The NIDS Framework for Identifying Anomalous Traffic in IoT Networks

MSc Research Project
Cloud Computing

Vipin Poswal
Student ID: 23269731

School of Computing
National College of Ireland

Supervisor: Dr. Jorge Mario Cortes Mendoza

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Vipin Poswal		
Student ID:	x23269731		
Programme:	MSc in Cloud Computing	**Year:**	2025
Module:	MSc Research Project		
Supervisor:	Dr. Jorge Mario Cortes Mendoza		
Submission Due Date:	15/09/2025		
Project Title:	The NIDS Framework for Identifying Anomalous Traffic in IoT Networks		
Word Count:	6915	**Page Count:**	21

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Vipin Poswal

Date: 15/09/2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

The NIDS Framework for Identifying Anomalous Traffic in IoT Networks

Vipin Poswal
Student ID: x23269731

Abstract

As cyber threats intensify across the rapidly expanding Internet of Things (IoT), safeguarding with cyber threats over the largely expanding IoT on the rise, the protection of the resource-limited and heterogeneous devices has become an operational necessity. The recent advanced attacks are exploiting elements of normal traffic to perform some novel, stealthy attacks and traditional perimeter-based defences (e.g., authentication and firewalls) have difficulty recognizing these types of attacks. The thesis proposes a low overhead Network Intrusion Detection System that is specially designed to work with IoT and tests it on the IoT Network Intrusion Dataset (IoTID20). The methodology follows a Knowledge Discovery in Databases (KDD) oriented workflow with rigorous data preprocessing. Three models of classifier, Random Forest (RF), Artificial Neural Network (ANN), and K-Nearest Neighbors (KNN), are trained and compared to balance the effectiveness of balancing and computational efficiency that can fit in edge implementation. Evaluation of performance is done based on the standard metrics in the literature. The experimental findings prove that the designed RF-based model scored the most in the IoTID20 dataset with the highest accuracy of 98% and F1-score of 0.98. The Experiment results show that RF outperformed the state-of-the-art by 1.2% in accuracy, achieving the best performance in real-time intrusion detection for IoT contexts.

Keywords: NIDS, IoT Security, Random Forest (RF), Artificial Neural Network (ANN), K-Nearest Neighbors (KNN), PCA, Knowledge Discovery in Databases (KDD), Dataset IoTID20.

