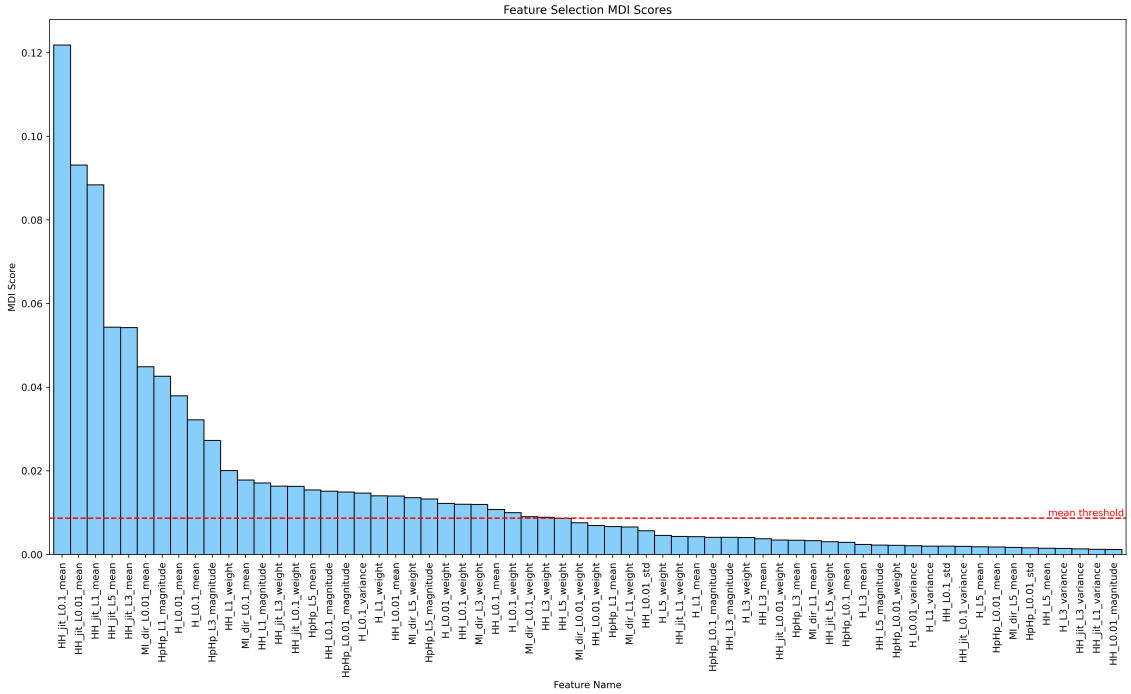


Figure 5: The MDI score result of the FS process.

we have prepared Figure 8b, which shows the CR at each point. In the RS stage, the CR is $13\times$, whereas in the next stage, RS+FS, the CR increases four times to $51\times$. Finally, at implementing the novel QUIDS model, the CR became $175\times$, up to three times the RS+FS and 13 times the RS. QUIDS is the compression method with the best CR.

B. Discussion

Several previous studies have produced papers on detecting botnet attacks on IoT using machine learning, such as the paper [9], [10]. However, the critical research opportunity for IDS in IoT is to reduce processing time due to the real-time nature of IoT. One method for optimizing processing time is to apply edge computing, where, in the case of IDS, security computing is run on the IoT end device, cutting the time required for communication with the cloud. Contribution Our research is a botnet attack detection on IoT that applies edge computing.

IoT device pairing is a crucial part of IoT because it is related to user experience. It cannot be separated

