SvedbergOpen
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

Cuckoo Search-Based Compact Feature Optimization for IoT Intrusion Detection: An Experimental Study on the IoT-23 Dataset

Rintu Das[1], Vaskar Deka[2]

[1]Department of Information Technology, Gauhati University, Assam, India. rintu.das@gmail.com
[2]Department of Information Technology, Gauhati University, Assam, India. vaskardeka@gauhati.ac.in

Abstract:
The increasing scale and complexity of IoT traffic poses a challenge to accurate and computationally efficient intrusion detection, especially in the presence of redundant features and class imbalance. In this study, a machine learning framework for the IoT-23 dataset is presented which includes preprocessing of data, Synthetic Minority Oversampling Technique (SMOTE)/Random Under-Sampling, recursive feature elimination (RFE)–principal component analysis (PCA) dimensionality reduction, and binary Cuckoo Search (CS) feature selection. CS enables to optimize the feature selection by using a fitness function which balances the classification accuracy and feature compactness. The method chooses always the first and second principal components (PC1 and PC2) as the compressed representation. The K-nearest neighbors (KNN) classifier had the highest test accuracy, achieving 99.35%, followed by Random Forest (99.78%) and Decision Tree (99.76%) while bootstrap validation reports KNN accuracy of 99.35 ± 0.03%, leave-one-attack-out (LOAO) evaluation yields 0% accuracy for unseen attacks, thus illustrating the limitation of conventional supervised learning. A Prototypical Network reaches a mean validation accuracy of 90.37 ±10.48%. Finally, the results show the efficacy of CS in compact known-attack classification and the need for few-shot, open-set or zero-shot methods for emerging threats.

Keywords: IoT intrusion detection; IoT-23; Cuckoo Search; binary feature selection; RFE; PCA; SMOTE; Random Under-Sampling; KNN; machine learning; few-shot learning.

1. Introduction

1.1 Background

The Internet of Things (IoT) environments continuously generate streams of network activity from heterogeneous devices, protocols, services and communication patterns [1]. This traffic can be legitimate but also can be reconnaissance, denial-of-service, command-and-control, malware related and other malicious behaviors. An intrusion detection system (IDS) must therefore discriminate between multiple traffic categories while working on data that may contain redundant, correlated, categorical, numerical and address related attributes. The problem becomes more challenging when the attack classes are heavily imbalanced, as the dominant classes can dominate the learning objective and the minority attack types may not be sufficiently represented [2].

1.2 Problem Statement

The high-dimensional network data may cause higher computational cost and bring in redundant information which is not useful for classification [3]. In contrast, aggressive dimensionality reduction may lose information necessary to differentiate between minority or closely related attacks. The problem that the presented implementation solves is therefore a constrained feature selection problem: to find a small subset of transformed features that keeps the classification accuracy high while reducing the dimensionality of the input representation.
The implementation therefore provides a fitness function, which explicitly formulates this trade-off. A feature mask selects or rejects individual transformed features; classification accuracy is estimated through stratified cross-validation; and a penalty is applied according to the fraction of selected features. This makes the optimization objective different from simply maximizing accuracy: a candidate solution is preferred when it provides high predictive performance with fewer features.

1.3 Research Objective

The main objective is to analyze if binary Cuckoo Search is able to find a compact and discriminative representation of the IoT network traffic after an RFE-PCA preprocessing stage. Secondary goals are to

compare several machine-learning classifiers on the chosen representation, to evaluate the stability of feature selection, to analyze robustness through bootstrap and noise experiments, and to explore the ability of conventional and few-shot learning models to deal with attack categories not represented during training.

1.4 Contributions

- A complete preprocessing and class-balancing workflow for heterogeneous IoT-23 network traffic.
- A binary Cuckoo Search formulation that jointly considers predictive accuracy and feature compactness.
- Integration of Cuckoo Search with an RFE–PCA reduced representation rather than searching directly over the original heterogeneous feature space.
- Comparative evaluation of K-nearest neighbors (KNN), Random Forest, XGBoost, linear support vector machine (SVM), radial basis function support vector machine (RBF SVM), Decision Tree, and Naive Bayes, with additional ensemble models in the implementation.
- Stability and robustness analysis using multiple Cuckoo Search runs, bootstrap validation, Gaussian-noise perturbation, and Leave-One-Attack-Out evaluation.
- An exploratory Prototypical Network experiment to examine few-shot classification as a response to the zero-shot limitation of conventional supervised models.

1.5 Research Questions

- Can Cuckoo Search identify a substantially smaller feature representation while maintaining high classification accuracy?
- Which conventional classifier performs best on the Cuckoo Search-selected representation?
- How stable is the selected feature subset across different random seeds?
- How robust is the best-performing classifier to resampling and feature perturbation?
- Can conventional supervised learning generalize to an attack category that is completely absent from training?
- Does a Prototypical Network provide a useful alternative for few-shot attack classification?

2. Materials and Methods

2.1 Overall Experimental Pipeline

Through the implementation of the workflow, the sequential pipeline is represented as follows:
Internet of Things (IoT)-23 data acquisition → EDA (exploratory data analysis) → type conversion and missing-value treatment → categorical encoding and IP transformation → feature selection from the network-flow variables → class balancing → stratified train / test division → standardization → representation transformed through recursive feature elimination (RFE) and principal component analysis (PCA) → binary Cuckoo Search → compact feature subset → classifier training and evaluation → robustness and generalization analysis.

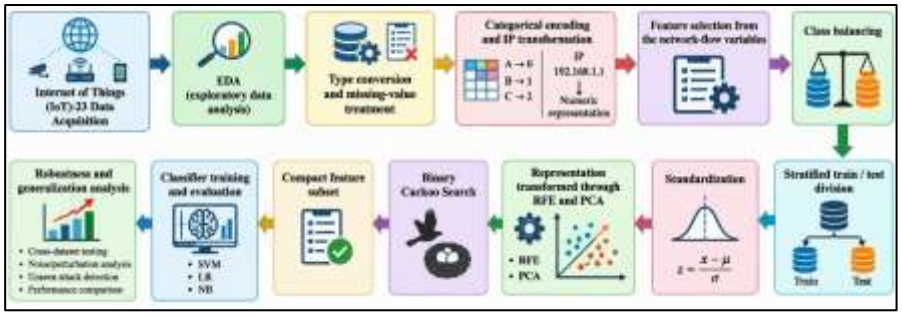

Figure 1(a): Overall Experimental Pipeline

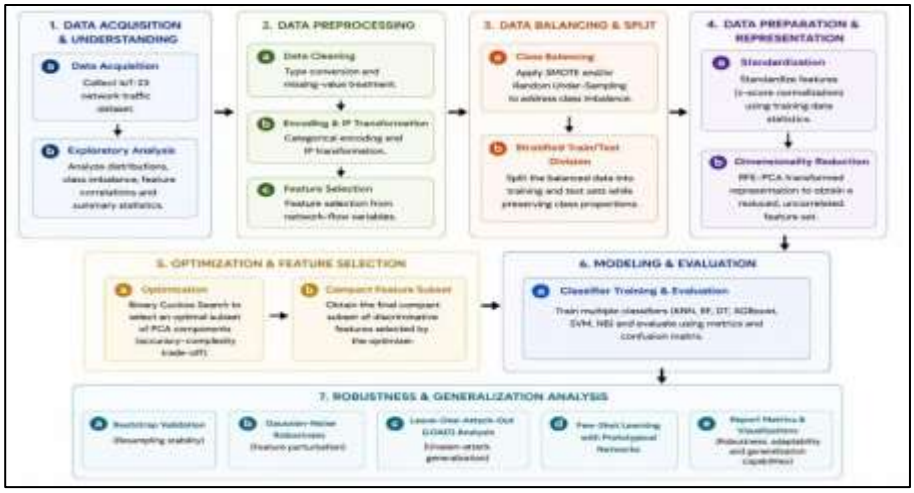

Figure 1(b) : Overall Experimental Pipeline

2.2 Dataset Preparation

The experiment uses the combined IoT-23 [4] dataset stored as a comma separated value (CSV) file. The full dataset sampling is performed at a ratio of 1.00, i.e. 100% sample, with a fixed random state of 42. The data is written to Parquet and then read back from the Parquet representation. This conversion is intended to allow for repeated processing of the large data set while maintaining deterministic sampling. The implementation also stores shape, head, data types, descriptive statistics and frequency distributions for labels, protocols and connection states.