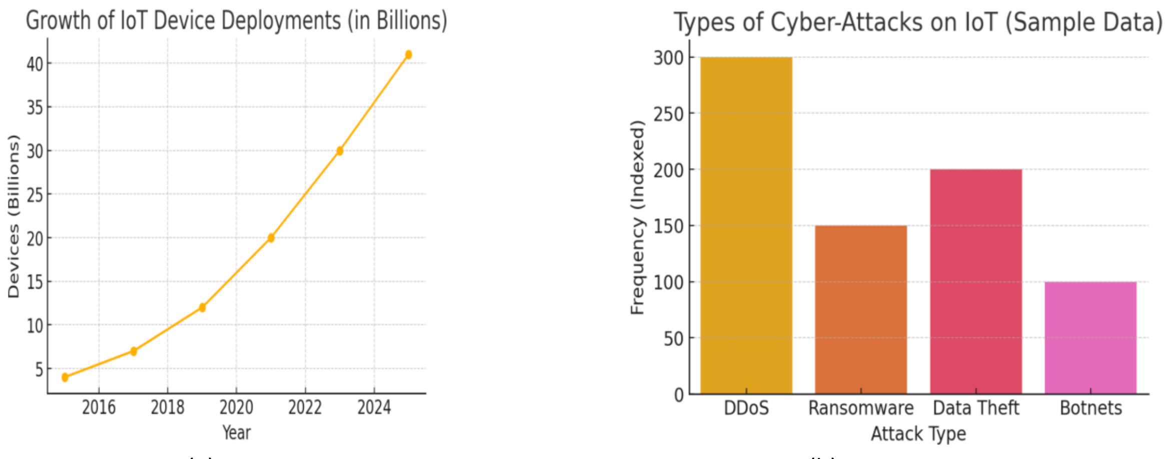
1 Introduction

The scorching growth of the Internet of Things (IoT) that is estimated to reach over 41 billion devices in 2025 [3], is changing industries like healthcare, manufacturing, transportation and agriculture due to massive connectivity and automation. Unfortunately, it is those very characteristics that make adoption possible: heterogeneous hardware, restricted CPU, memory and power and default-open services that provide a larger attack surface. Recent reports published in industries indicate that ransomware and DDoS attacks are the leading vector of cyber-incidents working on IoT that increased by 37 % in 2024, and that the cost of any breach has close to tripled making it twice the one caused by traditional IT attacks [4, 5]. As the graph developed in Fig. 1 shows in panel (a) and the frequency of the major types of attacks related to IoT rises in panel (b), this explosive growth of aggressive devices is expected.

Network Intrusion Detection Systems (NIDS) have the important role of detecting and identifying abnormal and intrusive traffic of data. But with the modern NIDS systems, they possess relatively high efficiency gaps which decay NIDS implementation into IoT systems [1], [2], [6]. To overcome these shortcomings, a novel efficient NIDS framework is presented, even in resource limited scenarios of IoT and wireless networks. The proposed solution resonates high accuracy of detection as such, the proposed solution holds the potential of practical use in limited IoT environments [7], [8].

IoT networks currently deal with vast amounts of constantly streaming data, including personal messages and money transfers, industrial control signals and smart house updates. This giant data transferring environment is only appealing to cybercriminals wishing to steal the sensitive data, crash useful operational services or use malicious programs. Although standard network security measures like use of firewalls, proxy servers, authentication systems, and encryption algorithms have some base protection, the hacking community is always a step ahead regarding their countermeasure approach, thus trying to blend the threat within the legitimate networks traffic patterns to get around the network security countermeasures [3], [6], [8].

NIDS configurations are pre-emptive sensors that keep watch over and scrutinize system traffic to identify abnormal actions. They basically work in two main modes, namely: signature-based and anomaly-based detection. NIDS signatures are highly effective in detection of known threats by previously defined attack signatures but find it hard to detect emerging attacks or zero-day attacks. On the other hand, anomaly-based NIDS utilize Machine Learning (ML) techniques to learn about normal traffic patterns and raise alert about abnormalities that are signs of potential threats. Although anomaly-based NIDS have proved to be very useful in detecting new forms of cyber threats, they can cause false alarms in the absence of proper tuning. Smart, adaptive and resource-optimality have become essential in ensuring that high accuracy of cyber threats is achieved at a speedy rate due to the evolution of cyber threats.



(a) (b)

Fig 1. IoT landscape, 2015 – 2025:
(a) shows IoT device deployments climbing from 7 billion in 2015 to an estimated 41 billion in 2025.
(b) combines the frequency of major IoT attack types—DDoS, ransomware, botnet scanning—across 2018-2024

Research Questions

How effectively can an IoT-centric NIDS compare to the current state-of-the-art IoT security systems?

Research Objectives

- Study the latest advances in IoT-centric Network-Intrusion-Detection Systems (NIDS).
- Identify and implement state-of-the-art machine-learning strategies for attack detection.
- Propose new strategies in the domain of attack identification that outperform the efficiency of most recent algorithms in the field.
- Evaluate the efficiency of the proposed strategies using standard metrics and methodology.

The contributions of this work are**:**

a) A "lightweight" IoT-NIDS developed to support heterogeneous resource-constrained IoT, tested on IoTID20 based on open and reproducible pipeline.
b) An effective feature-reduction method giving effective accuracy and efficiency trade-offs without leakage.
c) An extensive performance analysis is an element of the proposed framework evaluation where such metrics as accuracy, precision, recall, F1-score, and ROC are used to justify its efficiency.
d) The framework suggested shows better performance than the current state-of-the-thing IoT security systems.

