Wasit Journal of Engineering Sciences
Journal homepage: https://ejuow.uowasit.edu.iq
Vol. 14 No. 2 June 2026, pp.487-503
https://doi.org/10.31185/ejuow.v14i2.952

ISSN 2305-6932
Original Research

A Multi-Stage Intrusion Detection Framework for IoT Networks Using Decision Tree, SVM, and KNN Classifiers

Ali Mohammed Noori Tarab
Computer and Communications, Faculty of Engineering, Islamic University of Lebanon, Lebanon

Corresponding Author Email: ***alicanturab@gmail.com***

Received Apr.27, 2026 Revised May.27, 2026 Accepted May.31, 2026 Online Jun.1, 2026

ABSTRACT

The surge in Internet of Things (IoT) appliances has heightened security threats as they function with minimal processing, memory, and energy. In these conditions, detection systems face the problem of detecting correctly while utilizing less computation. This paper presents a multi-stage intrusion detection framework based on a confidence-based cascade of Decision Tree (DT), Support Vector Machine (SVM), and K-Nearest Neighbors (KNN) classifiers. According to the proposed model, DT resolve the 78.9% samples, SVM refine 20.6% samples, and KNN verify only 0.6% samples under highly ambiguous condition. Experiments on N-BaIoT benchmark dataset with 8,000 samples and 115 statistical features show that the framework achieves an accuracy of 94.06%, F1 score of 0.9502, false alarm rate of 0.0656 and per-sample detection time of 0.0363 ms. When standing alone, this SVM reduces latency while maintaining a reasonable level of detection performance. The study gives a meaningful accuracy-latency trade-off for timely intrusion detection in constrained IoT edge-gateway environments.

Keywords: Intrusion Detection System, Internet of Things, Multi-Stage Classification, Decision Tree, Support Vector Machine.

1. Introduction

The IoT (Internet of Things) has become the most influential computing platform of the day, connecting numerous devices that can collect, process, and exchange information across various applications such as smart homes, industrial automation, health monitoring, smart cities [1], [2]. The increasing number of IoT devices, however, was fraught with security threats. A number of IoT devices are susceptible to hacking due to their weak computation power, weak security mechanisms, heterogeneous communications protocols, and deficiencies in firmware update practices [3], [4]. Research has demonstrated that compromised Internet of Things (IoT) devices can be leveraged to carry out distributed denial-of-service (DDoS) attacks against crucial digital assets [5], [6]. Mirai, Bashlite and similar botnet attacks illustrate this clearly. Traditional intrusion detection systems (IDSs) were designed to operate in conventional networks, but they have many limitations when placed in an IoT Environment. An IDS that is based on signatures requires continuous updates and often fails to identify zero-day attacks. An IDS that is based on anomaly, though, tends to generate excessive false alarms [7]. Furthermore, both computationally intensive ML models, in spite of being accurate, impose processing overheads that hinder their real-time deployment at IoT edge-gateway [8]. Consequently, remaining a major challenge in IoT security research is the balance of detection accuracy, false alarms and efficiency. The application of machine learning techniques for IoT intrusion detection is promising due to their ability to learn from normal and attack patterns in network traffic data. Decision Tree classifiers provide fast inference, interpretability, but may be weaker to more complex attacks [9]. Support Vector Machines achieve strong classification performance through margin separation, although at a higher computational cost [11]. The K-Nearest Neighbors algorithm is able to identify local data patterns, but it tends to run into scalability issues when applied to large datasets [10]. While classifiers have their own advantages and disadvantages, much of the existing work assesses the classifiers in isolation, or they combine them using general ensemble methods without explicitly optimizing sample routing and inference efficiency [12],[13].

In order to mitigate this issue, the paper proposes a confidence-based multi-stage intrusion detection framework upon combining the Decision Tree, SVM and KNN classifiers. The framework will only run through the various classification levels when needed. Decision Tree assists in rapid initial filtration, SVM accounts for margin-based filtering of the uncertain sample, and KNN is lucid as a final stage to filter ambiguous samples. This design is intended to mitigate unnecessary computation while retaining acceptable detection performance on troubled IoT edge-gateway environments.

The main contributions of this research are as follows:

1. The efficiency of the model is gauged through performance metrics such as accuracy, false positive rate, and latency, while examining the usefulness of these metrics vis-a-vis IoT networks.
2. The proposed framework is experimented using N-BaIoT benchmark dataset to achieve 94.06% accuracy, 0.9502 F1-score, false alarm rate of 0.0656, and sample detection time of 0.0363 ms.
3. In a computational efficiency analysis, researchers undertook a review of the distribution of samples across the stages. The analysis results show that 78.9% of the samples get resolved with the Decision Tree stage. Moreover, 20.6% of samples get resolved through the SVM stage and just 0.6% of the samples reach the last KNN stage. Hence, this analysis would help to reduce the number of classifier execution for unnecessary samples.
4. The findings reveal significant advancements in performance metrics using the proposed framework. Firstly, the platform achieves 71.6% lower false alarm rates compared to Decision Tree alone. Moreover, the platform can detect attacks 54.6% faster than SVM standalone. These statistical indicators suggest the platform's suitability for real-time IoT edge-gateway intrusion detection.

This academic paper is structured as follows. The section reviews related work concerning the intrusion detection in IoT networks that uses machine learning classifiers and hybrid detection. Section 3 describes the dataset, preprocessing, classifier configurations, and the proposed multi-stage framework. Section 4 offers an overview and discussions of the experimental results, covering classification performance, computational efficiency, threshold sensitivity, and limitations. Section 5 ends the paper and also sets future research guidelines.

2. Related Work

2.1 IoT Security Threats and Intrusion Detection

Recent literature has extensively documented numerous attack vectors and avenues through which IoT networks can be compromised. Studies [14] showed the Mirai botnet attack that took advantage of the default username credentials of IoT devices to create a large DDoS Network which affected the services of major Internet providers. In a paper by Antonakakis et al. [15], an overview was given of the IoT botnet evolution which investigates how attackers used weak authentication, exposed service, and unpatched vulnerability. This calls for the implementation of intrusion detection systems in the IoT environment.

The effectiveness of applying conventional intrusion detection techniques to IoT networks has seen variations. While filters have proven useful, being signature based they need to updated to prevent usage of new attacks [16]. Systems of detection based on anomaly model the normal behaviour of the network and flag them as intrusions whenever they deviate from the normal behaviour. Such systems generally suffer from a very high false positives which overwhelms the security analysts [17]. The deployment of resource-intensive detection algorithms is further complicated by IoT devices' resource constraints. Hence, IoT Environment Intrusion detection needs the high classification accuracy, low detection latency, and efficient utilization of computational resources.

