DMPedia Lecture Notes in Computer Science & Engineering
Volume 1, pp. 309-322, 2026

International Conference on Multidisciplinary Perspectives in Advanced Computing and Technology (IMPACT 2026)
G. B. Pant University of Agriculture and Technology, Uttarakhand, India. Jan. 10-11, 2026

The IDS Framework for Recognizing Unusual Activity in IoT Networks

Vipin Poswal[1], Jorge M. Cortés-Mendoza[2], Yogendra Kumar[3]

[1]National College of Ireland, Dublin, Ireland

[2]National College of Ireland, Dublin, Ireland

[3]Bennett University, Greater Noida, India

vipin.poswal5@gmail.com, JorgeMario.CortesMendoza@ncirl.ie, yogendrak348@gmail.com

Abstract

Protecting against cyberattacks is becoming increasingly crucial as the Internet of Things (IoT) and its resources expand rapidly. The most recent sophisticated attacks leverage characteristics of ordinary traffic to launch novel, covert attacks, while more conventional perimeter-based defences (such as firewalls and authentication) struggle to identify them. The IoT Network Intrusion Dataset (IoTID20) is used to test a lightweight network intrusion detection system designed specifically for the Internet of Things. The approach is based on a workflow for Knowledge Discovery in Databases (KDD) that incorporates thorough data preprocessing. To optimise the trade-off between processing efficiency and efficacy, Artificial Neural Networks (ANNs), K-Nearest Neighbours (KNN), and Random Forests (RFs) are the classifier models created and evaluated. They can also be adjusted for edge implementation. The usual metrics mentioned in the literature are followed while implementing the performance measure. According to the experiment's findings, RF outperformed the state of the art by 1.2% in accuracy, achieving the best performance in real-time intrusion detection for IoT scenarios.

Keywords: *IoT Security, Random Forest, Artificial Neural Network, K-Nearest Neighbors, Dataset IoTID20*

1. Introduction

Through widespread connectivity and automation, the IoT is transforming the healthcare, manufacturing, transportation, and agricultural industries as it expands at a rate of over 41 billion devices by 2025 [1], Unfortunately, the same reasons that make adoption easy include a wide variety of hardware, limited CPU, memory and power, and open-by-design services, which create an even greater attack surface. According to the recent industry reports, attacks on the Internet of Things have increased by 37%, with ransomware being the most common type of attack, and the average cost of the attacks has soared as of 2024 to twice as much as that of conventional IT attacks [2], [3]. As shown in Fig. 1, there has been a swift increase in the number of such explosive devices.

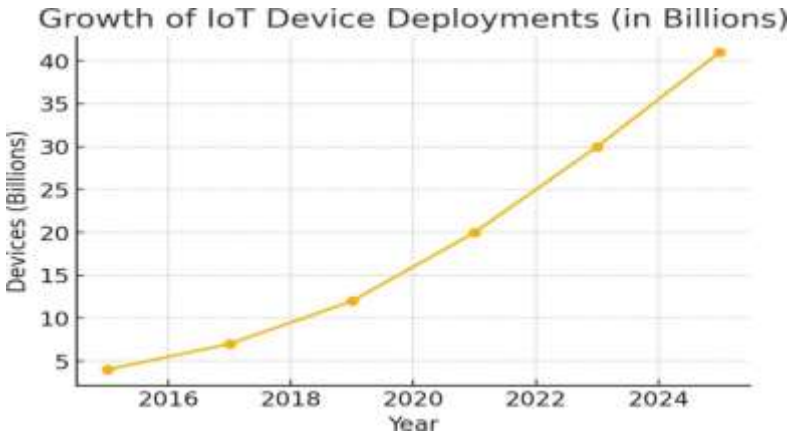


Fig. 1: IoT environment, 2015–2025: The expected number of IoT devices deployed increased from 7 billion in 2015 to 41 billion by 2025 [1].

Large volumes of continuously flowing data, such as financial transactions, industrial control signals, smart home updates, and personal conversations, are handled by IoT networks. To obtain confidential information, disrupt critical services, or plant malicious software, cybercriminals are drawn to this valuable data environment. Attackers constantly modify their strategies to incorporate malicious traffic into normal network flows to avoid detection, even when baseline protection is provided by common security mechanisms such as firewalls, proxy servers, authentication systems, and encryption [4], [5].

To detect and identify unusual and invasive traffic, Network Intrusion Detection Systems (NIDS) are essential. Nevertheless, there are efficiency gaps in contemporary NIDS systems that prevent their integration into IoT contexts [6], [7]. This study offers an innovative and effective NIDS framework that can operate effectively in IoT and wireless network scenarios with limited resources to address these drawbacks. High detection accuracy is attained by the suggested method, which also exhibits great promise for real world implementation in limited IoT environments [8].

There are two basic applications for NIDS: anomaly detection and signature-based detection. By using known attack signatures, signature-based NIDS can successfully identify known threats; however, they might not be able to identify zero-day/new attacks. Utilizing Machine Learning (ML), NIDS based on anomalies to define usual traffic behavior. The defined typical traffic is then used as a model against which deviations are identified as potentially dangerous. Although anomaly-based methods can identify new online threats, they often produce false positives unless properly tuned. Consequently, efficient, adaptable, and resource- saving NIDS are essential in the realization of high detection levels in the ever-evolving threat environment in the IoT.