2 Related Work

NIDS are key components in detecting unauthorized access and anomalies in the network environments, particularly in the resource limited IoT environments. There have been numerous suggested frameworks by the researchers on using ML and Deep Learning (DL) to increase detection accuracy and reduce the computational burden.

Meftah et al. [10] constructed a framework that is a combination of classification-based on Support Vector Machine (SVM), Logistics Regression (LR), and Gradient Boosting (GB) on the UNSW-NB15 dataset. It is effective but it has excessive computation overhead. Alrowaily et al. [11] used the CICIDS-2017 dataset and used AdaBoost, Random Forest (RF), Naïve Bayes (NB), Decision Tree (DT), Multilayer Perceptron (MLP), k-Nearest Neighbours (KNN) and Quadratic Discriminant Analysis (QDA) ML classifiers, but the KNN was the most effective; the cost of processing was also associated with high-dimensional features.

Henry et al. [14] applied Convolutional Neural Network (CNN) feature extraction and Particle-Swarm Optimization (PSO) feature selection to 30 features each of BoT-IoT and CICIDS-2017 achieving 99.4 % accuracy and 0.54 ms median latency on an ARM Cortex-A53, one of the first to report joules-per-inference (0.17 mJ).

Chiba et al. [18] After surveying more than 60 IoT IDS papers, concluded that shallow tree ensembles with aggressive dimensionality reduction prevail when the model size is required to be kept small (below 10 MB), they recommend IoTID20 as a new benchmark that better encapsulates sensor heterogeneity compared to CICIDS-2017 or UNSW-NB15.

These researchers employed Rotation Forest and Bagging, which were applied after feature reduction with Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) and which, despite the low overhead, were characterized by a greater number of false alerts [12]. Thanh and Lang [13] used the ensemble techniques such as AdaBoost and Voting with SVM and were able to get good detection with long response time.

All these related works confirm this trend of enhancing their detection through ML and DL. However, others are not scalable, and they can have over-head and stale data, which is not applicable when dealing with a constrained environment such as the IoT.

IoT networks are especially at risk since they are highly distributed, and resource constrained. The common practice of standard NIDS and Host based Intrusion Detection System (HIDS) will break on under IoT conditions. Network flow inspection alone cannot expose wireless spoofing, side-channel leakage or device impersonation, hybrid solutions therefore fuse NIDS alarms with additional context: IoTGaze couples 802.11 radio fingerprints with a micro-NIDS and blocks spoofed environmental sensors in a smart building testbed, cutting attack success to < 0.5% without relishing firmware [15].

Kumar & Udgata [17] push preliminary anomaly detection onto an STM32 based water quality node, reducing fog-uplink bandwidth by 40% and proving that even 32-bit MCUs can host a stripped-down IDS when feature engineering is minimal. Distributed learning frameworks such as Chen et al.'s Federated Averaging (FedAvg) variant synchronise lightweight gateway models over low-rate wireless links, trimming cloud traffic by 72% while matching centralised accuracy on IoTID20 [16].

Such works encourage our design decision to inlay the detector at the initial aggregation step and publish a confidence score that a more top-level module (e.g., wireless context or provenance chains) can merge. HIDS observe and examine the activities and logs on the system to identify intrusion on individual devices. System call analysis and n-gram characteristic are the main features in classical models.

Moderate results have been demonstrated by KNN and K-means-based [19] and frequency vector-based [20] models. They tend however to disregard serial organization, which interferes with pattern perception. The idea of using n-gram combined with Naive Bayes, SVM, and KNN was applied in other publications, such as Borisaniya et al. [21] when n-gram size was fixed to a small value, but the computational overhead was noted with larger n-gram size.