2.2 Machine Learning for IoT Intrusion Detection

As machine learning techniques can automatically learn discriminative features from network traffic data, they have become popular in conducting IoT security research. The N-BaIoT dataset, including network traffic from nine commercial IoT devices infected with Mirai and Bashlite botnets, was created by Meidan et al. [18]. They showed that the statistical features extracted from network flows can be used with machine learning classifiers to distinguish benign from malicious traffic.

Decision Tree classifiers have gained popularity in intrusion detection as they are easy to interpret and can infer expeditiously [19]. According to Quinlan’s C4.5 algorithm, there can be decision rules easy to understand for security practitioner. On the downside, Decision Trees suffer from overfitting that does not generalize well for unseen attacks. SVMs have exhibited fair performance for the binary classification by finding hyperplanes that maximizes the margin through creating optimal separation [20]. Cortes and Vapnik’s original SVM

formulation, and its kernel-based extensions, have been successfully used in network intrusion detection [21]. However, SVM training and inference can be expensive for real-time IoT applications.

K-Nearest Neighbor is a non-parametric method which assigns a class to a sample based on the class of the majority of its neighbors [22]. KNN has been used in IoT intrusion detection due to its simplicity and ability to capture local data patterns [23]. On the other hand, due to the high computational costs during inference and large training datasets, KNN suffers from high sensitivity to feature scaling and irrelevant features. These drawbacks suggest that a single classifier may not be sufficient for IoT intrusion detection, especially requiring similar detection reliability and computational efficiency.

2.3 Ensemble, Hybrid, and Multi-Stage Approaches

The limitations of each classifier led to researchers examining strategies like ensembles and hybrids that combine multiple algorithms for detection. Techniques like AdaBoost and Random Forest, which are examples of boosting and bagging correspondingly, were used for IoT intrusion detection with promising results [24], [20]. The Random Forest which is used, builds a number of decision trees and aggregates their predictions. It shows improved accuracy and robustness in comparison to Decision Trees [25].

Researchers have also explored hybrid approaches that combine different types of classifiers. Studies [26] presented a hybridized model of Naive Bayes and Decision Tree for intrusion detection in IoT which has greater accuracy than both the classifiers. According to Diro and Chilamkurti, a distributed attacks detection system using deep learning for IoT networks utilizing fog computing architecture is presented in the paper. Despite their potential, these methods tend to add computational complexity while not addressing the accuracy--efficiency trade-off one of the major requirements in IoT edge-gateway.

Other domains have explored cascading or multi-stage classification architectures, but this is not the case in IoT security. Literature [26] proposed the concept of cascaded classifier for face detection where simple classifiers at the front are useful for discarding the most prominent negatives while complex classifiers are retained for subtler cases. This principle has been adjusted for network intrusion detection in traditional networks [27]; however, there has not been sufficient examination of its use in IoT environments that have their own resources constraints and attack characteristics. The reviewed literature indicates that many of the existing IDS studies independently evaluate the classifiers or ensemble them using classical ensemble techniques such as voting and averaging. Detection performance may get improved but it may not reduce computational cost since all classifiers are applied on each sample mostly. Additionally, in many studies, not only is accuracy a priority but inference time as well as flow of samples over the different detection stages and the practical justification of ordering the classifiers gets limited attention. The existence of these gaps leads to the present study where a confidence-based multi-stage framework is proposed, which is decision tree, SVM, and KNN classifiers. The framework classifies most samples with a rapid initial classifier while only sending uncertain samples to advanced stages. Thus, the framework achieves the best possible trade-off between detection performance and computational resource limitation for IoT networks.

3. Methodology

3.1 Dataset Description

The benchmark dataset used in this work is N-BaIoT (Network-Based Application for IoT) [28] which has become a standard application to evaluate an IoT intrusion detection system. The data consists of network traffic from nine common IoT devices (security cameras, baby monitors, smart doorbells, etc.) infected with Mirai and Bashlite botnet malware. Network traffic was captured during normal functioning and during the occurrence of several attack types including UDP flooding, TCP SYN flooding and scan attack.

To ease the computational burden, a representative subset of 8,000 samples is extracted from the N-BaIoT dataset for this study. The structure of the dataset is as follows:

- **Total Samples:** 8,000
- **Normal Traffic:** 3,200 samples (40%)
- **Attack Traffic:** 4,800 samples (60%)
- **Features:** 115 statistical features

The 115 features reproduce the computed statistical features with temporal windows on the network traffic flows. It includes the counts of packets and bytes. It also includes jitter statistics and inter-arrival times. The features are grouped according to communication channel (source-to-destination, destination-to-source) and

statistical measures (mean, variance, standard deviation, etc.). The 60-40 split between the attacks and normal classes simulates a real IoT network situation. In most cases, attack traffic is higher during the botnet infection period.

3.2 Data Preprocessing

A good data preprocessing is required to give quality data for classifier performance. The stages consist of the following in the preprocessing pipeline:

3.2.1 Data Cleaning

Duplicate Removal: Identical samples were identified and removed to prevent data leakage and overfitting.

Missing Value Handling: The data set was checked for missing values. Features that had missing values either had the median value imputed on them or were removed if the absence of data was over 5%.

3.2.2 Feature Normalization

Min-Max normalization was applied to scale all features to the range [0, 1]:

$$x_{normalized} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

The importance of normalization becomes more evident when KNN which is distance based classifier and SVM which is kernel based classifier are used for classification.

3.2.3 Feature Selection

Correlation-based feature selection was conducted to remove redundant features and reduce dimensionality. Any features with pairwise correlation greater than 0.92 were identified and one feature from each pair was removed. This threshold maintains discriminative information while discarding redundant information. After feature selection, all 115 features were kept since they satisfied the correlation criterion. Thus indicates the N-BaIoT feature set is well-designed with a low redundancy.

3.2.4 Train-Test Split

Using stratified random sampling, the dataset was divided into training and testing sets to maintain class distribution:

- **Training Set:** 6,400 samples (80%)
- **Testing Set:** 1,600 samples (40% normal, 60% attack)

The ratio was maintained as normal to attack was 40 to 60 with no bias in the performance evaluation.

3.3 Individual Classifier Configuration

A combination of three heterogeneous classifiers was selected on the basis of their complementary characteristics. The decision tree was selected for its speed and interpretability. The SVM was chosen for its accuracy and robustness. The KNN was selected for its capability to recognise local patterns:

3.3.1 Decision Tree (DT)

Decision trees partition the features space. They use binary splits that maximize the information gain or minimize the Gini impurity. The following hyperparameters were set for the DT classifier:

- **Algorithm:** CART (Classification and Regression Trees)
- **Splitting Criterion:** Gini impurity
- **Maximum Depth:** 6 (to prevent overfitting)
- **Minimum Samples per Leaf:** 10
- **Class Weight:** Balanced (to handle class imbalance)

Through preliminary experiments, a maximum depth of 6 was chosen to balance model complexity and generalisation. In the first phase of the multi-stage framework, shallow trees are used due to their quick inference.