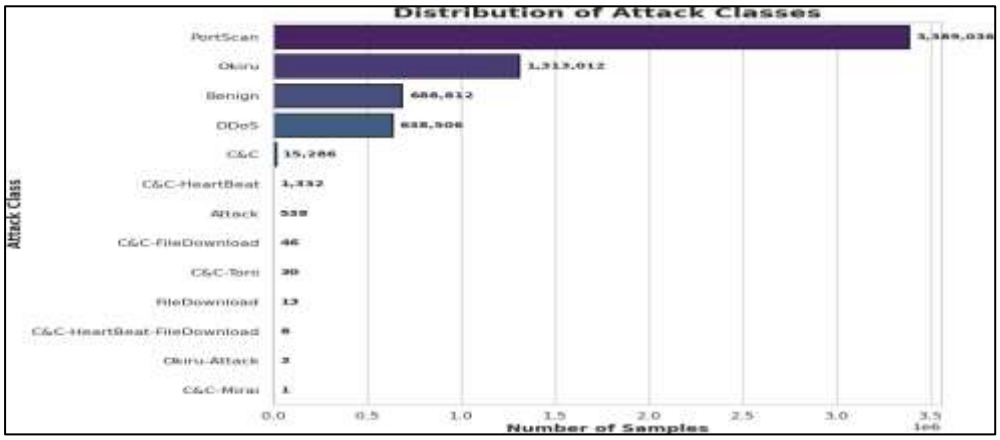

Figure 2: Attack Class Distribution

2.3 Data Cleaning and Numerical Representation

This stage includes the removal of columns with all equal values or nulls, which hinder the convergence of the algorithm. Missing numerical values, such as duration and byte-count fields, were replaced by zero to keep all network-flow records and to have a consistent numerical representation. LabelEncoder is employed to encode six categorical variables: protocol, service, connection state, local origin, local response and history. IP addresses are converted to numbers; IPv4 addresses are converted via padded octets and IPv6 addresses are reconstructed into eight hexadecimal groups before being converted to an integer. If an address cannot be translated, a bounded hash fallback is used. Raw data features include timestamp, source and destination address and port, protocol, service, duration, byte count, connection state, local flags, missed byte, history, packet count, and ip byte count. Exclude unique identifiers, such as the UID and the unnamed index The first set of network-flow features was chosen for their relevance to network communication characteristics, such as flow duration, packet/byte statistics, traffic rates and connection-related properties. At this stage features that were identifiers or redundant meta data or not suitable for numerical modeling were excluded. The resulting flow-level attributes were then subjected to RFE and PCA for systematic dimensionality reduction and Binary Cuckoo Search for compact subset selection.

2.4 Exploratory Statistical Analysis

The implementation does preliminary descriptive and correlation analysis before the modeling. A check for negative values in the numerical columns is performed. A Pearson type correlation matrix is generated based on the numeric representation. The resulting heatmap [5] is used to inspect possible redundancy and association among the traffic variables. The workflow generates class-distribution plots to show the imbalance that the subsequent balancing procedure addresses.

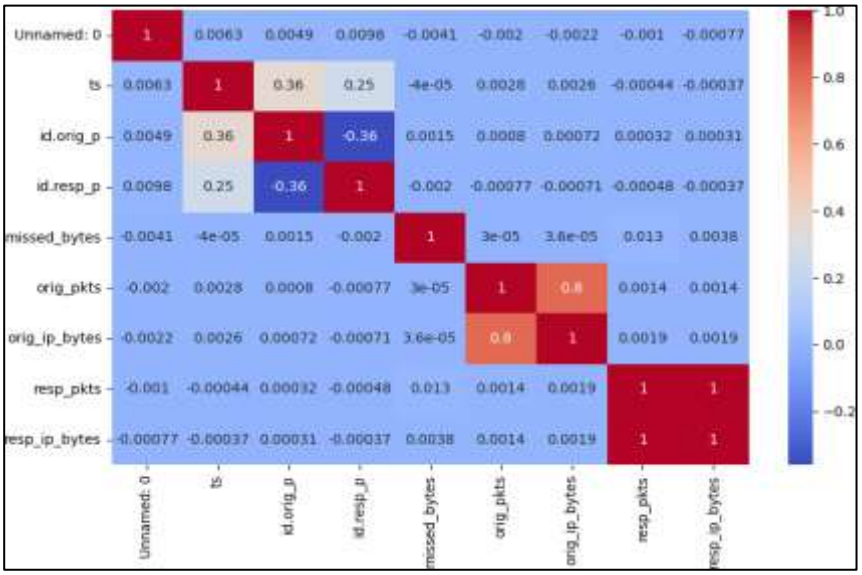

Figure 3: Correlation Heatmap

2.5 Class Balancing

We address class imbalance using a hybrid oversampling and undersampling strategy [6]. First, the implementation calculates the set of classes to be resampled according to the number of valid classes (at least two samples are needed). Then it dynamically computes a target sample size as the maximum of 8,000 and 0.8% of the size of the largest class [7]. These values were experimentally chosen to avoid overrepresenting the minority class, while still maintaining a sufficient representation of the less common attack classes. Classes below this target are allocated a SMOTE [7] over-sampling strategy while classes above the target are allocated a Random Under-Sampling. SMOTE is configured with one nearest neighbor to accommodate very small minority classes. The two operations are assembled in an imbalanced-learn pipeline and applied to the feature matrix and label vector, which provides a consistent mechanism for the subsequent resampling procedure.

Given the computational implications of class-wise resampling, this hybrid approach is important because simply oversampling[8] every class to the size of the dominant class could result in an extremely large dataset. Conversely, aggressive undersampling could discard valuable observations from common attack or benign classes. The dynamic target therefore attempts to create a compromise between minority-class representation [9]and computational feasibility. In the context of this implementation, the strategy is not intended to reproduce the original class distribution; it deliberately modifies the distribution to create a more balanced learning problem.

2.6 Train / Test Partition and Scaling

Following balancing, the data are split into training and testing partitions using an 80:20 stratified split and random state 42. StandardScaler is fitted to the training data and subsequently applied to the test data. This preserves a separation between training-derived scaling parameters and the held-out test representation. The processed arrays and scaler are saved for later experiments.

2.7 Recursive Feature Elimination–Principal Component Analysis (RFE–PCA) Representation

The Cuckoo Search stage does not operate directly on the original heterogeneous feature set. Instead, the implementation loads previously generated RFE–PCA transformed training and test arrays from the experiment directory. The code notes that the Cuckoo Search is performed on the RFE–PCA[10] transformed representation, which contains four features before Cuckoo Search. Through the reduction of the search space, this design allows the binary optimizer to concentrate on the selection of the most useful principal components, which improves the efficiency of the search.

Through the transformation of the original variables into principal components, the components should not be interpreted as individual physical network features, but rather as summaries of directions of variation in the preprocessed traffic space This distinction becomes important in the interpretation of

feature importance: the Decision Tree importance analysis in this work reports importance for components generated through principal component analysis[11] (PCA) rather than for the original raw variables.

2.8 Binary Cuckoo Search Feature Selection

Cuckoo Search is implemented as a population-based binary optimization procedure [12]. Each nest is a binary vector of length equal to the number of transformed features. A value of one means that the corresponding feature is retained, whereas zero means that it is excluded. Initial nests are generated randomly with a selection probability of 0.5. New candidate solutions are generated through a simplified Lévy-flight-inspired bit-flip operation, in which a fraction of the bits is inverted. A greedy replacement rule retains a new candidate only when its fitness exceeds the current nest fitness. A fraction of the worst nests is then abandoned and randomly reinitialized. The implemented fitness function is $F(M) = w1 \times Acc(M) - w2 \times |M|/d$, where M is the binary feature mask, Acc(M) is the mean three-fold cross-validation accuracy obtained using the selected features, |M| is the number of selected features, d is the total number of available transformed features, w1 = 0.9, and w2 = 0.1[13]. A solution selecting no features is assigned zero fitness, which ensures that the objective rewards accurate solutions while discouraging unnecessarily large subsets.