The primary contributions consist of the subsequent research:

1) An IoT-NIDS with limited resources that can adapt to different IoT scenarios and is tested on an IoTID20 using an open, reproducible process and methodology.

2) A productive method to minimize features that results in achieving high accuracy and efficiency while ensuring no data leakage occurs.

3) A thorough evaluation using metrics like accuracy, precision, recall, F1-score, and ROC assessment.

4) Proof of improved efficiency compared to the top Internet of Things security systems already in use.

2. Research Methodology

The research is conducted in the Data Science methodology (KDD) that involves dataset selection, final evaluation, and interpretation. The IoTID20 originally downloaded was the IoT Dataset. During preprocessing, the original labels (DDoS, Mirai, and Recon) were to be divided into five groups: Mirai, MITM ARP Spoofing, DoS, Normal, and Scan. All other classes were changed to 83,135 samples each, but the Mirai class was reduced to 20% of its initial size. Data cleaning consisted of removing duplicate columns, handling FlowPkts/s as reliable, replacing infinite values in FlowByts/s and FlowPkts/s with the maximum of the corresponding columns, and refilling the remaining NaNs with the means of the features.

Pearson correlations for attributes were computed throughout the transforming process; one from each pair with a correlation greater than 0.90 was eliminated, leaving 55 features. Every feature was made to have a mean of zero and an average of one. Five-fold stratified cross-validation was used for model selection and assessment, with the data split into a 70/30 train/test ratio and stratified by the target variable. Patterns associated with classifier performance. Three data mining classifiers were compared: RF, ANN, and KNN. 30% of the data was used for testing, while 70% was used for hyperparameter optimization during training using GridSearchCV.

A. Research Design

The study's comparative experimental design includes evaluating three edge-centred classifiers on the stabilised IoTID20 dataset. Following feature trimming and pre-processing, models were tested on the 30% held-out split of the dataset after being trained on 70% of it using 5-fold cross-validation. There were 290,972 training and 124,703 test occurrences out of the 415,675 flow records. Accuracy, ROC- AUC, F1-score, precision, Confusion matrices, recall, runtime, and maximum RAM consumption were used to assess efficacy and efficiency.

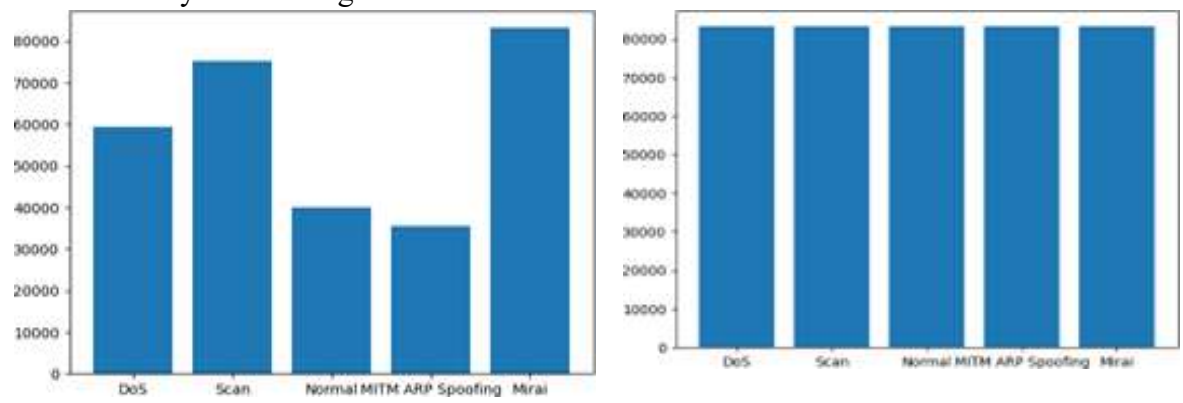
B. Description of the Dataset and Preprocessing

There are 17 descriptive tags and 58 numerical parameters in 625,783 bidirectional streams in the unprocessed IoTID20 dataset. To reduce sparsity, these labels were merged into five functional categories: Miria, MITM-ARP, Normal, DoS, and Scan. Mirai accounted for 66% of total flows; it was downsampled to 83,135 flows (20%) to minimise bias and reduce training duration. A balanced dataset with 415,675 flows was then obtained by applying SMOTE [22] to ensure that all classes had 83,135 flows apiece. Data quality evaluations found 631 infinite values (0.001%), which were replaced with finite maxima, and 1,204 (0.002%) missing values, which were filled with feature averages. Box and violin plots showed fewer than 0.4% extreme values, which were retained to preserve attack signatures. Pearson correlation analysis with a multicollinearity threshold of $|r| > 0.90$ identified three redundant metrics, reducing the feature set to 55 while preserving domain representation. The 70/30 stratified division yielded 290,972 training and 124,703 test flows, with every feature standardized using z-score trans- formation.