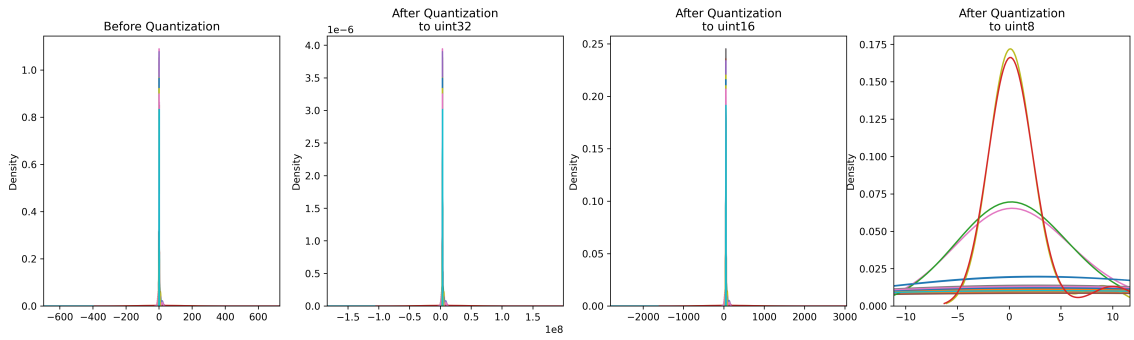


Figure 6: KDE Comparison of four different quantizations on the botnet attack dataset

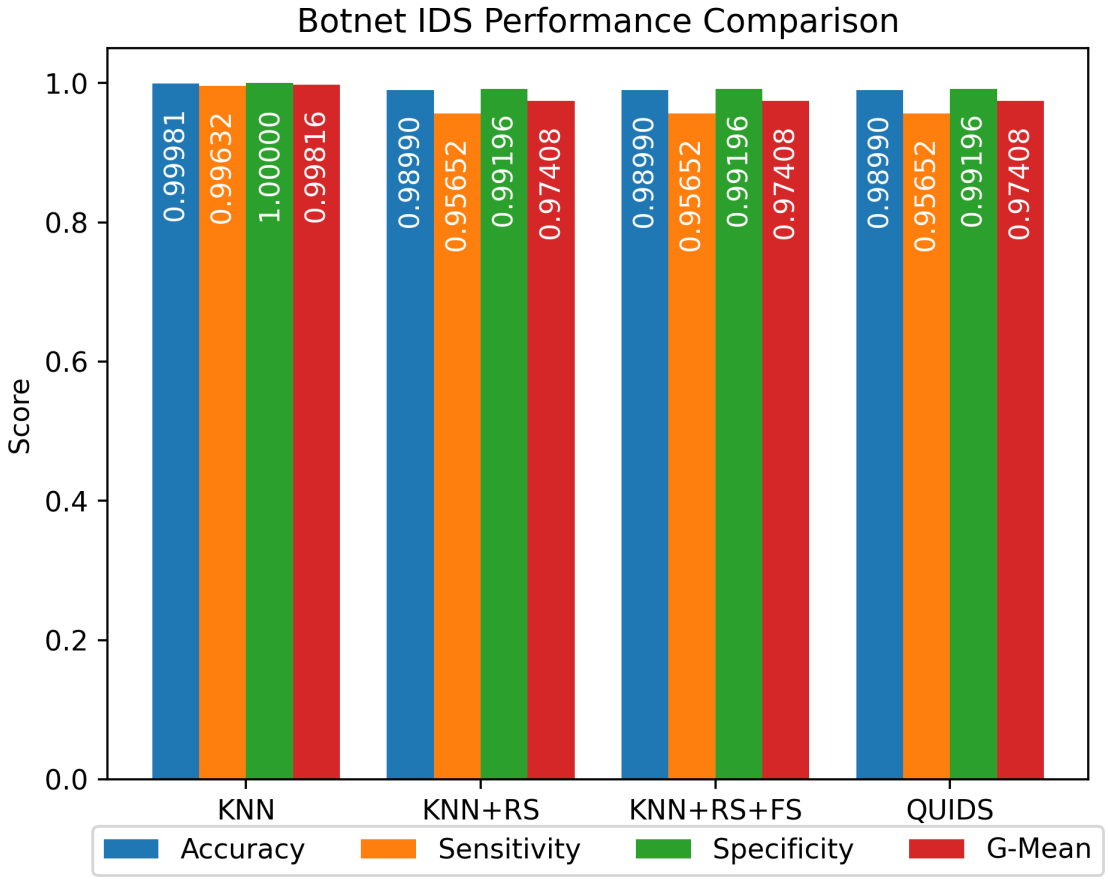
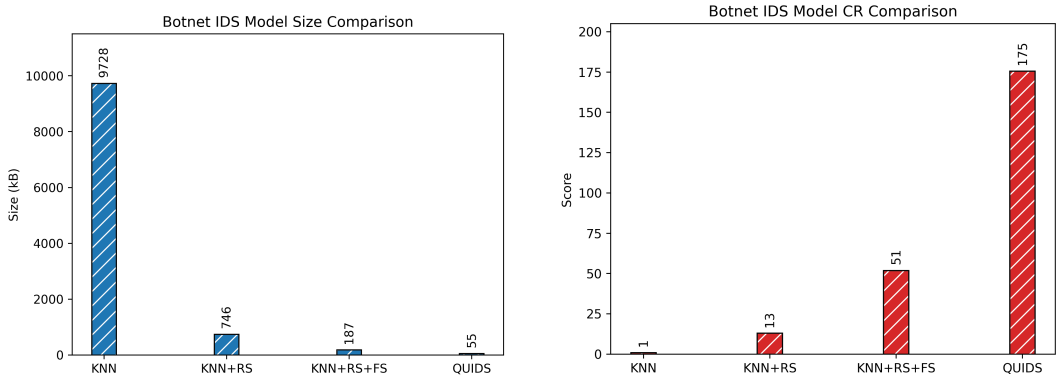