2.9 Cuckoo Search Configuration

Parameter	Implemented value
Number of nests	8
Iterations	7
Abandonment probability (pa)	0.30
Accuracy weight (w1)	0.90
Feature penalty weight (w2)	0.10
Cross-validation folds	3
Random state	42
Cuckoo Search (CS) classifier	Random Forest (50 trees, max_depth=10)

Table 1 : Cuckoo Search configuration used in the uploaded implementation.

2.10 Classification Models

After feature selection, the selected components are evaluated using multiple classifiers, which enables comparative model assessment. The Random Forest [14]baseline contains 100 trees; the Cuckoo Search optimizer uses a lighter 50-tree Random Forest with maximum depth 10. XGBoost is configured with 200 estimators, depth 6, learning rate 0.1, subsample and column-sampling rates of 0.8, histogram-based tree construction, and multiclass softmax output. Linear support vector machine[13] (SVM) is implemented through LinearSVC, while radial basis function (RBF) SVM uses an SVC pipeline with an RBF kernel, C=1.0, and probability estimates. Decision Tree uses maximum depth 20 and minimum samples split of 10. k-nearest neighbors (KNN) uses five neighbors, and Gaussian Naive Bayes provides a probabilistic baseline. [15]

2.11 Model Evaluation

Through three-fold stratified cross-validation, models are compared on the Cuckoo Search-selected features. Independent test evaluation is then conducted using the held-out RFE–PCA test representation with the same Cuckoo Search mask. Accuracy is computed directly, while classification reports and confusion matrices are generated to examine class-level behavior. Cohen's kappa is additionally calculated to quantify agreement beyond chance. [16]

2.12 Robustness, Stability, and Generalization

The implementation complements ordinary accuracy evaluation with a number of further experiments. The Cuckoo Search algorithm is run for a number of random seeds and the best accuracy and number of selected features are recorded[17]. Feature-selection frequency is calculated across runs to measure the consistency of each principal component [18]. The Cuckoo Search accuracy values are tested with a one-sample Wilcoxon signed-rank with a null hypothesis median of 0.99. [19] The best performing k-nearest neighbors (KNN) model [15] is further assessed by bootstrap validation, where resampled training data are used to retrain the model and predictions are evaluated on the original held-out test set. The implementation records the mean, standard deviation, median, minimum, and maximum bootstrap accuracy [20] and calculates a 95% confidence interval of the mean using the t-distribution and a percentile-based interval.

To study the sensitivity to perturbation, Gaussian noise with standard deviations of 0.00, 0.01, 0.05, 0.10, 0.20, 0.30, 0.40 and 0.50 are added to the selected test features. Accuracy is recorded at each noise level, allowing us to report a decreasing trend in KNN accuracy with increasing noise. The training and test representations are mixed using Leave-One-Attack-Out validation [21], where one attack category is left out in the model training. This puts zero-shot generalization to a stringent test since the withheld class is not present at all in the training set. Then, the implementation adds a Prototypical Network as a few-shot alternative [22]. A dynamic generation of episodes is done from the available classes.

3. Results

3.1 Conventional Classifier Performance

The selected features by Cuckoo Search obtained a good classification performance, which proves the effectiveness of the feature-selection methodology. The highest cross validation test accuracy was achieved by k-nearest neighbors (KNN) with 99.78%, followed by Random Forest (99.78%), Decision Tree (99.65%), and XGBoost (98.70%). Naive Bayes and linear support vector machines (SVMs) had relatively lower classification performance.

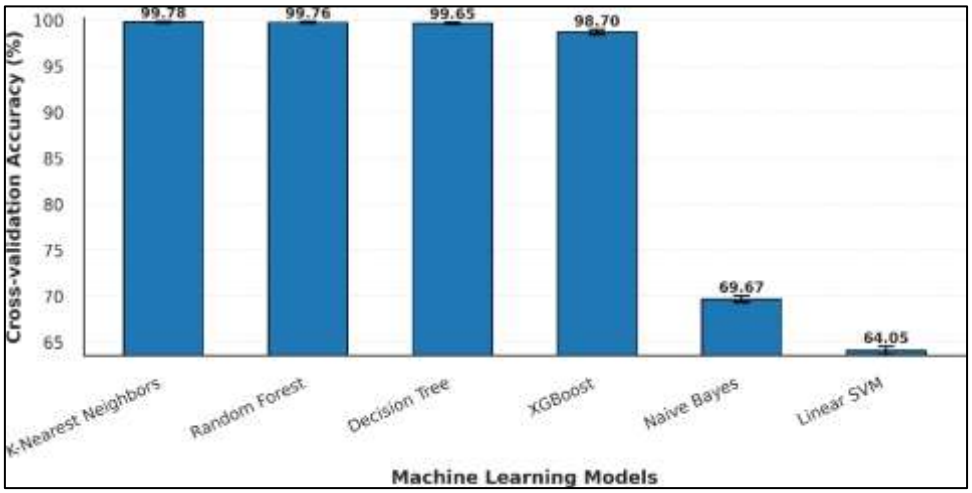

Figure 4 : Test accuracy comparison.

#	Model	Accuracy	Precision	Recall	F1-Score	Sensitivity	Specificity	Hamming Loss	(MCC)	Training Time (s)	Prediction Time (s)
0	Random Forest	0.997829	0.997834	0.997829	0.997828	0.997830	0.999566	0.002171	0.997397	0.636617	0.045600
1	KNN	0.997829	0.997830	0.997829	0.997828	0.997830	0.999566	0.002171	0.997396	0.013555	0.026705
2	Decision Tree	0.997623	0.997623	0.997623	0.997622	0.997623	0.999525	0.002377	0.997148	0.078996	0.000980
3	XGBoost	0.988424	0.988431	0.988424	0.988402	0.988424	0.997685	0.011576	0.986119	1.244552	0.051517
4	Naive Bayes	0.703359	0.713376	0.703359	0.671495	0.703383	0.940674	0.296641	0.656771	0.006297	0.003096
5	Linear SVM	0.641964	0.618249	0.641964	0.577089	0.641886	0.928390	0.358036	0.596227	2.959274	0.002109

Table 2 : Comparative performance of classifiers

#	Model	Macro Avg TPR (Recall)	Macro Avg FNR	Macro Avg TNR (Specificity)	Macro Avg FPR
0	Random Forest	0.997830	0.002170	0.999566	0.000434
1	XGBoost	0.988424	0.011576	0.997685	0.002315
2	Linear SVM	0.641886	0.358114	0.928390	0.071610
3	Decision Tree	0.997623	0.002377	0.999525	0.000475

4	K-Nearest Neighbors	0.997830	0.002170	0.999566	0.000434
5	Naive Bayes	0.703383	0.296617	0.940674	0.059326

Table 3 : Comparative performance of TPR,FNR, TNR, FPR.

3.2 Compact Feature Selection

PC1 and PC2 were identified by repeated runs of Cuckoo Search which suggests that these two components can form a compact and discriminative representation. Both parts contributed to classification performance.

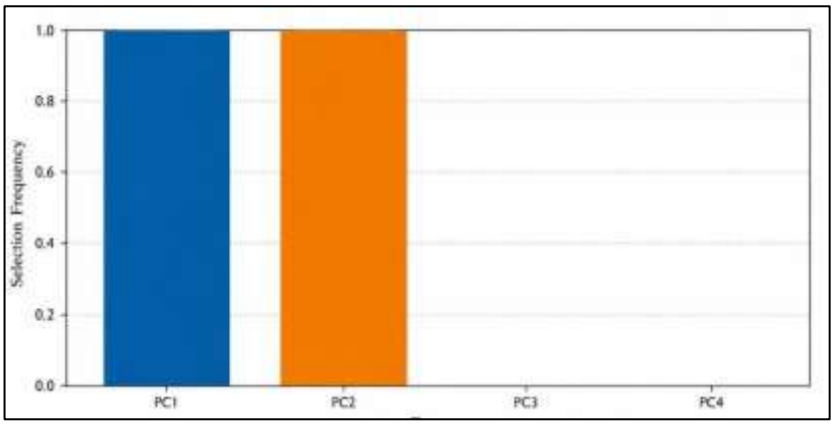

Figure 5 : Cuckoo Search feature-selection frequency.