C. Exploratory Data Analysis (EDA)

An extensive EDA was conducted to assess class balance and feature distinguishability. Fig. 2a shows the distribution after Mirai down-sampling, at which Miria maintains 2.3× the MITM-

ARP fluxes, hence requiring synthetic balancing. Fig. 2b shows the post-SMOTE results for a properly balanced corpus. With only 0.4% outliers, the Flow KBytes/s box plots reveal that the DoS median byte rate is higher than Normal.



(a)Data Distribution (b) Balanced Data

Fig. 2. Data Balancing: (a) Data Distribution, (b) Balanced Data.

The Plots of the FlowIATMean violin show Attacks had an inter-arrival time of 0.4s, while normal is 2.6s. The correctness of the balancing pipeline and feature separability is confirmed by the scatter plots of TotFwdPkts vs. TotBwdPkts, which clearly cluster DoS traffic in the high-forward, low-backwards quadrant.

D. Tools and Libraries

Google Colab was used for all experiments, with GPU runtime enabled. NumPy, Pandas, and Python 3.8 were utilized for data processing. The Seaborn and Matplotlib were used for visualizations. Scikit-learn (v0.24) and imbalanced- learn for SMOTE oversampling were used to implement classic Machine learning and cross-validation. DL tests were conducted using the Keras high-level API and TensorFlow 2.x. All scripts were combined into a single Colab notebook, and Git was used for version control to ensure reproducibility.

E. Justification of Choice of Methods

RF has been chosen for its ability to reduce variance through bagging with minimal bias and its tolerance for high-dimensional, noisy data. Critical byte/packet-rate and timing indicators are revealed by feature importance scores, which also help with interpretability.

ANN (MLP) captures non-linear feature interactions, such as simultaneous spikes in forward/backward packet counts, that may be missed by linear or tree-based models. Overfitting is alleviated by implementing dropout regularisation and early-stopping conditions.

Considering KNN serves as a simple, independent baseline that is helpful for identifying leakage or overfitting when classes form clusters, improvements over it are therefore noteworthy. SMOTE was chosen to create synthetic minority samples by interpolation rather than duplication, strengthening decision boundaries, in light of the dataset's original class imbalance. Finally, 5-fold stratified cross-validation allows for a fair comparison of RF, KNN, and ANN by maintaining class proportions across folds and producing reliable, objective estimates of F1 score, recall, accuracy, and precision.

F. Module for Data Ingestion

Uploading the raw dataset to Google Drive is the initial stage in the workflow. The storage is

likely to be mounted in the Google Colab environment so that the file can be conveniently accessed. The dataset is read into memory using pandas.read_csv() function, producing an initial Pandas DataFrame (data). Checking at the pre-loading stage ensures correct ingestion and thus provides a secure foundation for further preprocessing.

G. IDS Architecture Proposal for IOT Networks

The recommended intrusion detection system for Internet of Things settings is a multi-stage process that examines unprocessed network traffic data, gets it ready for modeling, and assesses detection performance using various ML algorithms (see Fig. 3). The architecture is separated into six main modules: evaluation, feature selection, data splitting, data ingestion, Model training, and data preprocessing.

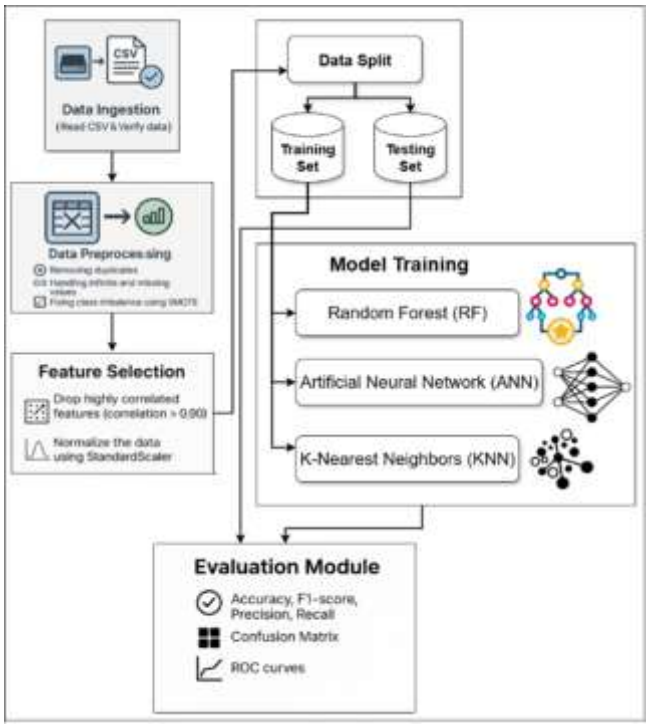


Fig. 3. The complete layout of the suggested IDS framework for IoT.

H. Data Preprocessing Module

Data cleaning is performed to handle inconsistencies and ensure uniformity:
Infinite values handling: The greatest finite value found in a column that contains infinite values is capped.
Imputation of Missing values*:* NaN entries are replaced with the mean of their respective columns.
Feature engineering: From Flow_Byts/s, a new attribute called Flow_KBytes/s is calculated.
Class filtering*:* The dataset may be divided into five primary categories: Mirai, MITM ARP Spoofing, Normal, DoS, and Scan.
Class balancing: SMOTE variation is used to balance all classes. Mirai class reduces to 20%.