Figure 7: The botnet attack detection performance comparison with different model compression techniques.

from security threats when users pair devices. Several studies have applied security to the device pairing process, such as the paper [11], [12]. Two sequences can be created for each method to clarify the process



(a) (b)

Figure 8: The botnet attack size and complexity characteristics comparison with different model compression techniques: (a) Size (b) CR.

of security attacks on device pairing and how to implement security. The contribution of our research is two-fold. First, a sequence diagram that explains the threat of botnet attacks on IoT device pairing. Second, a sequence diagram that illustrates security to ward off external attacks on IoT device pairing.

Our previous research has offered several novelties in model compression in KNN, for example, with RS [4] and with quantization [5]. It is a research opportunity to combine these two concepts and then add feature selection. Apart from that, applying this method to IDS is also an opportunity for contribution. Our research contribution is QUIDS, a novel IDS that applies quantization, RS, and FS, which has optimal CR.

V. Conclusion

In this research, we implemented an IDS that uses the edge computing concept, where the IDS can be embedded in an IoT end device. In addition, this IDS is to withstand attacks on the IoT device pairing process, where this IDS also applies a compression model to enable edge computing. The processes involved are RS, FS, and quantization. The test results show that the optimum number of RS is 1200 samples. Then, the FS process reduces the number of features from 115 to 30. Quantization from the 64-bit floating point data type to the 16-bit unsigned integer does not change the distribution of feature values. The RS process reduces the accuracy of the KNN model from 0.99981 to 0.98990. However, the FS and quantization processes do not reduce performance any lower. Finally, at implementing the novel QUIDS model, the CR became $175\times$, up to three times the RS+FS and 13 times the RS. QUIDS is the compression method with the best CR.

Acknowledgment

We deeply appreciate TU Delft for generously offering us a conducive working environment that facilitated the progress and success of our research. Their support and resources were instrumental in our ventures, allowing us to dredge into this study and contribute meaningfully to the field.

Reference

[1] Manoj S Koli and Manik K Chavan. An advanced method for detection of botnet traffic using intrusion detection system. In *2017 International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pages 481–485. IEEE, 2017.

[2] Bandari Pranay Kumar, Gautham Rampalli, Pille Kamakshi, and T Senthil Murugan. Ddos botnet attack detection in iot devices. In *Smart Trends in Computing and Communications: Proceedings of SmartCom 2022*, pages 21–27. Springer, 2022.

[3] Alaa Dhahi Khaleefah and Haider M Al-Mashhadi. Detection of iot botnet cyber attacks using machine learning. *Informatica*, 47(6), 2023.

[4] Aji Gautama Putrada, Maman Abdurohman, Doan Perdana, and Hilal Hudan Nuha. Shuffle split-edited nearest neighbor: A novel intelligent control model compression for smart lighting in edge computing environment. In *Information Systems for Intelligent Systems: Proceedings of ISBM 2022*, pages 219–227. Springer, 2023.