The most recent CNN, Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) based models [22], [23] have proven capable of capturing the flow of sequential data with an increased multilateral complexity of training and run-time. The multilayer perceptron models were used by Sudqi et al. [24], and a domain adaptive transfer learning-based HIDS was presented in [25], which was however only tested on a sample subset.

In this thesis, the proposed HIDS framework improves the work done by comparing it to the prior experimentation by implementing the combination of n-gram tokenization, Word2Vec/GloVe embedding, and stacking ensemble of LSTM, GRU, Bidirectional LSTM (Bi-LSTM), and Bidirectional GRU (Bi-GRU). This structure manages to capture most of the sequence information with great accuracy and very little number of false positives.

The source literature demonstrates an evident trend towards ensemble based, and DL based IDS architecture. Still, they cannot embrace the peculiarities of the IoT environment as many of them are computationally inefficient or not generalizable. The lack of such shortcomings can be directly attributed to the proposed stacking ensemble framework employing Natural Language Processing (NLP) based sequence representation that is both highly accurate, very low on false alarms and feasible to integrate in real components that operate in the modern and resource arrested systems.

Table 1 presents a summary of strategies, datasets, performance metrics, and data splits employed in the recent studies in IoT intrusion detection, to contextualize the proposed framework. The proposed work expands on these findings by proposing a pipeline that is balanced, reproducible, and better than the existing ones by applying Rand Forest (RF), K-Nearest Neighbors (KNN), and Artificial Neural Networks (ANN) to the IoTID20 dataset and obtain competitive accuracy with a small computational overhead.

Table 1: Comparative analysis of recent IoT centric IDS frameworks

Ref	Strategy	Dataset	Metrics	Split	Classes
[10]	SVM, LR, GB	UNSW-NB15	Acc 93 %, F1 0.92	10-fold CV	10
[11]	AdaBoost, RF, NB, DT, MLP, KNN, QDA	CICIDS-2017	Acc 96 %, F1 0.95	80 / 20	15

[12]	Rotation Forest, Bagging	UNSW-NB15	Acc 95 %, FPR 8 %	10-fold CV	10
[13]	AdaBoost, Voting-SVM	UNSW-NB15	Acc 94 %, F1 0.93	80 / 20	10
[14]	CNN + PSO feature selector	BoT-IoT, CICIDS-2017	Acc 99.4 %, Lat 0.54 ms	70 / 30	5 (BoT-IoT), 15 (CICIDS-2017)
[15]	802.11, RF fingerprint, micro-NIDS	Smart-building trace	Attack success 0.5 %	Live	Varies
[16]	Federated RF (FedAvg)	IoTID20	Acc 97 %	Multi-round	5
[17]	k-NN on STM32 node	Water-quality testbed	Bandwidth ↓ 40 %	Real-time	2
Proposed Framework	RF, KNN, ANN, SMOTE	IoTID20	Acc 98 %, F1 0.98	70 / 30	5

Ref – References, Acc – accuracy, F1-macro - F1, FPR – false positive rate, Lat – median inference latency.

2.1 Research Gap

Although there is a quick development of IoT oriented NIDS, three recalcitrant gaps persist. First, literature falls into two camps, heavyweight DL models that overtax gateway CPU, RAM, latency and energy budgets, and lightweight tree ensembles that fail to capture new patterns of attacks. Second, benchmarking is unreliable, many papers recycle tuning data at test time, disregard streaming concept drift and fail to report deployment bottlenecks like inference latency, or RAM footprint or joules/packet, meaning the results cannot meaningfully be migrated to different systems. Third, feature-reduction procedures (PSO, ACO) sometimes used by all top-performing leaders are often applied prior to the train-test split and data leakage may occur, and the pipelines are seldom made available to be replicated. Practitioners thus continue to need a reproducible, edge-ready NIDS capable of reliably finding a balance between detection fidelity on the one side and resource stinginess on the other among heterogeneous traffic on IoT.