I. Feature Selection Module

To reduce redundancy and improve efficiency, the *Pearson correlation matrix* is computed for all numeric features. Attributes with a correlation coefficient greater than 0.90 are removed.

J. Data Split Module

Training and testing sets are separated from the normalized dataset using Scikit-learn's split function in 70:30 ratio. Stratification (stratify = y1) ensures that the class proportions remain consistent in both subsets:

i. Training data: X_{train}, y_{train}

ii. Testing data: X_{test}, y_{test}

K. Model Training Module

As part of the proposed project, three ML models: ANN, KNN, and RF, were trained for intrusion detection in IoT contexts. To enhance their predictive performance and avoid overfitting, GridSearchCV was employed for exhaustive hyperparameter tuning. By using cross-validation [10], this method systematically evaluates every combination of hyperparameters and selects the configuration with the best performance on the validation set.

The following is a list of the hyperparameters considered for each model:

Random Forest Hyperparameters:

• The number of estimators: [50, 100, 150]

• maximum depth: [None, 10, 20]

• minimum samples split: [2, 4]

• bootstrap: [True, False]

Hyperparameters of KNN:

• The neighbors are: [3, 5, 7, 9]

• Metric: ["manhattan," "euclidean"]

• Weights: [distance, uniform]

Hyperparameters of ANN:

• Units hidden: [64, 128]

• The dropout rate: [0.0, 0.2]

• optimizer: [Adam, RMSprop]

• epochs: [50, 100]

• The batch size: [32, 64]

After applying GridSearchCV, the optimal configurations for each model are shown in Table I.

Table 1. Summary Of Each Model's Ideal Hyperparameters

Model	Best Values
RF	bootstrap = False, min_samples_split = 4, max_depth = None, n_estimators = 50
KNN	Weights = "distance" with n neighbors = 5
ANN	batch size = 32, epochs = 50, optimizer = Adam, hidden units = 128, dropout

	rate = 0.2

L. Evaluation Module

Model evaluation is carried out after training the selected algorithms (RF, KNN, ANN) with optimal hyperparameters determined via GridSearchCV. Performance is assessed in two stages:

Validation phase: Accuracy, classification reports are computed for each cross-validation fold.
Testing phase*:* The best-performing models are evaluated on unseen data to produce final graphs.

3. Theory and Calculation

The theoretical underpinnings and mathematical formulation of the proposed internet of things network intrusion detection (IoT-NIDS) framework are provided in this section. It connects the methodological implementation, which focuses on the key learning paradigms and computations used in this work, to the introduction environment.

The proposed framework uses three machine learning classifiers that include Random Forest (RF), Artificial Neural Network (ANN), and K-Nearest Neighbors (KNN). The reason why these models are chosen is that they are suitable to resource-constrained IoT systems where it is important to balance between the detecting accuracy and the computation efficiency effectively [5], [10], [11].

RF is an ensemble learning method, which takes several decision trees bootstrapped together (bagging) to decrease variance and implementation of overfitting. Its strength in seeking noisy and high dimensional data and the interpretability provided by feature importance measures makes it especially applicable in the case of IoT intrusion detection [5], [10].
KNN is a distance-based, non-parametric, classification model which is a lightweight baseline model. It is useful in verifying the structure of feature space and for detecting data leakage or overfitting because it is efficient when attack and benign traffic are separable clusters in the feature space [11], [17].

ANN allow the modelling of non-linear correlations in network traffic behaviour, including correlated changes in packet rate and flow statistics, which can be missed by linear and tree models. This feature offers a complementary deep learning view for detecting advanced intrusion patterns in IoT traffic [14], [15]. These models are combined into Knowledge Discovery in Databases (KDD) workflow so that they organize data prior to processing, feature selection, and model evaluation to fit the IoT network traffic.

3.1 Mathematical Expressions and Symbols

A. Pearson Correlation for Feature Selection

To eliminate redundant features, Pearson correlation coefficients were computed.

The formula for calculating the coefficient for Pearson correlation is:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})\,(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\ \sqrt{\sum_{i=1}^{n}(y_i - y)^2}} \qquad (1)$$

where:
• r_{xy} = The coefficient of correlation of Pearson between y
and x variables
• $\bar{x},\ \bar{y},$ = mean of x and y respectively
• n = number of observations
• x_i, y_i = independent sample points

B. Z-score Standardization

To ensure equal contribution of all features during model training, Z-score normalization was applied [4].

$$z = \frac{x - \mu}{\sigma} \qquad (2)$$

where:
• x represents the initial feature value
• μ is the mean for feature
• σ = standard deviation for feature
• z is the value of the standardized feature

C. Random Forest Prediction

For classification, the Random Forest ensemble aggregates predictions from Tdecision trees using majority voting:

$$\hat{y} = \text{mode}\{\, h_t(x) \mid t = 1,2,\dots,T \,\} \qquad (3)$$

where $h_t(x)$ is the prediction of the t-th decision tree.