[5] Aji Gautama Putrada, Maman Abdurohman, Doan Perdana, and Hilal Hudan Nuha. Edgesl: Edge-computing architecture on smart lighting control with distilled knn for optimum processing time. *IEEE Access*, 2023.

[6] Yun Cai, Hong Gu, and Toby Kenney. Rank selection for non-negative matrix factorization, 2022.

[7] Luxi Jiang and Xiuhong Chen. Spectral feature selection via low rank decomposition and local preservation. In *2023 3rd International Conference on Neural Networks, Information and Communication Engineering (NNICE)*, pages 518–522. IEEE, 2023.

[8] Zezhou Zhu, Yuan Dong, and Zhong Zhao. Learning low-rank representations for model compression. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2023.

[9] Marta Catillo, Antonio Pecchia, and Umberto Villano. A deep learning method for lightweight and cross-device iot botnet detection. *Applied Sciences*, 13(2):837, 2023.

[10] Yan Naung Soe, Yaokai Feng, Paulus Insap Santosa, Rudy Hartanto, and Kouichi Sakurai. Machine learning-based iot-botnet attack detection with sequential architecture. *Sensors*, 20(16):4372, 2020.

[11] Arne Bruesch, Ngu Nguyen, Dominik Schürmann, Stephan Sigg, and Lars Wolf. Security properties of gait for mobile device pairing. *IEEE Transactions on Mobile Computing*, 19(3):697–710, 2019.