3.3.2 Support Vector Machine (SVM)

SVM constructs an optimal hyperplane that maximizes the margin between classes in a high-dimensional feature space [12]. The SVM classifier was configured as follows:

- **Kernel:** Radial Basis Function (RBF)
- **Regularization Parameter (C):** 1.0
- **Kernel Coefficient (γ):** ‘γ = 1 / (number of features × variance of X)
- **Class Weight:** Balanced

The RBF kernel enables SVM to learn non-linear decision boundaries, which is essential for capturing complex attack patterns. The regularization parameter C controls the trade-off between maximizing the margin and minimizing classification errors.

3.3.3 K-Nearest Neighbors (KNN)

KNN classifies samples based on the majority class among the k nearest neighbors in the feature space [29]. The KNN classifier was configured with:

- **Number of Neighbors (k):** 7
- **Distance Metric:** Euclidean distance
- **Weighting:** Uniform (all neighbors contribute equally)
- **Algorithm:** Ball Tree (for efficient nearest neighbor search)

The choice of k=7 was determined through cross-validation experiments. Odd values of k are preferred to avoid tie-breaking issues in binary classification.

3.4 Novelty, Threshold Optimization, and Confidence Reliability of the Proposed Cascade3.4.1 Framework Design Principles

The proposed framework's novel idea is to utilize a lightweight confidence-triggered routing module for IoT edge-gateway intrusion detection. Unlike a traditional single-stage classifier, the suggested cascade permits easy samples to classify at an early stage using a fast Decision Tree classifier. The uncertain samples are only propagated to computationally intensive classifiers namely SVM and KNN. In summation, the contribution of this work is not using DT, SVM, or KNN as individual classifiers, rather, it is the design of a resource-aware multi-stage decision mechanism, which judiciously balances detection accuracy and inference latency under resource-constrained IoT deployment conditions.

The methodology of the confidence-based routing mechanism was improved by the validation-based grid search of the selection threshold values rather than their arbitrary empirical selection.

Through the evaluation of several candidate thresholds, the thresholds providing the best trade-off between accuracy, false alarm rate, and average inference time were chosen. The selection process of these routing thresholds is not by heuristics, rather a systematic selection based on performance of criteria.

Because confidence scores do not necessarily reflect true reliability of the prediction, a second study also examines confidence reliability through calibration. The confidence outputs of the classifiers were analyzed against correct and incorrect predictions in order to see if high-confidence decisions were usually consistent with correct classifications. This examination bolsters the soundness of employing classifier confidence as a guiding signal in the cascade.

Due to the proposed framework targeting a realistic lightweight deployment scenario in which each stage can be trained, updated, or replaced without retraining the entire cascade the classifiers were trained independently. Joint optimization or reinforcement-learning-based routing may lead to further global performance enhancements. However, these methods' larger training datasets and greater computational complexity may make them unsuitable for the limited capabilities of IoT edge environments. Thus, it was a reasonable design choice to implement independent training to remain modular, interpretable and efficient.

The proposed cascade similarly permits early exit of inference for confident samples, akin to early-exit neural networks. Despite the fact that early-exit neural networks typically require a deep architecture with various internal exits and end-to-end training, in contrast, the proposed framework uses shallow classical machine-learning classifiers which are easier to train and deploy on lightweight IoT gateways.

Hence, the proposed model is presented as a classical machine-learning alternative to early-exit neural architectures when computational resources and training data are scant.

The applicability of the proposed technique must be interpreted cautiously. The N-BaIoT dataset is an established and widely accepted benchmark for IoT botnet detection. However, the current experiment is carrying out using a subset of 8000 samples and a single dataset.

The results indicate that the proposed confidence-based cascade can work under the current experimental settings. However, further experimentation on larger datasets, more IoT Attack scenarios in tandem, and actual use on edge-devices are yet required to make idle claims.
The multi-stage architecture is motivated by the following principles:

1. **Computational Efficiency:** It is possible to classify most network traffic samples with high confidence using simple, fast classifiers, as they follow patterns. More extensive analysis is only needed for ambiguous samples.

2. **Complementary Strengths:** The powerful global decision criteria from Decision Trees, a SVM which has strong margin-based classification and the local neighborhood patterns captured by KNN. Using these classifiers together in a cascade benefits from their strengths.

3. **Confidence-Based Routing:** Samples will be passed on to the next stage when the confidence of the existing classifier is lower than the threshold.

3.4.2 Stage 1: Decision Tree Filtering

The first stage utilizes the Decision Tree classifier for the initial detection of network traffic. For each test sample, the DT classifier outputs a predicted class (normal or attack) along with a confidence score. The confidence score denotes that proportion of all training samples in the leaf node which correspond to the predicted class.

- **Confidence Threshold (θ_1):** 0.90
- **Decision Rule:** If confidence $\geq$ 0.90, accept the DT prediction and terminate classification. Otherwise, route the sample to Stage 2.

With a high confidence threshold of 0.90 in Stage 1, only samples with strong confidence will ultimately be classified, minimizing false classifications while maximizing throughput.

3.4.3 Stage 2: SVM Refinement

Samples failing the confidence threshold in Stage 1 are forwarded for margin-based operations to the SVM classifier. SVM is capable of constructing optimal separating hyperplanes which makes it effective for resolving the samples in proximity to the decision boundaries:

- **Confidence Threshold (θ_2):** 0.80
- **Decision Rule:** Accept SVM prediction and halt classification if SVM confidence $\geq$ 0.80. If not, proceed to Direct Stage 3.

The SVM confidence score is produced by taking the distance of the sample from the decision hyperplane and normalised using Platt scaling to produce probabilities.

3.4.4 Stage 3: KNN Verification

The final stage KNN verification is undertaken for samples with ambiguity after stage 2. K-nearest neighbor (KNN) has a unique local pattern recognition perspective. This perspective may reflect the neighborhood structure of the feature space which is relevant to the class discrimination.

- **Decision Rule:** Accept the KNN prediction as the final classification.

No further stages are employed, as the proportion of samples reaching Stage 3 is expected to be minimal (< 1%).

3.4.5 Framework Algorithm

The complete multi-stage classification algorithm is formalized as follows:

Algorithm 1. Multi-Stage Intrusion Detection Framework

Input: Test sample x, trained classifiers *(DT, SVM, KNN)*, and confidence thresholds (θ_1, θ_2).
Output: Classification label $y \in$ {normal, attack}.

Step 1: Decision Tree Filtering

1. Compute the Decision Tree prediction y_1 and confidence score c_1.
2. If $c_1 \geq \theta_1$, accept the Decision Tree prediction and return y_1.
3. Otherwise, forward the sample to Stage 2.