D. K-Nearest Neighbors Decision Rule

In the K-Nearest Neighbors classifier, the class label of a test sample $x$is determined by majority voting among its $k$nearest neighbors:

$$\hat{y} = \text{mode}\{\, y_i \mid x_i \in N_k(x) \,\} \qquad (4)$$

where $N_k(x)$denotes the set of k nearest neighbors based on Euclidean distance.

E. Artificial Neural Network Activation

For a neuron $j$with inputs x_i, weights w_{ij}, and bias b_j, the activation output is computed as:

$$a_j = \Phi\left(\sum_{i=1}^{n} w_{ij}\, x_i + b_j\right) \qquad (5)$$

where $\Phi(\cdot)$represents the ReLU activation function used in the hidden layers. The output layer uses the Softmax function to estimate multi-class probabilities.

4. Results and Discussion

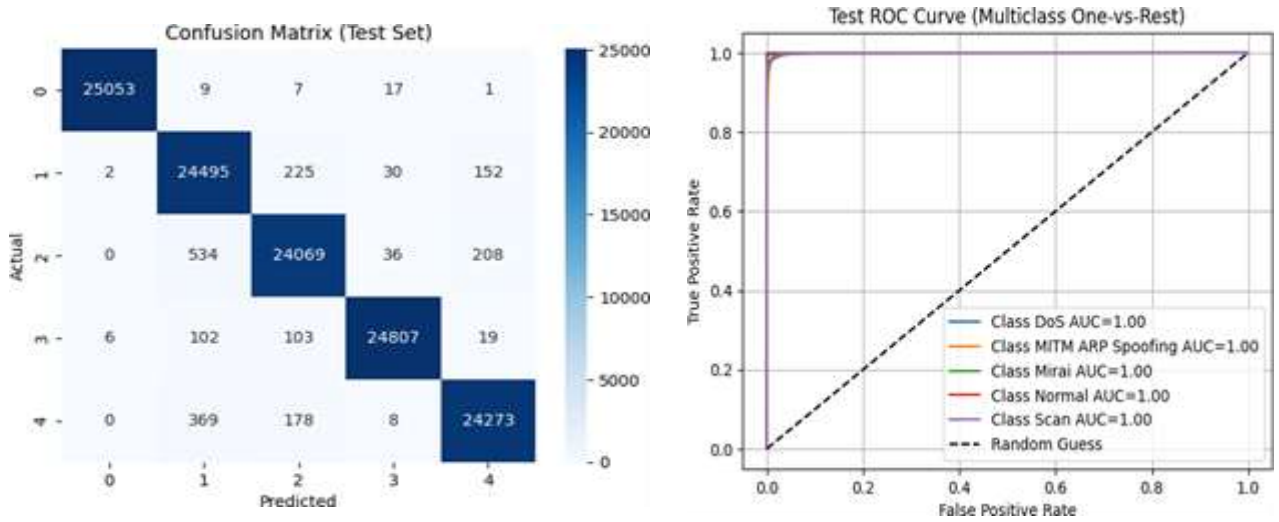
The three adopted strategies KNN, ANN, and RF, are examined in this part based on their performance on validation and test datasets. Relevant to the study's emphasis, the primary evaluation parameter utilized to present the findings is accuracy. The Research Methodology included every stage of preprocessing, parameter adjustment, and training.

A. Random Forest

RF maintained excellent performance on the test dataset and demonstrated high Accuracy on the validation dataset. Table 2 provides a thorough analysis of classification performance across all target classes, displaying segment-level F1 scores, precision, and recall for both datasets. Furthermore, the corresponding confusion matrices and ROC curves, as illustrated in Figure 4, provide a graphical depiction of the model's predictive behaviour.

Table 2: The Effectiveness of RF on Test And Validation Data

Class	Validation			Test		
	Prec.	Rec.	F1	Prec.	Rec.	F1
DoS	1.00	1.00	1.00	1.00	1.00	**1.00**
MITM ARP	0.96	0.99	0.97	0.96	0.98	**0.97**
Mirai	0.98	0.97	0.98	0.98	0.97	**0.97**
Normal	1.00	0.99	0.99	1.00	0.99	**0.99**
Scan	0.99	0.98	0.98	0.98	0.98	**0.98**
Accuracy	**0.9900**			**0.9800**		



(a) The Confusion Matrix (b) The ROC Curve

Fig. 4. Random Forest (RF): (a) confusion matrix, (b) ROC curve.

The Confusion Matrices show the true positive, true negative, false positive, and false negative, distributions for each class, making the misclassifications easy to understand. In the meantime,