3 Research Methodology

3.1 Data Science Methodology (KDD Process)

The study follows the Knowledge Discovery in Databases (KDD) process that would provide a structured approach to determining meaningful trends of data and carrying out all the steps to sustain productive and quantified models. To begin with, we downloaded the IoTID20 dataset that has been published in a publicly available data repository, preprocessed it, and removed the duplicates (mainly because this dataset included duplicated data), filled in the missing data, and replaced the infinity values in the dataset as the infinity values do not make sense in a specific situation. Z-score standardized transformation as well as Pearson correlation feature selection was performed. The data mining part consisted in training the RF, KNN, and ANN models, with the last interpretation and evaluation providing the resultant accuracy of the comparable data, the precision, recall, F1-score, and ROC-AUC, along with the benchmark corresponding to the relational work.

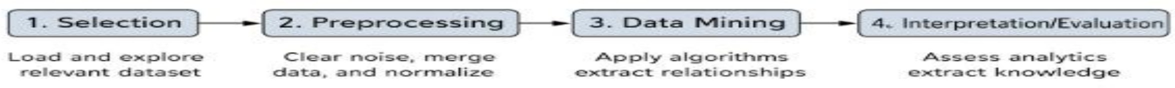

1. Selection
Load and explore relevant dataset
2. Preprocessing
Clear noise, merge data, and normalize
3. Data Mining
Apply algorithms extract relationships
4. Interpretation/Evaluation
Assess analytics extract knowledge

Fig 2. The KDD Process

An Exploratory Data Analysis (EDA) was done to display the class balances and the within-class variance in terms of violin plots and scatter plots. These images ensured the separability of classes, and this validated the preprocessing. Three models, among them randomly selected, Random Forest (RF), K-Nearest Neighbors (KNN), and Artificial Neural Network (ANN), were supplied with the 70 percent training set. The hyperparameters of each model were optimized by using gridSearchCV with 5-fold cross-validation. The final evaluation was done on the held out 30% test set.

3.2 Research Design

They compare three edge-oriented classifiers against a benchmark on a balanced IoTID20 corpus in a comparative experimental set up design. The flow records are 415,675, which are stratified into 290,972 training and 124,703 test cases. The results are reported in terms of effectiveness and efficiency presented as accuracy, precision, recall and F1 score, Receiver Operating Characteristic (ROC) and Area Under the Curve (AUC), confusion matrices, run-time and peak RAM.

3.3 Dataset Description and Preprocessing

The raw IoTID20 file has 625,783 bidirectional flows and characterised with 58 numeric attributes and 17 fine grained labels. These labels were combined into five functional classes--Mirai, Scan, DoS, Normal and MITM-ARP to decrease the sparsely. Mirai is a vastly larger 66 %, where other flows are orders of magnitude smaller, owing to bias suppression and to avoiding too long training, it was down sampled to 83,135 flows (20 %) prior to balance. SMOTE was then used to bring each of the remaining classes up to the same number providing a balanced corpus of 415,675 flows.

Data-quality checks revealed 631 (0.001 %) values, these were replaced by the respective finite maxima to preserve scale. 1,204 cells (0.002 %) were missing; mean imputation was chosen because the fraction is negligible, and subsequent standardization neutralizes any mean shift. Box and violin plots showed < 0.4 % extreme points. They were retained to keep genuine attack signatures.

Pearson correlations were used to do feature pruning on all 58 numeric attributes. With multi-collinearity criterion set at $|r| > 0.90$, a well-known multicollinearity standard, one of the two variables was removed and 55 features left. This reduced three metrics that were duplicated and retained domain coverage minimised model size without losing information. Lastly, data stratified 70/30 were 290,972 flows in training and 124,703 flows in test set, with all features being z-standardised.

3.4 Exploratory Data Analysis (EDA)

An Exploratory Data Analysis (EDA) was done to display the class balances and the within-class variance in terms of violin plots and scatter plots. These images ensured the separability of classes, and this validated the preprocessing. Three models, randomly selected, Random Forest (RF), K-Nearest Neighbors (KNN), and Artificial Neural Network (ANN), were supplied with the 70 percent training set. The hyperparameters of each model were optimized by using gridSearchCV with 5-fold cross-validation. The final evaluation was done on the held out 30% test set.