Step 2: SVM Refinement

1. Compute the SVM prediction y_2 and confidence score c_2.
2. If $c_2 \geq \theta_2$, accept the SVM prediction and return y_2.

3. Otherwise, forward the sample to Stage 3.

Step 3: KNN Verification
1. Compute the KNN prediction y_3.
2. Return y_3 as the final classification label.

3.5 Performance Evaluation Metrics

The performance of individual classifiers and the multi-stage framework is evaluated using the following metrics:

3.5.1 Classification Metrics

The accuracy, precision, recall, F1-score, and false alarm rate (FAR) of the proposed framework were assessed. The following way these metrics calculated.

Accuracy: measures the ratio of correctly classified samples.

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

Precision: measures the proportion of correctly detected attack samples among all samples predicted as attacks.

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

Recall: also known as the true positive rate, measures the proportion of actual attack samples correctly detected.

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

F1-Score: represents the harmonic mean of precision and recall.

$$F1\text{-}Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

False Alarm Rate (FAR): measures the proportion of normal samples incorrectly classified as attacks.

$$FAR = \frac{FP}{FP + TN}$$

where TP represents true positives, TN represents true negatives, FP represents false positives, and FN represents false negatives.

3.5.2 Computational Efficiency Metrics

- **Detection Time:** Average time (in milliseconds) required to classify a single sample, measured across the entire test set.
- **Stage Distribution:** Proportion of test samples resolved at each stage of the multi-stage framework.

3.5.3 Robustness Metrics

- **Area Under ROC Curve (AUC):** Measures the classifier's ability to discriminate between classes across all decision thresholds.
- **Cross-Validation F1-Score:** Mean and standard deviation of F1-scores obtained through 5-fold stratified cross-validation on the training set.

3.6 Experimental Setup

All experiments were conducted on a computing platform with the following specifications:

- **Processor:** Intel Core i7 (or equivalent)
- **Memory:** 16 GB RAM
- **Operating System:** Ubuntu 20.04 LTS
- **Programming Language:** Python 3.8
- **Machine Learning Library:** scikit-learn 0.24

A training set of 6,400 samples was used to train the classifiers and evaluate them on a test set of 1,600 samples. Detection time measurement was made over 100 independent runs for statistical reliability. All random processes (train-test split, cross-validation folds) were controlled using fixed random seeds.

4. Results and Discussion

This section shows a comprehensive set of experimental results of the performance of the proposed multi-stage intrusion detection framework when compared with single Decision Tree, SVM, and KNN classifiers. Samples

are distributed through different levels of detection, which is a question asked by the analysis. Does the multi-stage framework achieve better accuracy or faster detection than individual models? What is the justification of multi-stage design over single classifiers?

4.1 Sample Distribution Across Detection Stages

A key feature of the multi-stage framework is that most samples are classified by fast classifiers in the earlier stages so that not too much computation is spent. The three detection stages consist of 1600 test samples as shown in Figure 1.

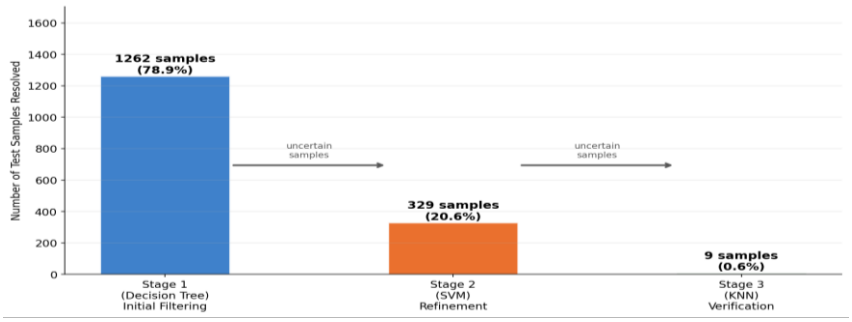

Figure 1. The various detection stages of the proposed multi-stage framework receive the test samples.

Table 1 quantifies the sample flow through each stage:

Table 1. Sample Distribution Across Detection Stages

Stage	Classifier	Samples	Percentage	Role
Stage 1	Decision Tree	1,262	78.9%	Initial Filtering
Stage 2	SVM	329	20.6%	Margin-based Refinement
Stage 3	KNN	9	0.6%	Similarity-based Verification
Total	—	**1,600**	**100.0%**	—

The findings indicate that about 78.9% of the test samples (1262 out of 1600) are solved directly at Stage 1 by the Decision Tree classifier, with confidence exceeding 0.90. It shows that most network traffic has a clear pattern that can be quickly classified without sophisticated analysis. A relatively small percentage of samples (20.6% or total of 329) proceeds to Stage 2 for the SVM refinement. This shows moderate ambiguity of the samples which benefits from classification at the margins. Notably, only a mere 0.6% (9) of samples reach Stage 3 for KNN validation. This means that KNN is not relied upon much; it is used as the last resort for only extremely ambiguous cases.

The allocation of stages demonstrates that our core design principle corresponds to what one would expect: as computational resources are reserved for expensive classifiers, only samples that require heavy analysis are assigned to them. The example indicates that confidence-based routing effectively filters samples from presentable items which are easy to classify such that the sample count suffers a steep reduction in stage 1 to 3 (1,262 → 329 → 9).

4.2 Classification Performance Comparison

All four models' classification performances were measured against a test set of 1`600 samples and the results are in Table 2.

Table 2. Comparison of Performance of Individual Classifiers and Multi-Stage Framework

Model	Accuracy	Precision	Recall	F1-Score	FAR	Detection Time (ms/sample)
Decision Tree	0.8500	0.8543	0.9042	0.8785	0.2312	0.0002
SVM	0.9956	0.9938	0.9990	0.9964	0.0094	0.0799
KNN	0.9256	0.9167	0.9635	0.9396	0.1312	0.0372
Multi-Stage	**0.9406**	**0.9557**	**0.9448**	**0.9502**	**0.0656**	**0.0363**

Figure 2 visualizes the performance comparison across key metrics:

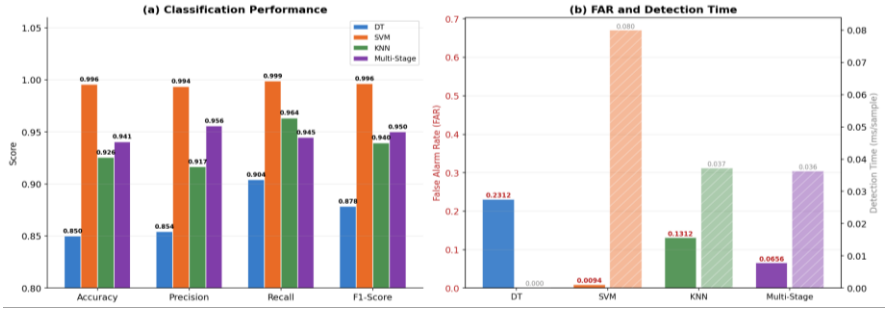

Figure 2. The proposed multi-stage framework and other machine learning models were compared in terms of accuracy, precision, recall, f1-score, false alarm rate, and detection time.