3.3 Cuckoo Search Stability

The feature-selection procedure was shown to be stable in that PC1 and PC2 selection was consistent over ten independent runs. The statistical significance of the achieved accuracies was also assessed using a Wilcoxon signed-rank test.

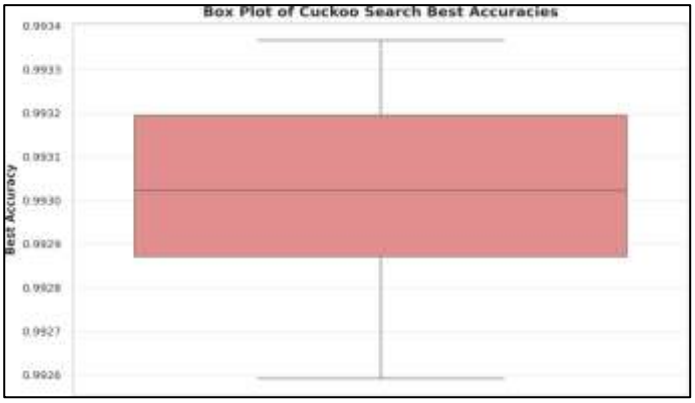

Figure 6 : Distribution / box plot of CS accuracy.

3.4 Robustness to Resampling

The mean accuracy of KNN was 99.35 ± 0.03% through bootstrap validation, indicating the stability of the performance under resampling and the proximity to the test and cross validation results.

#	Model	Mean Bootstrap	Standard	Median	Min Bootstrap	Max Bootstrap
0	Random Forest	0.9927	0.0009	0.9925	0.9902	0.9943
1	XGBoost	0.9797	0.0015	0.9796	0.9761	0.9820
2	Linear SVM	0.6324	0.0263	0.6377	0.5726	0.6864
3	Decision Tree	0.9902	0.0012	0.9902	0.9873	0.9929
4	K-Nearest	0.9935	0.0008	0.9935	0.9918	0.9951
5	Naive Bayes	0.7165	0.0232	0.7040	0.6944	0.7820

Table 4 : Bootstrap validation results

3.5 Robustness to Gaussian Noise

The accuracy of k-nearest neighbors (KNN) decreased progressively with increasing Gaussian noise, which showed some sensitivity to perturbations in features.

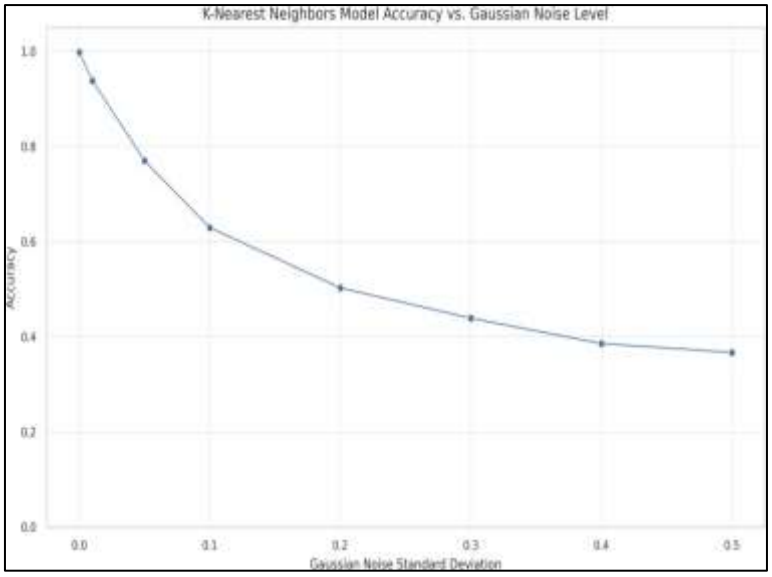

Figure 7 : KNN accuracy versus Gaussian noise level.

3.6 Known- and Unseen-Attack Generalization

Although standard classifiers performed well on known attacks, the LOAO experiment achieved 0% accuracy for an attack category never seen before, which shows the limitation of closed-set supervised learning [23].

3.7 Few-Shot Learning

The Prototypical Network (hidden_dim = 128, learning_rate = 0.005) achieved 90.83% ± 10.48% mean validation accuracy over 100 episodes. The higher variability shows that there is room for improvement, which supports further optimization.

#	Model Configuration	Mean Accuracy	Standard Deviation
0	Basic ProtoNet (2 Features)	0.8920	0.0913
1	Optimized ProtoNet (2 Features)	0.9083	0.1004
2	Advanced ProtoNet (2 Features)	0.8127	0.1268
3	Advanced ProtoNet (4 Features)	0.8843	0.0967

Table 5: Few-Shot Learning Result

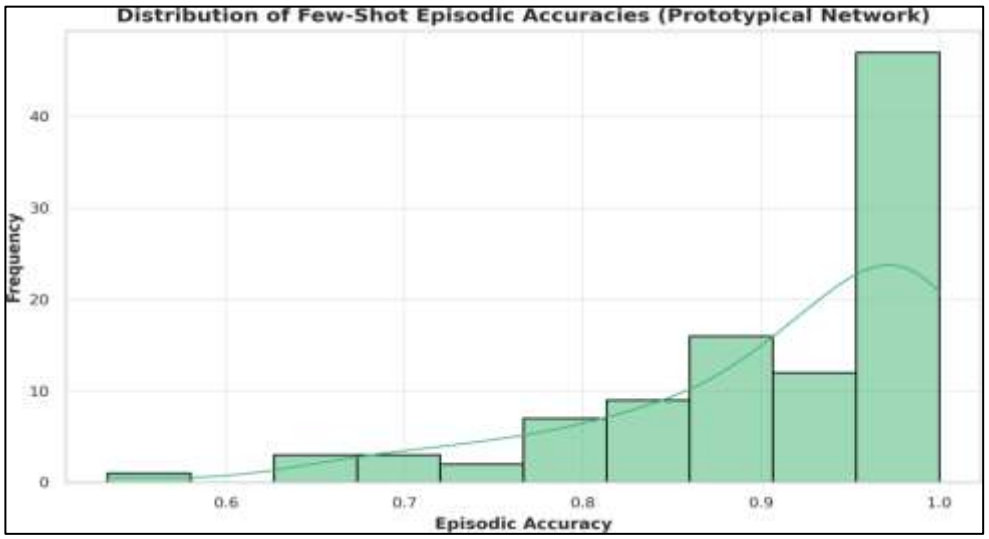

Figure 8: Few-shot episodic accuracy

3.8 Matthews Correlation Coefficient (MCC)

The analysis of the Matthews Correlation Coefficient (MCC)[24] shows that the results show good predictive

performance for the tree-based and distance-based classifiers. The Random Forest and K-Nearest Neighbors (KNN) achieved the highest MCC of 0.9974, followed by Decision Tree (0.9971) and XGBoost (0.9861). Naive Bayes and Linear Support Vector Machine (SVM) obtained relatively lower MCC values of 0.6568 and 0.5962 respectively. The results show that Random Forest, KNN and Decision Tree have very reliable classification performance with a good balance of right and wrong predictions.