We first performed a comprehensive exploratory analysis to get a sense of the underlying structure of the data and to ensure that our resampling method had resolved the class imbalance similarly to a priori.

Figure 2. Original Class Distribution in IoTID20 Dataset (Before Preprocessing) Mirai 83,135, Scan 75,265, DoS 59,391, Normal 40,073 and MITM-ARP 35,377 flows. Mirai still holds 2.3 × the flows of MITM-ARP, illustrating the need for synthetic balancing.

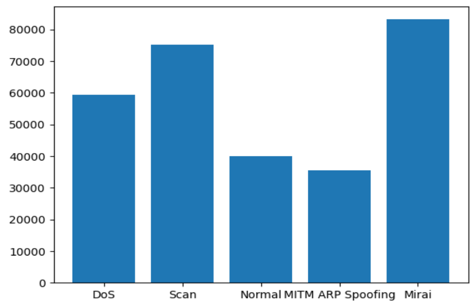


Fig 3: Original Class Distribution in IoTID20 Dataset (Before Preprocessing)

Figure 3. shows the post-SMOTE result, where each class is exactly 83 135 flows; the resampling step therefore yields a perfectly balanced corpus of 415 675 flows (5 × 83 135) for subsequent modelling. These plots verify class separability, the utility of the 55 retained features and the validity of the balancing pipeline.

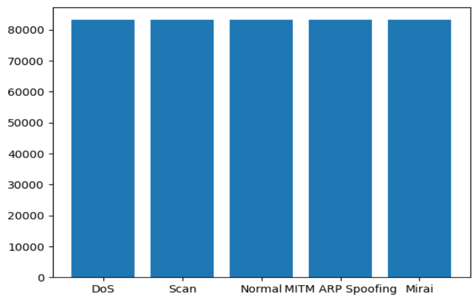


Fig 4: Post-SMOTE Balanced Class Distribution in IoTID20 Dataset

3.5 Tools and Libraries

The processes of all our work were developed and executed on a Google Colab platform (in its GPU run to speed up). Preliminary processing of the data took place with the assistance of Python 3.8, Pandas (v1.3.0), and NumPy (v1.21.0). The exploratory and diagnostic visions were made using Matplotlib (v3.4.3) and Seaborn (v0.11.2). When concerning classic ML and cross-validation, we applied Scikit-learn (v0.24) and the imbalanced-learn (v0.8.0) extension of SMOTE oversampling. The DL-based experiments were realized on the basis of TensorFlow (v2.6) and Keras API high level API. Controls Each code cell resided in a single Colab notebook and reproducibility, and contribution tracking were implemented using the aid of Git. The orchestration of all the code was done in one Colab notebook allowing reproducibility and control of versions via git.

3.6 Justification of Choice of Methods

RF was selected due to the ability to handle noisy and high-dimensional features as well as the variance reduction offered by bagging with minimal bias [10], [11]. The scores of feature importance are useful towards interpretation, as well as eliminating manual search, and revealing byte/packet-rate and timing signals [26], [27]. KNN is a low complexity, non-parametric baseline that is competitive when classes form clusters and can be helpful to reveal leakage/overfitting, any improvements over KNN are likely to be substantial [11], [17]. An ANN (MLP) infers non-linear correlations that trees or linear models might not get, e.g. simultaneous spikes in forward/backward packet indicators. it can serve as an accuracy baseline vs. lightweight models, just applying dropout and early stopping to prevent overfitting [14], [24], [25]. Since the data were greatly biased, SMOTE creates realistic minority samples by interpolating realistic samples among records, and thereby bolsters decision boundaries rather than replicating records. Lastly, 5-fold stratified CV maintains class proportions at each fold and therefore gives stable, unbiased estimates of Accuracy/precision/recall/F1 and makes it possible to make apples-to-apples comparisons of RF, KNN and ANN.