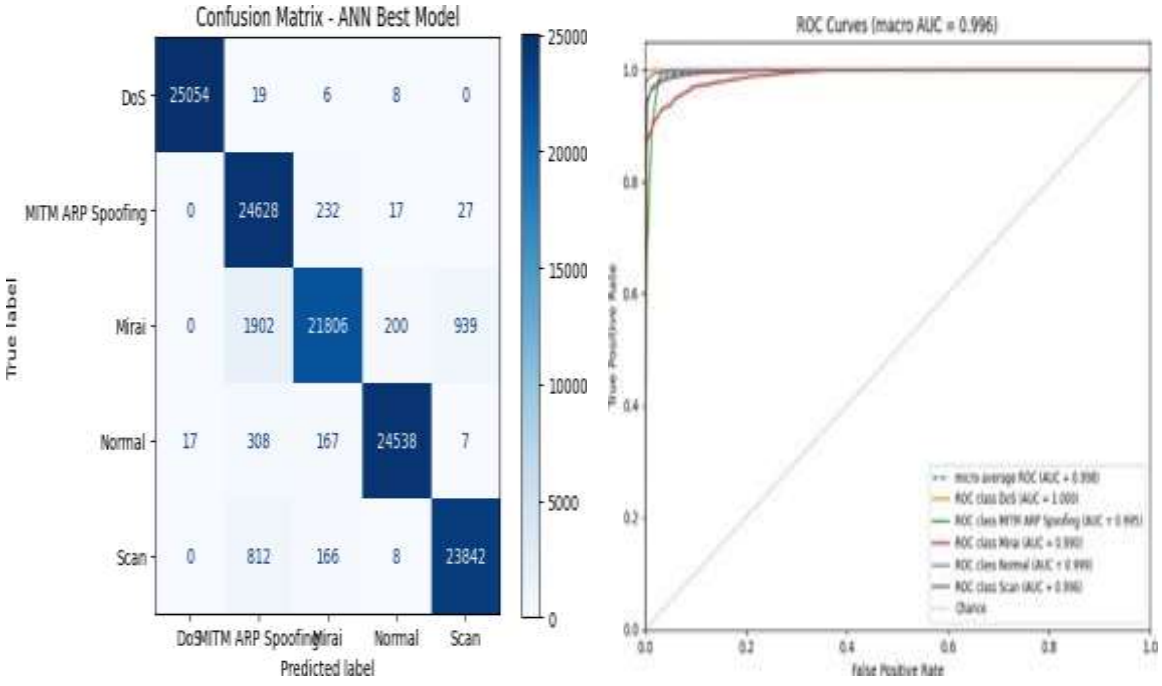
the ROC curves show the trade-off between each class's true-positive and false-positive rates, highlighting the model's strong discriminatory power across a range of decision thresholds.

B. Artificial Neural Network

On the test dataset, ANN showed significant development, even though with a lesser accuracy than RF and KNN. Table 3, presents the class-level performance, including the F1-scores, precision, and recall for each class in both the test and validation phases. The corresponding confusion matrices and ROC curves, presented in Figure 5, visually illustrate the classification outcomes and discriminative power of the ANN model. There are clear patterns of misclassification as the confusion matrices show the right and wrong predictions for each class. The ROC curves confirming the ANN's consistent and competitive performance despite somewhat lower accuracy.

Table 3 The Effectiveness of ANN on Test and Validation Data

Class	Validation			Test		
	Prec.	Rec.	F1	Prec.	Rec.	F1
DoS	1.00	1.00	1.00	1.00	1.00	**1.00**
MITM ARP	0.89	0.98	0.93	0.89	0.99	**0.94**
Mirai	0.97	0.89	0.93	0.97	0.88	**0.93**
Normal	0.99	0.98	0.99	0.99	0.98	**0.99**
Scan	0.95	0.96	0.95	0.96	0.96	**0.96**
Accuracy	**0.9691**			**0.9617**		



(a) The Confusion Matrix (b) The ROC Curve

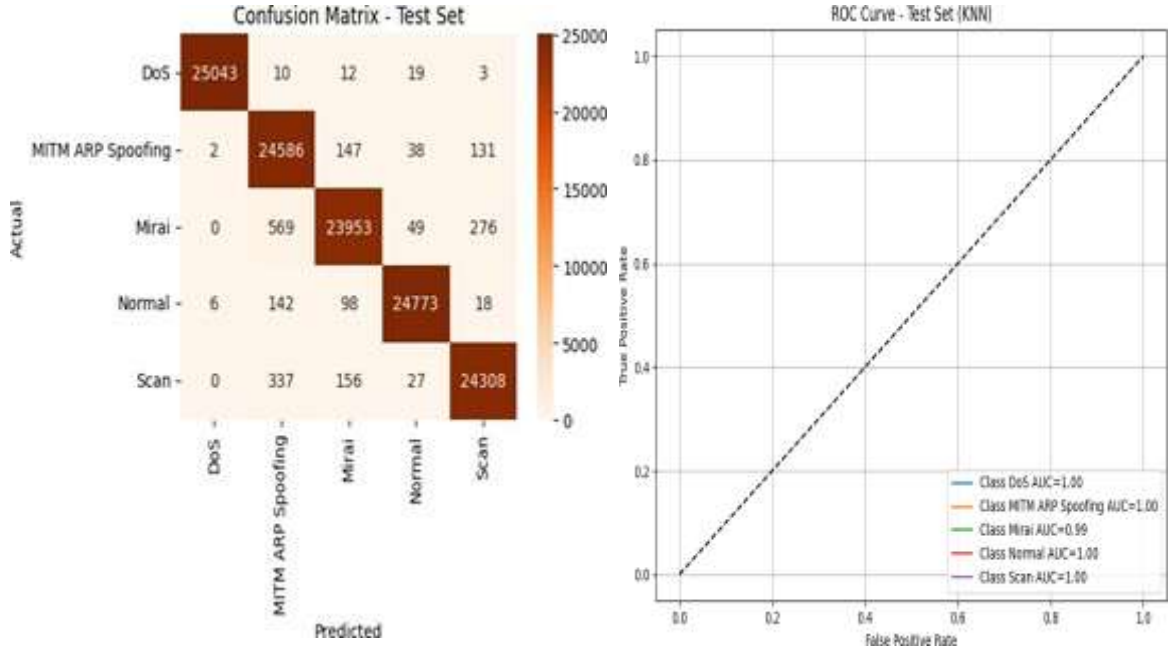
Fig. 5. ANN: (a) confusion matrix, (b) ROC curve.