[12] Habiba Farrukh, Muslum Ozgur Ozmen, Faik Kerem Ors, and Z Berkay Celik. One key to rule them all: Secure group pairing for heterogeneous iot devices. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 3026–3042. IEEE, 2023.
[13] Pradeeka Seneviratne and Pradeeka Seneviratne. Connecting with iot servers using a restful api. *Beginning LoRa Radio Networks with Arduino: Build Long Range, Low Power Wireless IoT Networks*, pages 171–194, 2019.
[14] Aji Gautama Putrada and Nur Ghaniaviyanto Ramadhan. A proposed hidden markov model method for dynamic device pairing on internet of things end devices. *Journal of ICT Research & Applications*, 14(3), 2021.
[15] Heka Bagaskara, Aji Gautama Putrada, and Endro Ariyanto. Proximity and dynamic device pairing based authentication for iot end devices with decision tree method. In *2020 6th International Conference on Interactive Digital Media (ICIDM)*, pages 1–5. IEEE, 2020.
[16] Amritanshu Pandey, Sumaiya Thaseen, Ch Aswani Kumar, and Gang Li. Identification of botnet attacks using hybrid machine learning models. In *Hybrid Intelligent Systems: 19th International Conference on Hybrid Intelligent Systems (HIS 2019) held in Bhopal, India, December 10-12, 2019 19*, pages 249–257. Springer, 2021.
[17] Aji Gautama Putrada, Nur Alamsyah, Syafrial Fachri Pane, Mohamad Nurkamal Fauzan, and Doan Perdana. Knowledge distillation for a lightweight deep learning-based indoor positioning system on edge environments. In *2023 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, pages 370–375. IEEE, 2023.
[18] Prashant Kumar, Gaurav Purohit, Pramod Tanwar, Chitra Gautam, and Kota Solomon Raju. Real time, an iot-based affordable air pollution monitoring for smart home. In *First International Conference on Sustainable Technologies for Computational Intelligence: Proceedings of ICTSCI 2019*, pages 837–844. Springer, 2019.
[19] Aji Gautama Putrada, Nur Alamsyah, Syafrial Fachri Pane, and Mohamad Nurkamal Fauzan. Xgboost for ids on wsn cyber attacks with imbalanced data. In *2022 International Symposium on Electronics and Smart Devices (ISESD)*, pages 1–7. IEEE, 2022.
[20] BL Kiran, J Chandan, BS Jeevan, C Mohananka, and Vallabh Mahale. A survey on door lock security system using iot. *Perspectives in Communication, Embedded-systems and Signal-processing-PiCES*, 5(2):40–43, 2021.
[21] Irfan, IM Wildani, and IN Yulita. Classifying botnet attack on internet of things device using random forest. In *IOP Conference Series: Earth and Environmental Science*, volume 248, page 012002. IOP Publishing, 2019.
[22] Rufaida Bibi Auliar and Girish Bekaroo. Security in iot-based smart homes: A taxonomy study of detection methods of mirai malware and countermeasures. In *2021 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, pages 1–6. IEEE, 2021.
[23] Pooja Kumari and Ankit Kumar Jain. A comprehensive study of ddos attacks over iot network and their countermeasures. *Computers & Security*, page 103096, 2023.
[24] Benjamin Vignau, Raphaël Khoury, Sylvain Hallé, and Abdelwahab Hamou-Lhadj. The evolution of iot malwares, from 2008 to 2019: Survey, taxonomy, process simulator and perspectives. *Journal of Systems Architecture*, 116:102143, 2021.
[25] Raphaël Khoury, Benjamin Vignau, Sylvain Hallé, Abdelwahab Hamou-Lhadj, and Asma Razgallah. An analysis of the use of cves by iot malware. In *Foundations and Practice of Security: 13th International Symposium, FPS 2020, Montreal, QC, Canada, December 1–3, 2020, Revised Selected Papers 13*, pages 47–62. Springer, 2021.
[26] Yair Meidan, Michael Bohadana, Yael Mathov, Yisroel Mirsky, Asaf Shabtai, Dominik Breitenbacher, and Yuval Elovici. N-baiot—network-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Computing*, 17(3):12–22, 2018.
[27] Rofif Irsyad Fakhruddin, Maman Abdurohman, and Aji Gautama Putrada. Improving pir sensor network-based activity recognition with pca and knn. In *2021 International Conference on Intelligent Cybernetics Technology & Applications (ICICyTA)*, pages 138–143. IEEE, 2021.
[28] Parlin Nando, Aji Gautama Putrada, and Maman Abdurohman. Increasing the precision of noise source detection system using knn method. *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control*, pages 157–168, 2019.
[29] Ikke Dian Oktaviani and Aji Gautama Putrada. Knn imputation to missing values of regression-based rain duration prediction on bmkg data. *Jurnal Infotel*, 14(4):249–254, 2022.
[30] Faza Ghassani, Maman Abdurohman, and Aji Gautama Putrada. Prediction of smarthphone charging using k-nearest neighbor machine learning. In *2018 Third International Conference on Informatics and Computing (ICIC)*, pages 1–4. IEEE, 2018.
[31] Sufen Chen, Xueqiang Zeng, et al. Progressive sampling-based joint automatic model selection of machine learning and feature selection. *Journal of Artificial Intelligence Practice*, 4(1):30–38, 2021.
[32] Edy Syuryawan Saputra, Aji Gautama Putrada, and Maman Abdurohman. Selection of vape sensing features in iot-based gas monitoring with feature importance techniques. In *2019 Fourth International Conference on Informatics and Computing (ICIC)*, pages 1–5. IEEE, 2019.
[33] Hanlin Lu, Changchang Liu, Shiqiang Wang, Ting He, Vijaykrishnan Narayanan, Kevin S Chan, and Stephen Pasteris. Joint coreset construction and quantization for distributed machine learning. In *2020 IFIP Networking Conference (Networking)*, pages 172–180. IEEE, 2020.
[34] Aji Gautama Putrada, Irfan Dwi Wijaya, and Dita Oktaria. Overcoming data imbalance problems in sexual harassment classification with smote. *International Journal on Information and Communication Technology (IJoICT)*, 8(1):20–29, 2022.
[35] Günce Keziban Orman and Serhat Çolak. Similarity based compression ratio for dynamic network modelling. In *IEEE EUROCON 2021-19th International Conference on Smart Technologies*, pages 227–232. IEEE, 2021.
[36] Tzu-Tsung Wong and Po-Yang Yeh. Reliable accuracy estimates from k-fold cross validation. *IEEE Transactions on Knowledge and Data Engineering*, 32(8):1586–1594, 2019.
[37] Ruslan Seifullaev, Steffi Knorn, and Anders Ahlén. A comparative investigation of information loss due to variable quantization on parameter estimation of compound distribution. *IFAC-PapersOnLine*, 53(2):2379–2384, 2020.