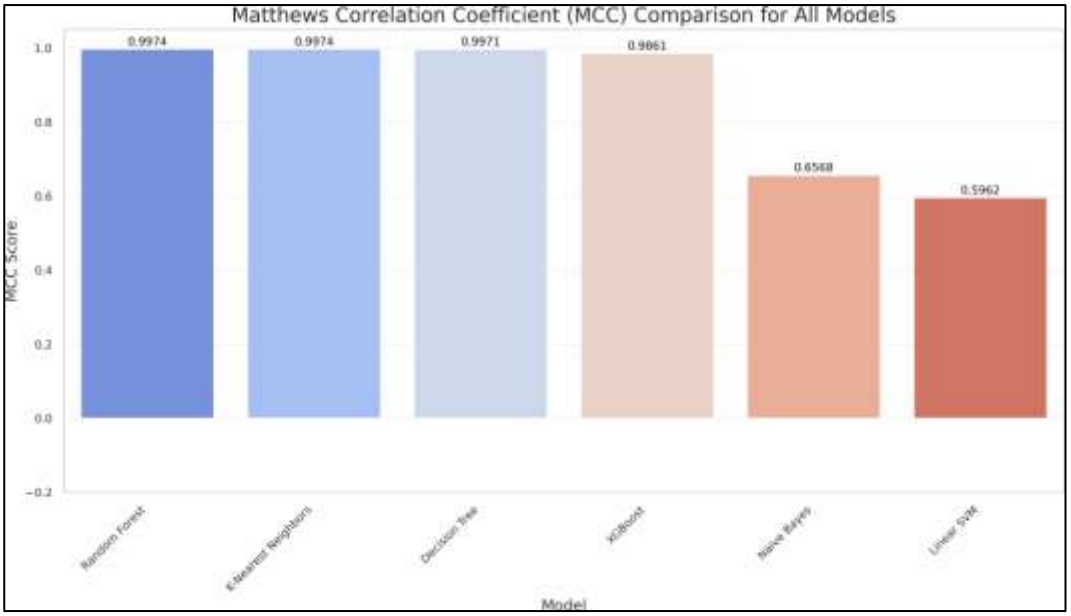

Figure 9: MCC comparison of all models

3.9 Receiver Operating Characteristic (ROC) Analysis

The receiver operating characteristic (ROC) curves[25] analysis shows that the proposed feature representation has good discriminative ability. The class discrimination was nearly perfect for Random Forest, XGBoost, Decision Tree and KNN with area under the curve (AUC) of approximately 1.00. The Naive Bayes[26] achieved an AUC of 0.92 and the Linear SVM obtained 0.86. All models significantly outperformed the random-guess baseline (AUC = 0.50).

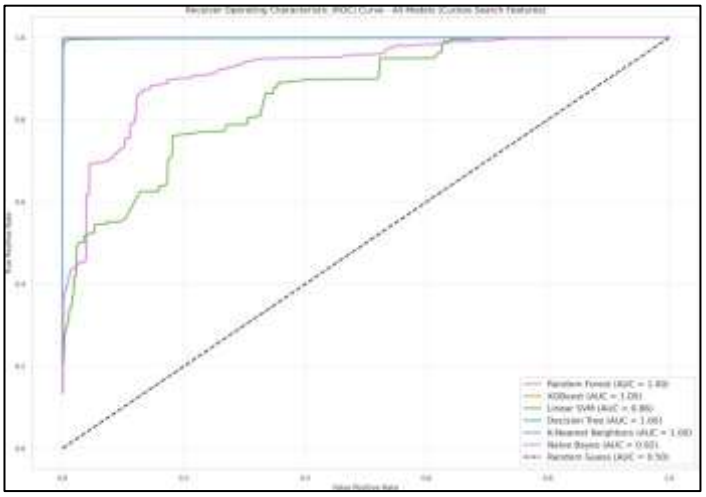

Figure 10: ROC curves of all models

The AUC values of the best classifiers are close to 1, which is consistent with their high MCC and classification accuracy, supporting the strong ability of the best classifiers to distinguish among the evaluated IoT attack classes.

3.10 Additional Model Performance Results

The analysis results of the F1-score also indicate the superior performance of the proposed Cuckoo Search selected representation. Random Forest and KNN became top F1-score of 0.9978, Decision Tree (0.9976) and XGBoost (0.9884) were close to the top. The F1-scores were much lower for Naive Bayes and Linear SVM, 0.6715 and 0.5771, respectively.

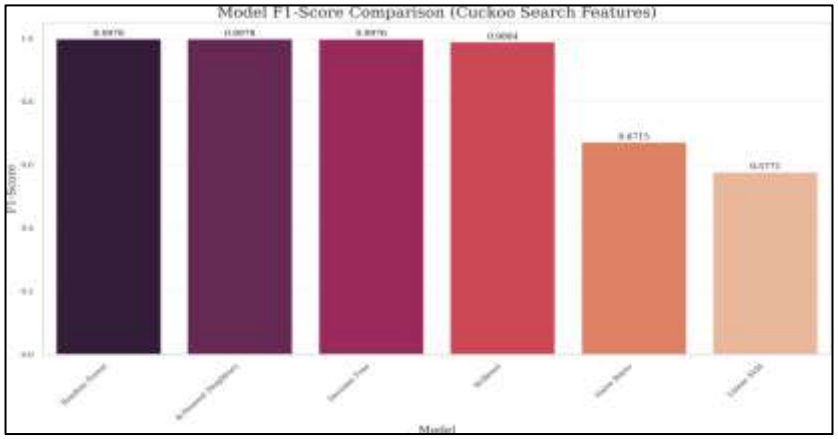

Figure 11: F1 Score Comparison

Joint assessment of accuracy, recall, precision and F1-score shows a stable pattern of performance, which is the basis for model distinction. The values for Random Forest, k-nearest neighbors (KNN) and Decision Tree are close to 1.0 for all the four metrics. In contrast, Naive Bayes and linear support vector machine (SVM) have significantly poor performance.

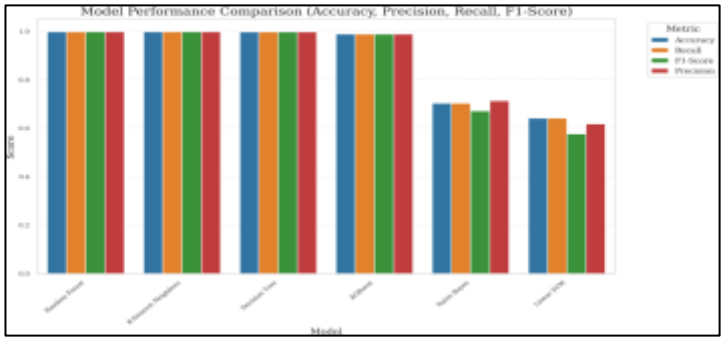

Figure 12: Accuracy, precision, recall, and F1-score comparison

These results are supported by findings based on hamming-loss [16]. The lowest Hamming loss was obtained by KNN and Random Forest (0.0022), followed by Decision Tree (0.0024) and XGBoost (0.0116). In contrast, Naive Bayes (0.2966) and Linear SVM (0.3580) have contributed significantly higher losses.

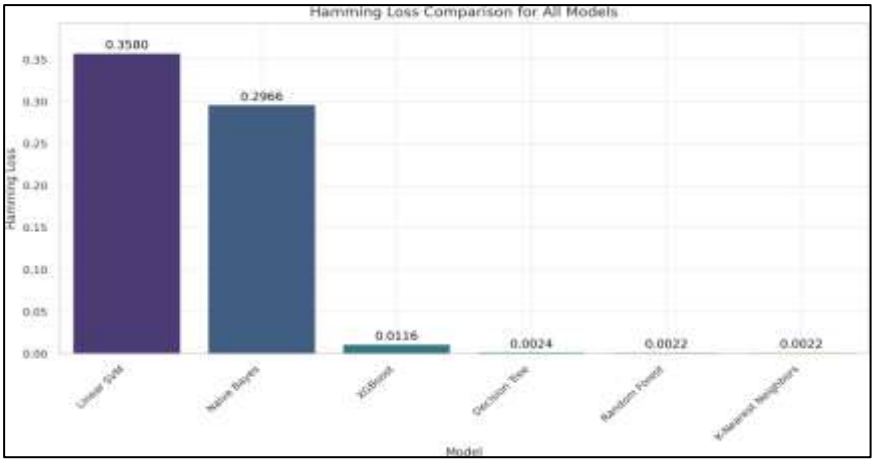

Figure 13: Hamming-loss comparison

From a computational analysis perspective, the linear support vector machine (SVM) was the slowest to train, followed by XGBoost and Random Forest based on training-time results. Decision Tree and KNN were trained in much less time. All models exhibited low prediction times overall, suggesting that the reduced feature representation enables efficient inference.