3.7 Machine Learning Models

3.7.1 Random Forest

RF is a common ensemble learning procedure which combines the result of several Decision Trees (DTs) to produce greater accuracy and generalization than a single DT. The method is based on the concept of bagging (bootstrap aggregating), in each DT a bootstrap sample (Db) drawn randomly with replacement, of the original training dataset (D) of size (N) is used. It is done with replacement so that data is exposed to each of these trees somewhat differently.

Another level of randomness can be introduced by RF to make DTs even more diverse: when selecting the best feature to split on at each decision node, it is no longer the full set of features that is looked at but instead a randomly selected subset of size m (where m << M, where M is the total number of features). This limits the correlation between DT and aids, the ensemble works well even when the data set comprises several irrelevant or noisy features.

For a classification problem, the prediction from a Random Forest with T trees is:

$$\hat{y} = mode \{h\_t(x) \mid t = 1, 2, \ldots, T\}$$

where *h_t(x)* is the class predicted by the t-th decision tree for the input vector x.

Bootstrap samples with random feature selection decreases the potential of overfitting and renders RF resistant to noise. It can cope with big data, operate on both numerical and categorical data types, and relatively minimal hyperparameter tuning is necessary.

3.7.2 K-Nearest Neighbors

KNN is an instance-based non-parametric learning algorithm that has both classification and regression usage. KNN saves all the training data rather than creating an explicit model as directly done in training, predicting by calculating the similarity of the input samples. The algorithm calculates this distance between each training sample and a particular test sample x by using a distance measure, e.g. Euclidean distance, Manhattan distance or Minkowski distance.

The acts that were closest (neighbors) are then identified. In classification, the predicted class ŷ is determined by a majority vote among the neighbors' classes:

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

where N_k(x) denotes the set of k nearest neighbors of x, and y_i is the class label of neighbor x_i.

Values of k are very important in determining the performance: lower values of k make the model susceptible to noise and high values of k might over-smooth the boundaries of decisions.

3.7.3 Artificial Neural Network

ANN is a theoretical computation model which is based on the structure and the way the human brain works. It is made up of interconnected processing elements called neurons, arranged in layers: an input layer, one or more hidden layers and an output layer. Every neuron has some inputs, performs a weighted sum, does some activation process, and produces an output to the following layer.
Mathematically, for a neuron j, the output a_j is given by:

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

where x_i represents the input features, w_{ij} are the weights connecting input i to neuron j, b_j is the bias term, and ϕ is a non-linear activation function (e.g., Sigmoid, ReLU, Tanh). Connectionist learning is achieved by allowing the network to optimize a loss function by changing the weights and biases within the network to determine the optimum answer to a learning problem. Mostly this is based on an Optimization method such as Gradient Descent coupled with the Backpropagation Algorithm to allow the efficient determination of the gradient of the loss function. In case the output layer is needed in the classification tasks, then the SoftMax activation is commonly used to generate the probability of belonging to a certain class, and linear activation tends to be applied in case of regression. ANN can represent non-linear relationships of complex models and is sufficiently used in computer vision, natural language processing and predictive analytics. But ANN performance is related to network structure, selection of activation functions, optimization method, and availability of training data of adequate quantity.