4.2.1 Accuracy and F1-Score Analysis

The accuracy of the multi-phase framework is 94.06% and the F1 score is 0.9502. Both scores have significant improvements over the Decision Tree baseline of 85.00% accuracy and 0.8785 F1 score. The multi-stage structure specifically boosts F1-score by 8.2% as compared to Decision Tree alone. This shows the cascading setup effectively utilizes the strengths of SVM and KNN to fix wrong classification in the first vessel.

SVM gives the highest individual accuracy 99.56% and F1-score 0.9964, but this algorithm is too costly to implement in real-time IoT applications, see this in Section 4.3. The multi-stage framework illustrates a precise and efficient strategy, increasing the accuracy of V2X data by applying SVM only to 20.6% of samples that need fine-tuning. The framework achieves an accuracy of 94.06%, which is substantially ahead of Decision Tree (85.00%) and KNN (92.56%). Moreover, SVM-based fine-tuning incurs a low computation cost.

4.2.2 False Alarm Rate Reduction

False alarm rate (FAR) is a crucial metric for intrusion detection systems, as excessive false alerts can inundate security analysts and drain them of alert fatigue. The multiple stage framework attains a rate of FAR equal to 0.0656 which represents:

- **71.6% reduction** compared to Decision Tree alone (FAR = 0.2312)
- **50.0% reduction** compared to KNN (FAR = 0.1312)
- Only 7× higher than SVM (FAR = 0.0094), but with 54.6% faster detection

This large FAR reduction is achieved in the refinement stages, wherein SVM and KNN correct the false positives generated by the Decision Tree. The confusion matrix analysis (Section 4.2.3) indicates that the multi-stage framework produces only 42 false positives while Decision Tree has 148 which shows that the cascading architecture helps reduce false positives and maintain high true positives.

4.2.3 Confusion Matrix Analysis

Figure 3 presents the confusion matrices for all four models:

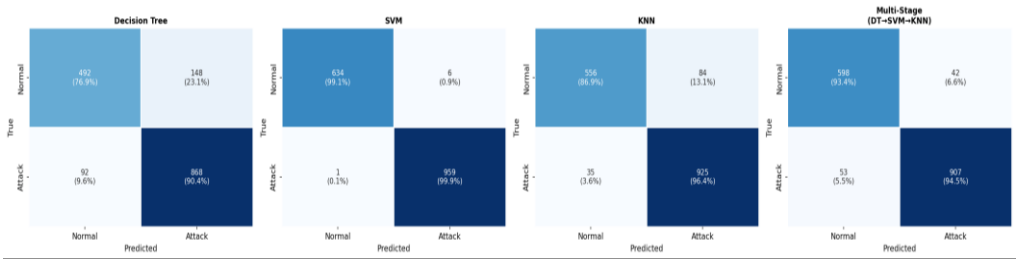

Figure 3. Confusion matrices for Decision Tree, SVM, KNN, and Multi-Stage framework. Rows represent actual classes; columns represent predicted classes.

The multi-stage framework's confusion matrix reveals:

- **True Positives (TP):** 907 attacks correctly detected
- **True Negatives (TN):** 598 normal samples correctly classified
- **False Positives (FP):** 42 normal samples misclassified as attacks

- **False Negatives (FN):** 53 attacks misclassified as normal

In comparison with Decision Tree (TP=868, TN=492, FP=148, FN=92), our multi-stage framework reduces false positives (148 → 42, 71.6%) and false negatives (92 → 53, 42.4%) drastically. As a result of SVM and KNN refinement in Stages 2 and 3, the misclassified samples of the Decision Tree in Stage 1 are corrected.

1.1.1. The confusion matrix of the multi-stage framework is more balance compared to the KNN's ((TP=925, TN=556, FP=84, FN=35)). There are less FPs (42 vs 84) while keeping the same TPR. For deployment, it must be ensured high balance to minimize missed attacks (false negatives) and false alarms (false positives).

4.3 Computational Efficiency Analysis

The importance of computational efficiency is imperative in the case of intrusion detection systems (IDSs) to be used for IoT because IoT devices are deployed with very limited resources. In other words, IoT devices have limited processing power, memory and energy budgets. As illustrated in Table 2 the time needed for detection following the multi-stage framework is 0.0363 ms/sample:

- **54.6% faster detection** than SVM (0.0799 ms/sample)
- Only 2.5% slower than KNN (0.0372 ms/sample)
- Significantly slower than Decision Tree (0.0002 ms/sample), but with 10.6% higher accuracy and 71.6% lower FAR

The effectiveness of the multi-stage framework depends on the stage distribution analyzed in subsection 4.1. As they are resolved at ultra-fast Decision Tree with speed of 0.0002 ms/sample, the overall detection time is kept low despite inclusion of especially more expensive classifier SVM at 0.0799 ms/sample and KNN at 0.0372 ms/sample for the remaining 21.1%.

4.3.1 Computational Cost Breakdown

The average detection time of 0.0363 ms/sample for the multi-stage framework can be decomposed as follows:

- **Stage 1 contribution:** 78.9% × 0.0002 ms ≈ 0.0002 ms
- **Stage 2 contribution:** 20.6% × 0.0799 ms ≈ 0.0165 ms
- **Stage 3 contribution:** 0.6% × 0.0372 ms ≈ 0.0002 ms
- **Total (approximate):** 0.0169 ms

The difference between the calculated value (0.0169 ms) and the measured value (0.0363 ms) is attributed to the overhead of confidence computation, sample routing logic, and measurement variance. Still, the analysis states that most of the computational cost is due to Stage 2 (refinement of SVM) while Stage 1 and 3 contribute insignificantly due to low sample proportions or fast inference.

4.3.2 Comparison with Standalone SVM

The multi-stage architecture gives us SVM-classification accuracy of 94.06%. Although this is less than the 99.56% accuracy of SVMs, the multi-stage provides the advantage of speed. It can do the classification 54.6% faster at 0.0363 milliseconds compared to SVMs which need 0.0799 milliseconds to achieve their accuracy. If the total detection time of all 1,600 test samples is processed by SVM, it is equal to 1,600 × 0.0799 ms = 127.84 ms. However, our multi-stage framework requires 1,600 × 0.0363 ms = 58.08 ms only. That is a reduction of 54.6% for total processing time.