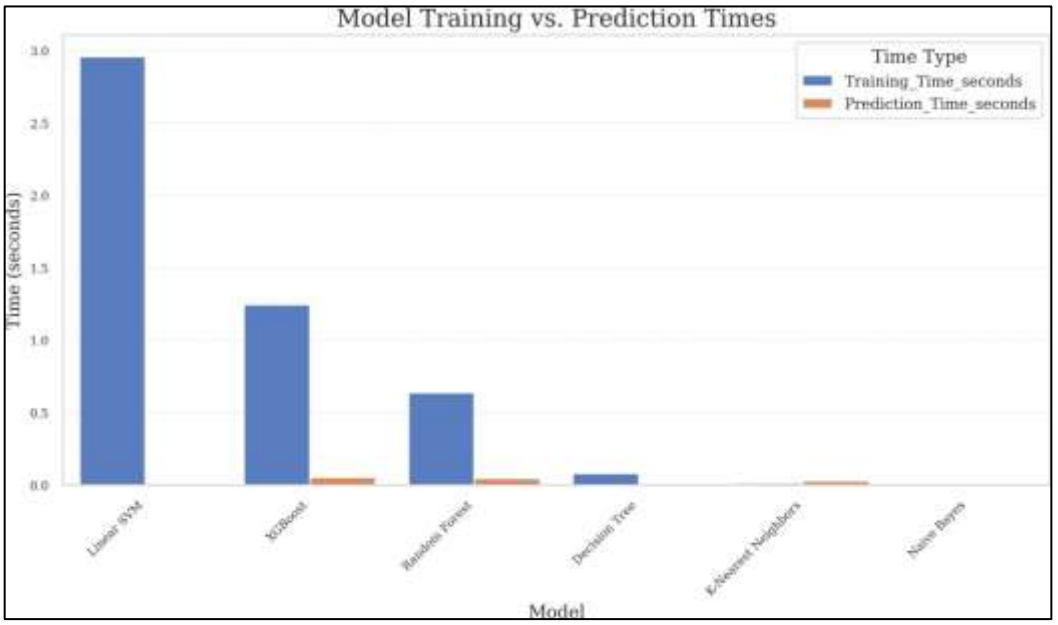

Figure 14 : Model training and prediction time comparison

The Cuckoo Search optimization showed relatively stable performance across random seeds. The best accuracy was achieved in a narrow band of approximately 0.9926-0.9934, with seed 51 doing best. Therefore, the random initialization did not matter much for the accuracy in the settings we tested.

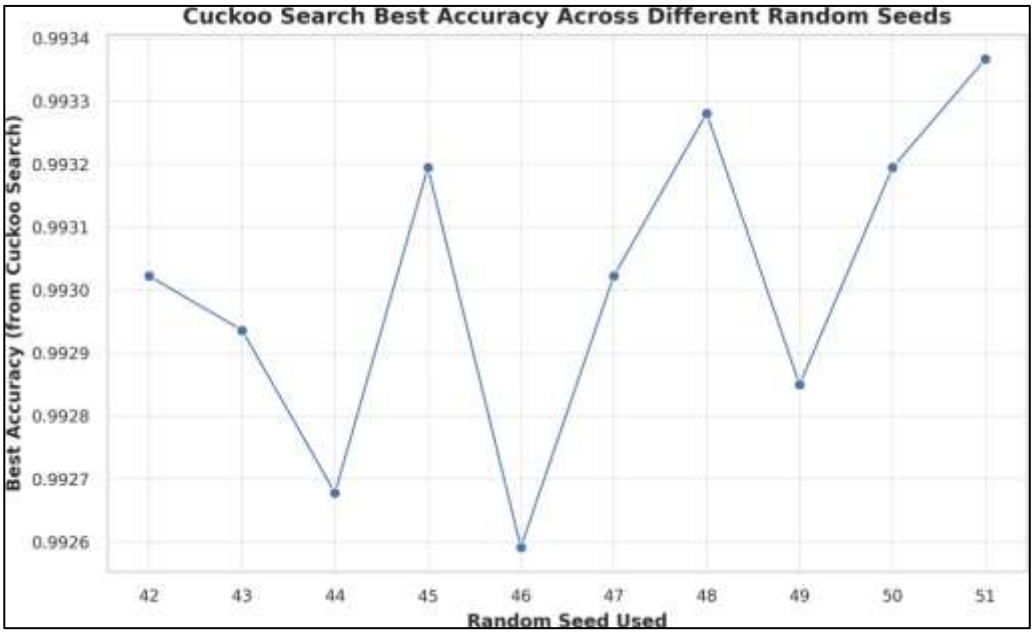

Figure 15: Cuckoo Search accuracy across random seeds

Validation of KNN with Bootstrap [15] showed a very tight distribution of accuracies around 0.9977 with most observations within a small range. This supports stability of the classifier to repeated resampling.

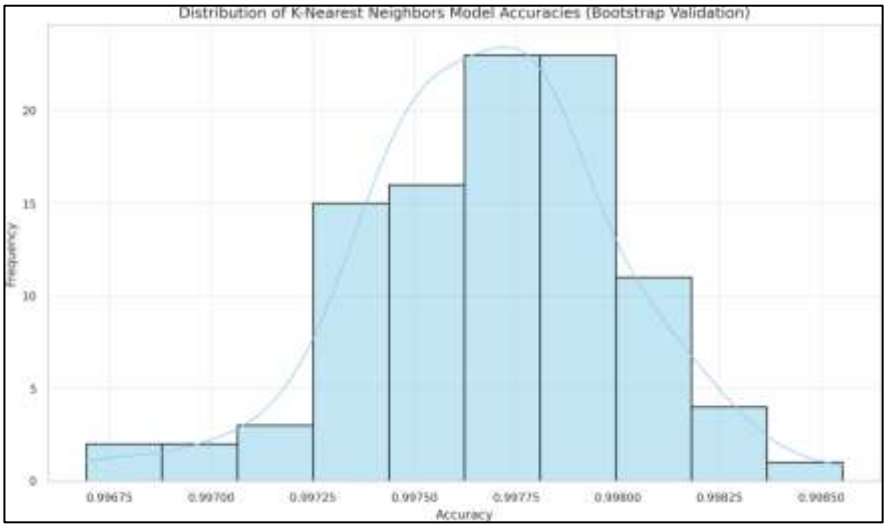

Figure 16: Bootstrap distribution of KNN accuracy

#	Model	Class	Precision	Recall (Sensitivity)	F1-Score	Specificity
0	K-Nearest Neighbors	Attack	0.9938	0.9994	0.9966	0.9988
1	Random Forest	Attack	0.9938	0.9981	0.9960	0.9988
2	Decision Tree	Attack	0.9944	0.9969	0.9957	0.9989
3	Random Forest	Benign	0.9963	0.9975	0.9969	0.9993
4	Decision Tree	Benign	0.9969	0.9957	0.9963	0.9994
5	K-Nearest Neighbors	Benign	0.9932	0.9957	0.9944	0.9986
6	Random Forest	C&C	0.9988	0.9913	0.9950	0.9998
7	Decision Tree	C&C	0.9956	0.9932	0.9944	0.9991
8	K-Nearest Neighbors	C&C	0.9975	0.9870	0.9922	0.9995
9	Random Forest	DDoS	1.0000	1.0000	1.0000	1.0000
10	Decision Tree	DDoS	1.0000	1.0000	1.0000	1.0000
11	K-Nearest Neighbors	DDoS	1.0000	1.0000	1.0000	1.0000
12	Random Forest	Macro Average	0.9978	0.9978	0.9978	0.9996
13	Decision Tree	Macro Average	0.9976	0.9976	0.9976	0.9995
14	K-Nearest Neighbors	Macro Average	0.9970	0.9970	0.9970	0.9994
15	Random Forest	Okiru	1.0000	1.0000	1.0000	1.0000
16	Decision Tree	Okiru	0.9994	1.0000	0.9997	0.9999
17	K-Nearest Neighbors	Okiru	0.9994	1.0000	0.9997	0.9999
18	Decision Tree	PartOf	0.9994	1.0000	0.9997	0.9999
19	Random Forest	PartOf	0.9981	1.0000	0.9991	0.9996
20	K-Nearest Neighbors	PartOf	0.9981	1.0000	0.9991	0.9996

Table 6: Performance of the top 3 overall models, broken down by each attack class