C. K-Nearest Neighbours

The KNN demonstrated the same accuracy as the RF test and provided high precision in both datasets, but with slightly lower validation accuracy. Table 4 summarizes the metrics, providing class-level recall, F1 scores and precision for both the validation and test phases. The associated confusion matrices and ROC curves, depicted in Figure 6, provide a visual representation of the KNN model's classification performance. The confusion matrices show the distribution of accurate and inaccurate predictions for each class, making it possible to spot trends in classification errors. Furthermore, the ROC curves demonstrate KNN's great versatility and competitive performance compared to ANN and RF.

Table 4 : The Effectiveness of KNN on Test and Validation Data

Class	**Validation**			**Test**		
	Prec.	Rec.	F1	Prec.	Rec.	**F1**
DoS	1.00	1.00	1.00	1.00	1.00	**1.00**
MITM ARP	0.96	0.98	0.97	0.96	0.99	**0.97**
Mirai	0.98	0.97	0.97	0.98	0.96	**0.97**
Normal	0.99	0.99	0.99	0.99	0.99	**0.99**
Scan	0.98	0.98	0.98	0.98	0.98	**0.98**
Accuracy	**0.9820**			**0.9800**		



(a) The Confusion Matrix (b) The ROC Curve

Fig. 6. KNN: (a) confusion matrix, (b) ROC curve.

D. Comparative analysis of experimental results

Table 5 provides comparisons of the accuracy of the proposed IDS framework. Table 6 includes a comparative analysis of the experimental results in the IoTID20 dataset. With an accuracy of 98.00%, our model outperformed Kasongo & Sun [21] (97.30%) and Al-Hawawreh et al. [20]

(96.80%). Methodological differences affect the results even though the three publications employed the same data set and measurements.

Class imbalance was another limitation that Al-Hawawreh et al. [20] did not fully address, noting it as a limiting factor for RF performance. Similarly, working with uneven data, Kasongo & Sun [21] focused predominantly on multiclass detection efficiency. However, as part of a scalable IDS workflow, we used downsampling and SMOTE to balance the dataset, along with RF and hyperparameter tuning. Since accuracy was consistently reported across the trials, it was the only parameter used for comparison to maintain fairness. The findings imply that fine-tuning and balanced data preparation can greatly improve RF performance for IoT intrusion detection.

Table 5: Model Accuracy Comparison

Model	Validation Accuracy	Test Accuracy
RF	0.9900	**0.9800**
KNN	0.9820	**0.9800**
ANN	**0.9691**	**0.9617**

Table 6: Comparison of Accuracy with Related Works (IOTID20)

Study	Ref.	Accuracy
Proposed Framework		**0.9800**
Al-Hawawreh et al.	[20]	**0.9680**
Kasongo & Sun	**[21]**	**0.9730**

5. Conclusions

The effectiveness of supervised machine learning techniques was tested in this work to detect Internet of Things (IoT) in a flow-based intrusion detection framework. The suitability of the random Forest (RF), Artificial Neural Network (ANN), and K-Nearest Neighbors (KNN) classifiers were evaluated to determine their aptitude in accurate and implementable intrusion detection within the context of the IoT. A multiclass dataset was balanced and then a single preprocessing pipeline was used and hyper-parameter optimization and five-fold stratified cross-validation were performed. An independent test set was used to assess the performance of the final evaluation to give reliable performance comparison. The experimental findings reveal that the Random Forest classifier performed better in all the evaluation measures. The test data yielded an optimised RF model with a validation accuracy of 0.9900, a mean Area Under the Curve (AUC) of 0.9993, and an F1-score of 0.98, which is better than both the ANN and KNN models. The KNN classifier achieved competitive results with validation and test accuracies of 0.9820 and 0.9800, respectively, whereas the ANN model achieved validation and test accuracies of 0.9691 and 0.9617, respectively. These results align with the existing

literature, which indicates that ensemble-based tree models are powerful baselines for tabular intrusion detection because they are robust and require minimal feature engineering. Although the results are promising, the study has several limitations that limit its generalizability. The experiments were carried out with the help of one uniform dataset and there was no temporal hold-out testing conducted in order to test the robustness to concept drift. Also, there was no support for runtime or memory profiling, which limits the conclusions about practical application in resource-constrained IoT or edge environments. The next research topic is cross-dataset validation of heterogeneous IoT traffic across different network configurations, device types, and time periods. Also, additional studies on the computational efficiency, model scaling, and the model's ability to run on the edge under edge-computing conditions will be conducted. The objectives of these extensions are to enhance the robustness, external validity, and utility of machine learning-based intrusion detection systems in real-world IoT settings.