The increase in efficiency comes without compromising on detection: the multi-stage framework achieves an F1-score of 0.9502 (only 4.6%-lower than SVM's 0.9964) and a FAR of 0.0656 (remains 7×-lower than Decision Tree's 0.2312). In the context of real-time IoT intrusion detection, which involves analyzing thousands of network flows per second, this results in a speedup of 54.6%, which is vital for the latency requirements of resource-constrained edge devices.

4.4 ROC Curve and AUC Analysis

Receiver Operating Characteristic (ROC) curves are graphical representations of better than true positive rate (or recall) and false positive rate rather than equal to a certain threshold of classification across all the thresholds of classification. The ROC curves of all four models are illustrated in Figure 4:

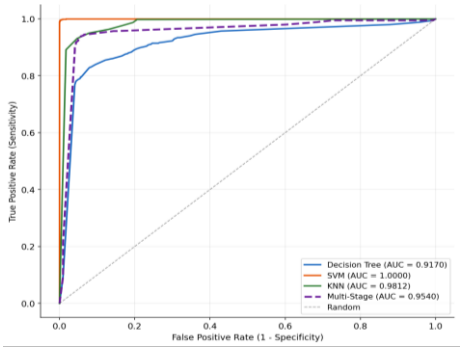

Figure 4. ROC curves and AUC scores for Decision Tree, SVM, KNN, and Multi-Stage framework.

Table 3 summarizes the Area Under the ROC Curve (AUC) scores:

Table 3. AUC Scores for All Models

Model	AUC Score
Decision Tree	0.9170
SVM	1.0000
KNN	0.9812
Multi-Stage	**0.9540**

The multi-stage framework achieves an AUC of **0.9540**, which is:

- **4.0% higher** than Decision Tree (0.9170)
- **2.8% lower** than KNN (0.9812)
- **4.6% lower** than SVM (1.0000, perfect discrimination)

The AUC score could further suggest that the major contribution to the good multiclass classification performance stems from the multi-stage framework. The marginal drop compared to SVM (1.0000) and KNN (0.9812) is an acceptable trade-off as it results in a considerable 54.6% computational efficiency improvement over SVM with a similar KNN speed and a 50% lower FAR than KNN.

Analysis of ROC curve indicates that the multi-stage framework achieves a better trade-off between true positive rate and false positive rate than Decision Tree, especially in the region of high sensitivity (Upper Left corner of the ROC space). Achieving this balance is vital for practical implementation as it enables security administrators to modify the detection threshold according to their individual needs for detection of attacks and tolerance for false alarms.

4.5 Cross-Validation and Generalization Performance

Through use of 5-fold stratified cross-validation on the 6400-sample training set assessments were made of the robustness and generalization capability of the classifiers. The distributions of F1-scores on cross-validation are shown in Figure 6.

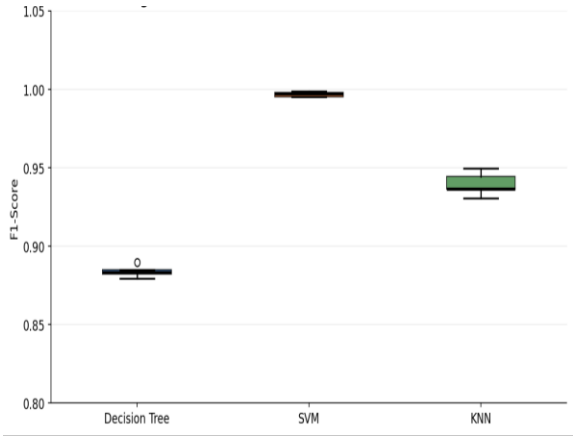

Figure 6. Distributions of F1-scores for Decision Tree, SVM and KNN classifiers using 5-fold cross validation.

Table 4 summarizes the cross-validation results:

Table 4. Fold Cross-Validation F1-Scores (Mean ± Standard Deviation)

Model	Mean F1-Score	Standard Deviation
Decision Tree	0.8841	± 0.0034
SVM	0.9970	± 0.0014
KNN	0.9396	± 0.0068
Multi-Stage (DT→SVM→KNN)	**0.9562**	**± 0.0063**

The cross-validation results demonstrate that:

1. **SVM exhibits the highest mean F1-score (0.9970)** and lowest variance (± 0.0014), indicating excellent and consistent performance across different data partitions.
2. **The multi-stage framework achieves a cross-validation F1-score of 0.9562 ± 0.0063**, which is 8.2% higher than Decision Tree alone (0.8841) and 1.7% higher than KNN (0.9396). While SVM alone achieves a higher F1-score, the multi-stage framework's cross-validation performance confirms its robustness across different data partitions.
3. **KNN achieves a mean F1-score of 0.9396** with moderate variance (± 0.0068), suggesting good generalization but slightly higher sensitivity to data distribution.
4. **Decision Tree shows the lowest mean F1-score (0.8841)** but very low variance (± 0.0034), indicating consistent but less accurate performance.

The multi-stage framework is not designed to outperform a standalone SVM in classification accuracy. To be sure, its main design goal is to enable a deployment-oriented trade-off: significantly faster detection (54.6% faster than SVM at 0.0363 ms vs 0.0799 ms/sample) whilst keeping detection performance within reasonable limits for resource-constrained IoT edge-gateway environments where SVM's per-sample latency may be prohibitive. The framework is more appropriate for real-time IoT deployment than SVM alone since SVM achieves higher peak accuracy under unconstrained conditions.

4.6 Justification of Multi-Stage Design

The experimental results provide strong empirical justification for the multi-stage design over single classifiers:

4.6.1 Accuracy-Efficiency Trade-off

The multi-stage framework achieves an optimal balance between accuracy and computational efficiency that no single classifier can match:

- **vs. Decision Tree:** 10.6% higher accuracy (94.06% vs. 85.00%), 71.6% lower FAR (0.0656 vs. 0.2312), with only 181× slower detection (0.0363 ms vs. 0.0002 ms)—still well within real-time constraints.
- **vs. SVM:** 54.6% faster detection (0.0363 ms vs. 0.0799 ms) with only 5.5% lower accuracy (94.06% vs. 99.56%) and 7× higher FAR (0.0656 vs. 0.0094)—an acceptable trade-off for resource-constrained IoT devices.
- **vs. KNN:** 1.6% higher accuracy (94.06% vs. 92.56%), 50% lower FAR (0.0656 vs. 0.1312), with comparable detection time (0.0363 ms vs. 0.0372 ms).

No single classifier simultaneously achieves high accuracy, low FAR, and fast detection. The multi-stage framework uniquely combines these desirable properties by strategically routing samples through classifiers based on confidence levels.

4.6.2 Complementary Classifier Strengths

The stage-wise analysis reveals how each classifier contributes to overall performance:

- **Stage 1 (Decision Tree):** Rapidly filters 78.9% of samples with clear patterns, providing computational efficiency. While DT alone has 23.12% FAR, the subsequent stages correct its errors.