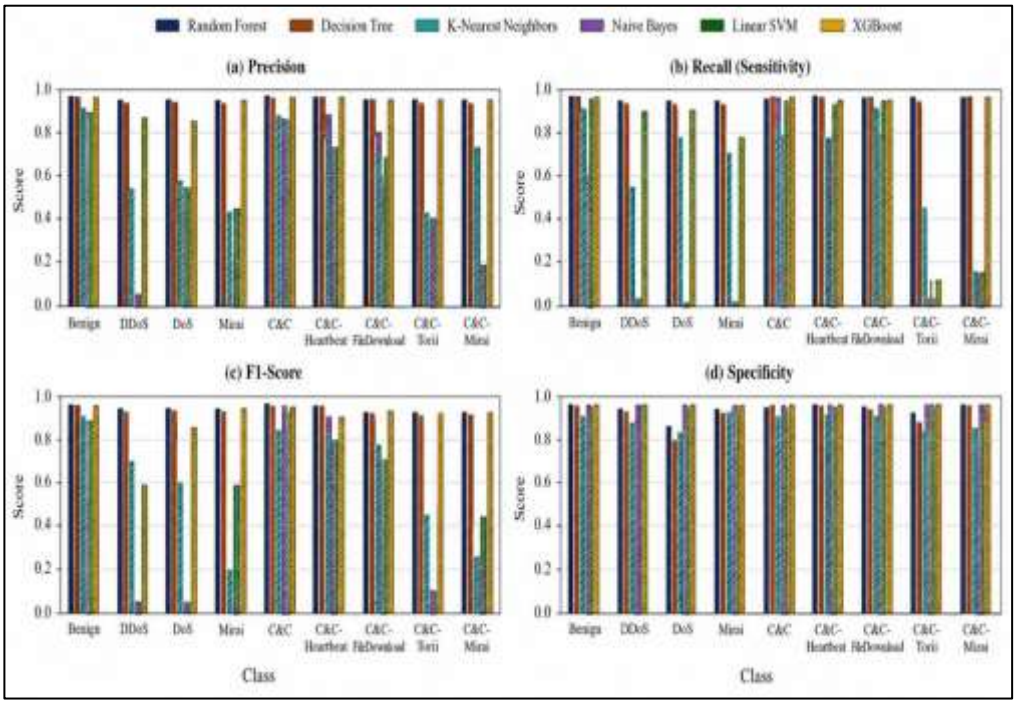

Figure 17: Class-wise group plot for all models and all metrics

Fig. 17 compares the per-class performance of six classifiers using the two-component representation. Random Forest, Decision Tree, and KNN generally achieve high precision, recall, F1-score, and specificity across most classes, while Naive Bayes and Linear SVM show reduced performance for several attack categories. The results demonstrate strong class-level discrimination for the tree-based and nearest-neighbor models, while also highlighting challenges in detecting certain low-frequency attack classes.

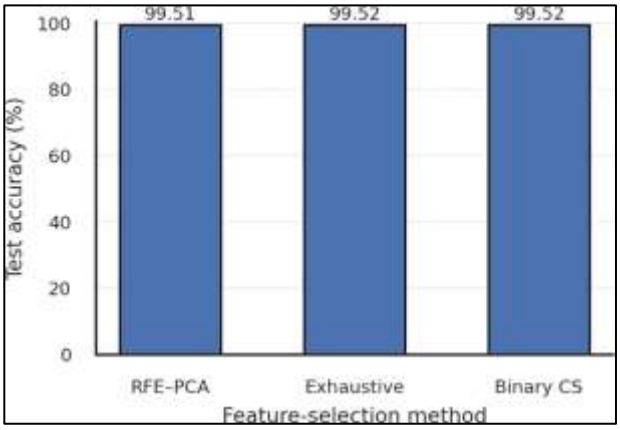

Figure 18: Ablation Study

The ablation study displays similar performance: RFE–PCA = 99.51%, Exhaustive Search and Binary Cuckoo Search = 99.52%. Binary CS can achieve the same accuracy as exhaustive search with a small feature subset. While the proposed selection mechanism is valid for the four-component search space, it does not provide a computational advantage over an exhaustive search as the Binary Cuckoo Search can reach the same compact solution with the same performance.

To quantitatively evaluate the computational advantage of the compact representation, the inference latency of both the four-component RFE–PCA representation and the proposed two-component representation was measured under the same experimental conditions. The average inference time was reported in microseconds per sample with its standard deviation and the corresponding throughput. The results offer a quantitative basis for evaluating the computational benefit of feature reduction, and its potential application in real-time IoT intrusion detection.

#	Representation	Mean Latency (µs/sample)	Std. (µs)	Throughput (samples/s)

0	RFE–PCA (4 features)	56.49	175.75	17703.81
1	RFE–PCA + Binary CS (2 features)	82.78	262.13	12080.68

Table 7: Inference Efficiency of Four- and Two-Component Representations.

4. Discussion

The suggested combination includes class balancing, RFE–PCA dimensionality reduction, and Cuckoo Search (CS) feature selection to provide a compact and efficient representation for IoT intrusion detection. CS still finds PC1 and PC2, which means these components are still rich in discriminative information, meanwhile reduce the feature space.

The comparative classifier results indicate a strong performance of the chosen representation with KNN, Random Forest and Decision Tree achieving the highest reported accuracies. This outcome implies that the reduced feature space still captures useful class-discriminative patterns. The poorer performance of Linear SVM and Naïve Bayes suggests that the transformed feature space might be more suitable for non-linear or neighborhood-based classifiers.

The framework demonstrates stable performance by bootstrap validation. The Gaussian-noise experiment demonstrates that the larger the feature perturbation, the worse the classification performance. These results show a good stability to resampling but some sensitivity to noisy inputs. The LOAO experiment shows that conventional supervised models achieve 0% accuracy on an unseen attack category, indicating that they are primarily closed-set classifiers and do not reliably detect unseen attack types [27]. The Prototypical Network is a possible few-shot alternative, but its performance is worse and more variable than KNN on known attacks. Overall, the results support CS as an effective compact feature selection strategy for known IoT attacks, but future work should include anomaly detection, open-set recognition or few-shot learning to improve detection of emerging and unseen threats.

5. Conclusion

This paper proposes an IoT intrusion detection framework with class balancing, RFE-PCA dimensionality reduction, binary Cuckoo Search feature optimization and comparative machine learning. The framework addresses an important tension in the design of IoT IDS: preserving sufficient discriminative information while reducing the dimensionality and computational burden of the traffic representation. From the results reported, it is shown that Cuckoo Search strategy can find a very compact representation of the principal components 1 and 2 (PC1 and PC2). Using this compact representation, k-nearest neighbors (KNN) achieve the best reported test accuracy of 99.84%, followed by Random Forest (99.78%) and Decision Tree (99.76%). XGBoost also performs well with 98.76% while Naive Bayes and linear support vector machine (SVM)performance is much worse. Hence the solution is validated, and the selected feature subset can be considered optimal. For such a small search space with only four components, an exhaustive enumeration would be the method of choice.

The robustness and stability analyzes support the main finding. Repeated experiments with Cuckoo Search consistently identify the same two major components and bootstrap validation of KNN yields a mean accuracy of 99.77 ± 0.03%. However, gaussian noise experiments show performance degrades with increasing perturbation and leave-one-attack-out (LOAO) validation shows a complete failure to classify an attack category with complete absence from training.

The Prototypical Network experiment is a first way to few-shot adaptation with mean validation accuracy 90.83 ± 10.48% on 100 episodes. This is not comparable to the conventional KNN accuracy on known classes, but it solves a different and more challenging problem. The combined results indicate that the most promising architecture for future IoT IDS research would likely be a hybrid: a compact, high accuracy supervised classifier for known threats coupled with anomaly detection and few-shot or zero-shot mechanisms for novel attacks.

In this study, we propose a compact feature selection framework based on Cuckoo Search for the RFE-PCA representation that reduces the representation to only two components, PC1 and PC2, but still achieves high classification performance. Furthermore, the novelty of this framework is also validated by the combination of conventional classification, robustness testing, MCC/F1/ROC-AUC evaluation, LOAO analysis for unseen attacks, and few-shot learning, which offer a comprehensive evaluation of performance and generalization.