Acknowledgements

This research was funded by the Irish Research Council under grant GOIPD/2023/1341

Conflict of Interest

The authors declare no conflict of interest.

References

[1] Cisco Systems, “Cisco annual internet report (2018–2023),” Cisco, White Paper, 2023. [Online]. Available: https://www.cisco.com. Accessed: Mar. 10, 2024.
[2] ENISA, “Cybersecurity challenges for the Internet of Things,” European Union Agency for Cybersecurity, Heraklion, Greece, 2022. [Online]. Available: https://www.enisa.europa.eu. Accessed: Mar. 12, 2024.
[3] IBM Security, “Cost of a data breach report 2023,” IBM Corp., Armonk, NY, USA, 2023. [Online]. Available: https://www.ibm.com/security. Accessed: Mar. 15, 2024.
[4] J. Han, M. Kamber, and J. Pei, Data Mining: Concepts and Techniques, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann, 2012.
[5] S. Haykin, Neural Networks and Learning Machines, 3rd ed. Upper Saddle River, NJ, USA: Pearson, 2009.
[6] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization,” in Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy (ICISSP), Funchal, Portugal, 2018, pp. 108–116, doi: 10.5220/0006639801080116.
[7] L. Breiman, “Random forests,” Mach. Learn., vol. 45, no. 1, pp. 5–32, Oct. 2001, doi: 10.1023/A:1010933404324.
[8] C. Cortes and V. Vapnik, “Support-vector networks,” Mach. Learn., vol. 20, no. 3, pp. 273–297, Sep. 1995, doi: 10.1007/BF00994018.
[9] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., San Francisco, CA, USA, 2016, pp. 785–794, doi: 10.1145/2939672.2939785.
[10] F. Pedregosa et al., “Scikit-learn: Machine learning in Python,” J. Mach. Learn. Res., vol. 12, pp. 2825–2830, Oct. 2011.

[11] M. Ring et al., “Flow-based benchmark data sets for intrusion detection,” in Proc. 16th Eur. Conf. Cyber Warfare Secur. (ECCWS), Dublin, Ireland, 2017, pp. 361–369.
[12] Y. Meidan et al., “ProfilIoT: A machine learning approach for IoT device identification based on network traffic analysis,” in Proc. ACM Int. Conf. Internet Things Des. Implement. (IoTDI), Orlando, FL, USA, 2017, pp. 506–509, doi: 10.1145/3054977.3055067.
[13] A. Doshi, N. Apthorpe, and N. Feamster, “Machine learning DDoS detection for consumer Internet of Things devices,” in Proc. IEEE Secur. Privacy Workshops (SPW), San Jose, CA, USA, 2018, pp. 29–35, doi: 10.1109/SPW.2018.00013.
[14] J. Gubbi et al., “Internet of Things (IoT): A vision, architectural elements, and future directions,” Future Gener. Comput. Syst., vol. 29, no. 7, pp. 1645–1660, Sep. 2013, doi: 10.1016/j.future.2013.01.010.
[15] N. Moustafa and J. Slay, “UNSW-NB15: A comprehensive data set for network intrusion detection systems,” in Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS), Canberra, ACT, Australia, 2015, pp. 1–6, doi: 10.1109/MilCIS.2015.7348942.
[16] M. Hodo et al., “Threat analysis of IoT networks using artificial neural network intrusion detection system,” in Proc. IEEE Symp. Comput. Commun. (ISCC), Heraklion, Greece, 2016, pp. 626–631, doi: 10.1109/ISCC.2016.7543834.
[17] A. Alrashdi et al., “Machine learning-based intrusion detection system for IoT networks,” J. Supercomput., vol. 77, no. 7, pp. 7531–7555, Jul. 2021, doi: 10.1007/s11227-020-03542-9.
[18] S. Rathore et al., “IoT security: Challenges and solutions,” in Proc. IEEE Int. Conf. Adv. Comput. Commun. Control Netw. (ICACCCN), Greater Noida, India, 2018, pp. 1–6, doi: 10.1109/ICACCCN.2018.8748796.
[19] M. A. Ferrag et al., “Security for 5G and IoT networks: A survey,” IEEE Access, vol. 6, pp. 38589–38605, 2018, doi: 10.1109/ACCESS.2018.2855152.
[20] Sagili, S.R., 2024, September. Prompt-Instructed Generative based AI for Enhancing Transformer effectiveness Analysis. In 2024 Asian Conference on Intelligent Technologies (ACOIT) (pp. 1-5). IEEE.
[21] Gaddam, M.K., 2025, September. Architecting Observability for AI-Driven Microservices at Scale. In 2025 3rd International Conference on Intelligent Cyber Physical Systems and Internet of Things (ICoICI) (pp. 1830-1838). IEEE.
[22] Anomaly, V.K.S.A.B., Detection for 5G Core and RAN Components-International Journal of Scientific Research in Engineering and Management (IJSREM) Volume: 06 Issue: 01| Jan-2022.