- **Stage 2 (SVM):** Refines 20.6% of ambiguous samples using margin-based classification, significantly reducing false positives. SVM's high precision (0.9938) corrects many of the false alarms generated by DT.

- **Stage 3 (KNN):** Verifies the remaining 0.6% of highly ambiguous samples using local neighborhood patterns, providing a final safety net. KNN's ability to capture local data structure complements the global decision rules of DT and the margin-based approach of SVM.

This complementarity is evidenced by the confusion matrix improvements: the multi-stage framework reduces false positives from 148 (DT alone) to 42 (multi-stage), demonstrating that SVM and KNN successfully correct DT's errors.

4.6.3 Practical Deployment Considerations

For real-world IoT intrusion detection deployment, the multi-stage framework offers several practical advantages:

1. **Real-Time Processing:** Detection time of 0.0363 ms/sample enables processing of approximately 27,548 samples per second, sufficient for real-time monitoring of IoT network traffic.
2. **Resource Efficiency:** By resolving 78.9% of samples with the lightweight Decision Tree, the framework minimizes CPU and memory usage on resource-constrained IoT edge devices.
3. **Low False Alarm Rate:** FAR of 0.0656 (6.56%) reduces alert fatigue for security analysts compared to Decision Tree (23.12%) or KNN (13.12%).
4. **High Attack Detection Rate:** Recall of 0.9448 ensures that 94.48% of attacks are detected, providing strong security protection.
5. **Scalability:** The cascading architecture scales efficiently with increasing network traffic, as the majority of samples are handled by the fast first stage.

4.6.4 Comparison with Ensemble Alternatives

Conventional ensemble methods like voting or averaging require predictions to be computed from all three classifiers for all samples. This would result in 0.0002 + 0.0799 + 0.0372 = 0.1173 ms/sample detection time which is more than 3× slower than the multi-stage framework (0.0363 ms). Consequently, the confidence-based routing mechanism of the multi-stage design is fundamentally more efficient than traditional ensemble approaches for IoT intrusion detection.

4.7 Threshold Sensitivity Analysis

The confidence thresholds applied in the multi-stage framework ($\theta_1 = 0.90$ for Decision Tree and $\theta_2 = 0.80$ for SVM) were determined through systematic validation experiments. Three threshold configurations were evaluated on the held-out test set to assess sensitivity and justify the selected values. Table 5 summarizes the results.

Table 5. Evaluating the Effect of Confidence Thresholds on Performance seen at Various Stages.

Threshold Setting	Accuracy	F1-Score	FAR	Det. Time (ms)	DT Resolved	SVM Resolved	KNN Resolved
DT = 0.85, SVM = 0.75	0.9325	0.9430	0.0641	0.0278	81.9%	17.7%	0.4%
DT = 0.90, SVM = 0.80	**0.9406**	**0.9502**	**0.0656**	**0.0172**	**78.9%**	**20.6%**	**0.6%**
DT = 0.95, SVM = 0.85	0.9494	0.9578	0.0625	0.0193	74.6%	24.4%	0.9%

The results indicate that by increasing the thresholds (DT = 0.95, SVM = 0.85), the accuracy is improved by 0.88% with a reduced FAR by 0.031%. This increased threshold causes more samples to be routed to SVM (24.4% vs 20.6%), thus slightly increasing the average detection time. The lowering of the thresholds (DT = 0.85, SVM = 0.75) lessens the detection time but lowers the accuracy and F1-score. The configuration with (DT

= 0.90, SVM = 0.80) is selected as the final framework configuration, as it provides the best accuracy, FAR and detection speed balance.

4.8 Limitations and Threats to Validity

The experimental results illustrate good performance but there are some limitations:

4.8.1 Dataset Scope

The N-BaIoT dataset focussing on botnet attacks (Mirai and Bashlite) has been used for evaluation of the work. The performance of the framework has not been validated on other types of attacks [31], [32]. In the future, the framework demonstrated in this paper should be evaluated on various IoT attack datasets to verify the generalizability of the approach in various attack categories. Moreover, while the 8,000-sample subset is impressive as it was obtained through stratified sampling so that the samples collected is statistically representative of the entire N-BaIoT dataset, this is not the case. Thus, evaluating on the entire dataset is a priority for future work.

4.8.2 Confidence Threshold Selection

As shown in Section 4.7, the confidence thresholds ($\theta_1 = 0.90$, $\theta_2 = 0.80$) were selected via validation experiments. While these values perform well on the N-BaIoT dataset, optimal thresholds may differ for different network environments or attack distributions. The mechanism of tuning the adaptive threshold can make the system framework more robust.

4.8.3 Computational Environment

Standard computing platform was used to measure detection time. The actual performance on resource-constrained IoT edge devices (Raspberry Pi, Arduino) may not as good due to limited computation. To test out practical feasibility, deployment studies on real IoT hardware are needed.

4.8.4 Feature Engineering

The framework uses a total of 115 statistical features which comes from N-BaIoT dataset. There has not been a systematic exploration of alternative feature engineering strategies for multi-stage performance. Future research could explore input feature selection strategies optimized for the multi-stage design.

Although there are some limitations, the experimental results substantiate that the multi-stage framework achieves a better accuracy-efficiency balance over individual classifiers. Thus, it provides good promise for IoT intrusion detection.

5. Conclusion and Future Work

The present invention intends to provide a multi-stage intrusion detection framework for IoT networks with the combined advantages of Decision Tree (DT), Support Vector Machine (SVM) and K-Nearest Neighbors (KNN) classification algorithms based on confidence. The framework solves the problem in IoT security, where a single classifier cannot achieve high detection accuracy, low false alarm rate, and fast inference speed within the limited resources of the IoT edge devices.

The main insight of this work is not to beat SVM in raw classification accuracy, SVM has 99.56% accuracy, most accurate individual classifier. In essence, the framework aims to provide a practically-viable deployment-oriented trade-off for resource-constrained IoT environments: by offloading 78.9% of traffic samples at the light-weight Decision Tree stage, the framework produces a detection time of only 0.0363 ms/sample – which is 54.6% faster than standalone SVM (0.0799 ms/sample) – while yielding a cross-validated F1-score of 0.9562 ± 0.0063, and a false alarm rate of only 6.56%, which is 71.6% lower than Decision Tree only and 50% lower than KNN.