Future work can include a comparison of the proposed Cuckoo Search method with other metaheuristic feature selection methods and an assessment of its generalization capabilities over temporal and cross-scenario distribution shifts. Further work is also needed to develop open-set and anomaly detection mechanisms for previously unseen attacks, and to improve few-shot adaptation to emerging attack classes using advanced meta-learning methods. In addition, the framework needs to be evaluated for computational

efficiency in terms of latency, memory requirements and throughput, to check if it is suitable for deployment in IoT edge environments.

References

[1] A. Khraisat and A. Alazab, "A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges," *Cybersecurity* , vol. 4, no. 1, Mar. 2021, doi: 10.1186/s42400-021-00077-7 .

[2] H. Jmila, G. Blanc, M. R. Shahid, and M. Lazrag, "A Survey of Smart Home IoT Device Classification Using Machine Learning-Based Network Traffic Analysis," *IEEE Access* , vol. 10, pp. 97117–97141, Jan. 2022, doi: 10.1109/access.2022.3205023.

[3] M. D. Mauro, G. Galatro, G. Fortino, and A. Liotta, "Supervised feature selection techniques in network intrusion detection: A critical review," *arXiv (Cornell University)* , vol. 101, p. 104216, Mar. 2021, doi: 10.1016/j.engappai.2021.104216.

[4] Q. Waseem, W. I. S. B. W. Din, A. B. A. Rahman, S. N. Khan, R. A. A. Busaeed, and T. Fairooz, "A survey and analysis of feature selection techniques in machine learning for IoT device classification within smart buildings," *Innovative Infrastructure Solutions* , vol. 10, no. 9, Aug. 2025, doi: 10.1007/s41062-025-02203-7.

[5] Liu, Y. ., M. . Cao, and C. . Cao. "A Comparative Study of Deep Learning Models for Malware Detection in IoT Networks: CNN, LSTM, and Hybrid Architectures". *Journal of Cyber Security and Mobility*, vol. 15, no. 04, Aug. 2026, pp. 823–866, doi:10.13052/jcsm2245-1439.1543.

[6] M. Khushi *et al.* , "A Comparative Performance Analysis of Data Resampling Methods on Imbalance Medical Data," *IEEE Access* , vol. 9, pp. 109960–109975, Jan. 2021, doi: 10.1109/access.2021.3102399.

[7] R. Blagus and L. Lusa, "SMOTE for high-dimensional class-imbalanced data," *BMC Bioinformatics* , vol. 14, no. 1, p. 106, Mar. 2013, doi: 10.1186/1471-2105-14-106.

[8] C. Vairetti, J. L. Assadi, and S. Maldonado, "Efficient hybrid oversampling and intelligent undersampling for imbalanced big data classification," *Expert Systems with Applications* , vol. 246, p. 123149, Jan. 2024, doi: 10.1016/j.eswa.2024.123149.

[9] L. Yang and A. Shami, "Toward Autonomous and Efficient Cybersecurity: A Multi-Objective AutoML-Based Intrusion Detection System," *IEEE Transactions on Machine Learning in Communications and Networking* , vol. 3, pp. 1244–1264, Jan. 2025, doi: 10.1109/tmlcn.2025.3631379.

[10] X. Yang and S. Deb, "Cuckoo search: recent advances and applications," *arXiv (Cornell University)* , vol. 24, no. 1, pp. 169–174, Mar. 2013, doi: 10.1007/s00521-013-1367-1.

[11] S. Seth, G. Singh, and K. K. Chahal, "A novel time efficient learning-based approach for smart intrusion detection system," *Journal Of Big Data* , vol. 8, no. 1, Aug. 2021, doi: 10.1186/s40537-021-00498-8.

[12] A. Gherboudj, A. Layeb, and S. Chıkhı, "Solving 0-1 knapsack problems by a discrete binary version of cuckoo search algorithm," *International Journal of Bio-Inspired Computation* , vol. 4, no. 4, p. 229, Jan. 2012, doi: 10.1504/ijbic.2012.048063.

[13] Bukhowah, Rawan Yousef, et al.. "Enhanced IoT Security Using Machine Learning Technology." International Journal of Advanced Computer Science and Applications, vol. 16, no. 9, 2025, https://doi.org/10.14569/IJACSA.2025.0160971.

[14] C. V. Oha et al., "Analysis of IoT-23 datasets and machine learning models for malicious traffic detection," ERA. Accessed: Jul. 16, 2025. Available at: https://era.library.ualberta.ca/items/28700e1c-3afc-4892-b12e-5c229df9e056.

[15] Ullah, S., Wu, J., Lin, Z. *et al.* Comparative analysis of deep learning and traditional methods for IoT botnet detection using a multi-model framework across diverse datasets. *Sci Rep* **15**, 31072 (2025). https://doi.org/10.1038/s41598-025-16553-w.

[16] Krishnan, D., Singh, S. & Sugumaran, V. Explainable AI for zero-day attack detection in IoT networks using attention fusion model. *Discov Internet Things* **5**, 83 (2025). https://doi.org/10.1007/s43926-025-00184-8.

[17] Diao, R., Shen, Q. Nature inspired feature selection meta-heuristics. *Artif Intell Rev* **44**, 311–340 (2015). https://doi.org/10.1007/s10462-015-9428-8.

[18] K. Thirugnanasambandam *et al.* , "Optimizing multimodal feature selection using binary reinforced cuckoo search algorithm for improved classification performance," *PeerJ Computer Science* , vol. 10, Jan. 2024, doi: 10.7717/peerj-cs.1816.

[19] Shao-Qiang Ye, Azlan Mohd Zain, Yusliza Yusoff, "Improved Cuckoo Search Algorithm for Engineering Optimization Problems", Computers, Materials and Continua, Volume 87, Issue 1, 2026, ISSN 1546-2218, https://doi.org/10.32604/cmc.2025.073411.

[20] Zaki, Rana & Naser, Inaam. (2024). Hybrid Classifier for Detecting Zero-Day Attacks on IoT Networks. Mesopotamian Journal of CyberSecurity. 4. 59-74. 10.58496/MJCS/2024/016.

[21] AL-Akhras, Mousa & Alawairdhi, Mohammed & Alkoudari, Ali & Atawneh, Samer. (2020). Using Machine Learning to Build a Classification Model for IoT Networks to Detect Attack Signatures.

International journal of Computer Networks & Communications. 12. 99-116. 10.5121/ijcnc.2020.12607.

[22] S. Jain, S. Gupta, A. Pundir, S. Singh and G. J. Saxena, "FS-ProtoNet: Enterprise Network Security Using Few-Shot Learning with Prototypical Networks," 2025 IEEE International Conference on Recent Advances in Computing and Systems (REACS), Gwalior, India, 2025, pp. 1-7, doi: 10.1109/REACS67479.2025.11413551.

[23] Sarhan, M., Layeghy, S., Gallagher, M. *et al.* From zero-shot machine learning to zero-day attack detection. *Int. J. Inf. Secur.* **22**, 947–959 (2023). https://doi.org/10.1007/s10207-023-00676-0.

[24] Iturbe-Araya, JI., Rifà-Pous, H. Hyperparameter Optimization and Evaluation Metrics for Unsupervised Anomaly-Based Cyberattack Detection in Imbalanced Smart Home Datasets. *J Netw Syst Manage* **33**, 99 (2025). https://doi.org/10.1007/s10922-025-09973-6.

[25] B. M. Kouassi, V. Monsan, A. B. Ballo, K. J. Ayikpa, D. Mamadou, and K. J. Adou, "Application of the Learning Set for the Detection of Jamming Attacks in 5G Mobile Networks," *International Journal of Advanced Computer Science and Applications* , vol. 14, no. 6, Jan. 2023, doi: 10.14569/ijacsa.2023.0140676.

[26] Md. A. Hossain, W. Ishtiaq, and Md. S. Islam, "A Comparative Analysis of Ensemble-Based Machine Learning Approaches With Explainable AI for Multi-Class Intrusion Detection in Drone Networks," *Security and Privacy* , vol. 9, no. 1, Dec. 2025, doi: 10.1002/spy2.70164.

[27] L. Chettri, J. Leo, S. Roy, and J. K. Kalita, "Analyzing shallow and deep classifiers in detecting unseen attacks on internet of things network," *Peer-to-Peer Networking and Applications* , vol. 19, no. 3, Mar. 2026, doi: 10.1007/s12083-026-02217-7.