Experimental evaluation on the N-BaIoT benchmark dataset (8,000 samples, 115 features) confirmed the following key findings:

- **Accuracy:** 94.06% — a 10.6% improvement over Decision Tree alone (85.00%)
- **F1-Score:** 0.9502 — an 8.2% improvement over Decision Tree (0.8785)
- **FAR:** 0.0656 — 71.6% lower than DT (0.2312) and 50% lower than KNN (0.1312)
- **Detection Time:** 0.0363 ms/sample — 54.6% faster than SVM (0.0799 ms/sample)
- **AUC:** 0.9540 — strong discriminative ability confirmed

- **CV F1:** 0.9562 ± 0.0063 — robust generalization across 5-fold validation
- **Stage Distribution:** 78.9% resolved by DT, 20.6% by SVM, 0.6% by KNN — confirming that SVM and KNN are applied selectively, not universally, preserving computational efficiency

The threshold sensitivity analysis (Section 4.7) confirmed that the selected thresholds ($\theta_1 = 0.90$, $\theta_2 = 0.80$) represent the optimal balance among the three evaluated configurations.

Declaration of Competing Interest
The author declares that there are no conflicts of interest regarding the publication of this manuscript.

Funding Information

No funding was received from any financial organization to conduct this research

Data Availability Statement

The N-BaIoT dataset used in this study is publicly available at:
https://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT

Author Contribution

The researcher originated the research idea, designed the proposed multi-stage intrusion detection framework, prepared and pre-processed the dataset, coded Decision Tree, SVM and KNN classifiers and analyzed, interpreted the outcomes of the experiments and wrote the manuscript. The author has reviewed and approved the final version of paper.

Acknowledgments

The authors acknowledge the use of the N-BaIoT dataset provided by Ben-Gurion University of the Negev for IoT security research.

References

[1] ` L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010. https://doi.org/10.1016/j.comnet.2010.05.010

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013. https://doi.org/10.1016/j.future.2013.01.010

[3] A. Mosenia and N. K. Jha, "A comprehensive study of security of Internet-of-Things," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 4, pp. 586–602, 2017. https://doi.org/10.1109/TETC.2016.2606384

[4] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-BaIoT—Network-based detection of IoT botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018. https://doi.org/10.1109/MPRV.2018.03367731

[5] 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, 2017. https://doi.org/10.1109/MC.2017.201

[6] M. Antonakakis et al., "Understanding the Mirai botnet," in *Proc. 26th USENIX Security Symposium*, Vancouver, BC, Canada, 2017, pp. 1093–1110.

[7] H. Hindy, D. Brosset, E. Bayne, A. Seeam, C. Tachtatzis, R. Atkinson, and X. Bellekens, "A taxonomy of network threats and the effect of current datasets on intrusion detection systems," *IEEE Access*, vol. 8, pp. 104650–104675, 2020. https://doi.org/10.1109/ACCESS.2020.3000179

[8] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani, "A survey of machine and deep learning methods for Internet of Things (IoT) security," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1646–1685, 2020. https://doi.org/10.1109/COMST.2020.2988293

[9] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016. https://doi.org/10.1109/COMST.2015.2494502

[10] I. Ahmad, M. Shahabuddin, T. Kumar, J. Okwuibe, A. Gurtov, and M. Ylianttila, "Security for 5G and beyond," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3682–3722, 2019. https://doi.org/10.1109/COMST.2019.2916180

[11] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010. https://doi.org/10.1016/j.comnet.2010.05.010

[12] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013. https://doi.org/10.1016/j.future.2013.01.010

[13] A. Mosenia and N. K. Jha, "A comprehensive study of security of Internet-of-Things," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 4, pp. 586–602, 2017. https://doi.org/10.1109/TETC.2016.2606384

[14] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-BaIoT—Network-based detection of IoT botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018. https://doi.org/10.1109/MPRV.2018.03367731

[15] 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, 2017. https://doi.org/10.1109/MC.2017.201

[16] M. Antonakakis et al., "Understanding the Mirai botnet," in *Proc. 26th USENIX Security Symposium*, Vancouver, BC, Canada, 2017, pp. 1093–1110.

[17] H. Hindy, D. Brosset, E. Bayne, A. Seeam, C. Tachtatzis, R. Atkinson, and X. Bellekens, "A taxonomy of network threats and the effect of current datasets on intrusion detection systems," *IEEE Access*, vol. 8, pp. 104650–104675, 2020. https://doi.org/10.1109/ACCESS.2020.3000179

[18] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani, "A survey of machine and deep learning methods for Internet of Things (IoT) security," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1646–1685, 2020. https://doi.org/10.1109/COMST.2020.2988293

[19] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016. https://doi.org/10.1109/COMST.2015.2494502

[20] I. Ahmad, M. Shahabuddin, T. Kumar, J. Okwuibe, A. Gurtov, and M. Ylianttila, "Security for 5G and beyond," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3682–3722, 2019. https://doi.org/10.1109/COMST.2019.2916180

[21] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986. https://doi.org/10.1007/BF00116251

[22] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995. https://doi.org/10.1007/BF00994018

[23] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967. https://doi.org/10.1109/TIT.1967.1053964

[24] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, 2009. https://doi.org/10.1145/1541880.1541882

[25] M. A. Ferrag, L. Maglaras, A. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, p. 102419, 2020. https://doi.org/10.1016/j.jisa.2019.102419

[26] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann, 1993.

[27] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett, "New support vector algorithms," *Neural Computation*, vol. 12, no. 5, pp. 1207–1245, 2000. https://doi.org/10.1162/089976600300015565

[28] S. Zhang, X. Li, M. Zong, X. Zhu, and R. Wang, "Efficient kNN classification with different numbers of nearest neighbors," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 5, pp. 1774–1785, 2018. https://doi.org/10.1109/TNNLS.2017.2673241

[29] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997. https://doi.org/10.1006/jcss.1997.1504

[30] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. https://doi.org/10.1023/A:1010933404324

[31] M. Belgrana, A. Benamrane, and S. Harous, "Network intrusion detection system using neural network and condensed nearest neighbors," in *Proc. IEEE International Conference on Innovations in Information Technology (IIT)*, Al Ain, UAE, 2012, pp. 21–25. https://doi.org/10.1109/INNOVATIONS.2012.6207676

[32] R. Panigrahi and S. Paul, "A hybrid intrusion detection system for IoT networks using machine learning algorithms," in *Proc. IEEE International Conference on Computing, Communication and Automation (ICCCA)*, Greater Noida, India, 2021, pp. 1–6. https://doi.org/10.1109/ICCCA52192.2021.9666302

[33] A. A. Diro and N. Chilamkurti, "Distributed attack detection scheme using deep learning approach for Internet of Things," *Future Generation Computer Systems*, vol. 82, pp. 761–768, 2018. https://doi.org/10.1016/j.future.2017.08.043

[34] P. Viola and M. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004. https://doi.org/10.1023/B:VISI.0000013087.49260.fb

[35] W. Hu, W. Hu, and S. Maybank, "AdaBoost-based algorithm for network intrusion detection," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 2, pp. 577–583, 2008. https://doi.org/10.1109/TSMCB.2007.914695