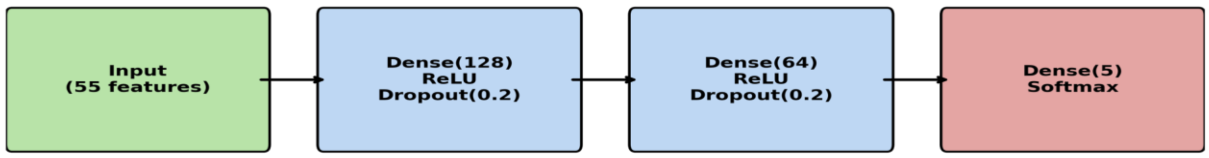
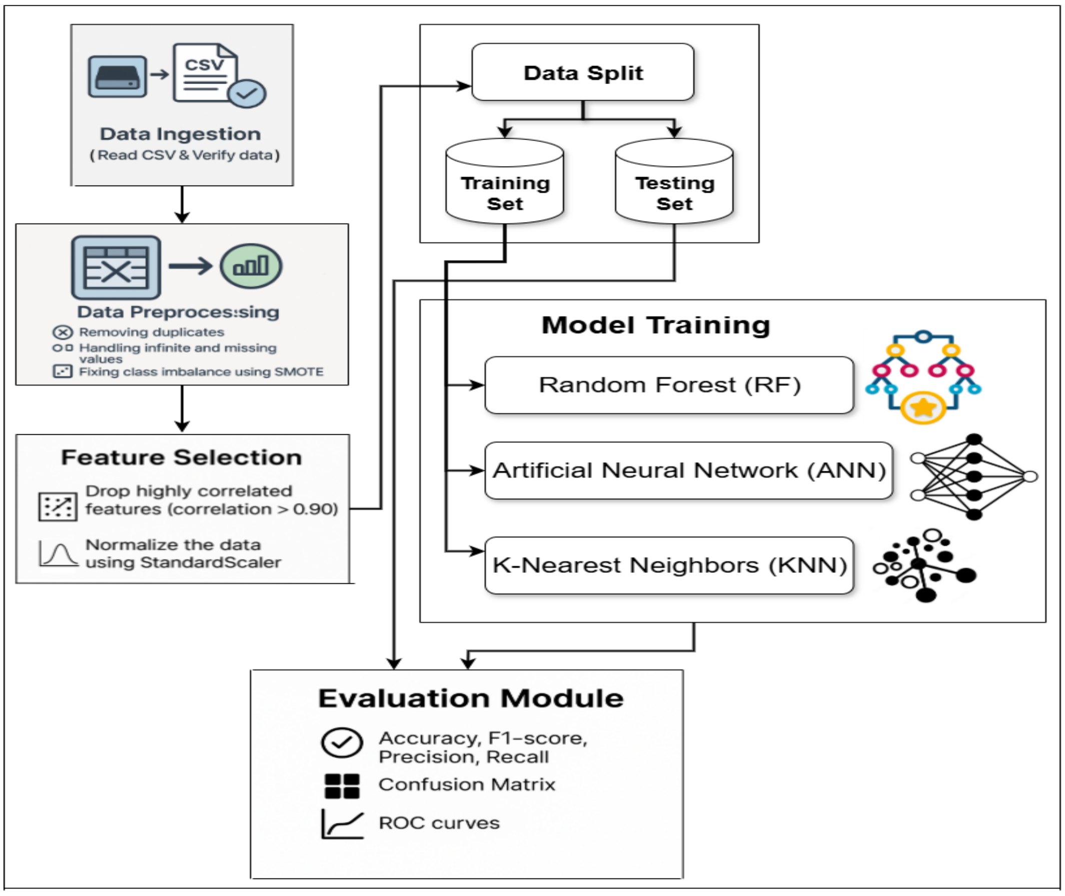


Fig 5. ANN Architecture

Figure 5 shows the ANN architecture used in this paper. It is comprised of an input layer having 55 nodes the chosen features, two hidden layers comprising of 128 and 64 nodes respectively, applying the ReLU activation function and a 20 percent dropout rate in mitigating overfitting. A layer of 5 neurons in the output layer with a Softmax activation of neurons generates the probability of the classes of the five attacks in the IoTID20 data set.

4 Proposed IDS Architecture for IoT Networks

The capabilities of the IDS in IoT environments consist of a multi-stage pipeline that sequentially ingest uncodified raw network traffic and transform the data into data in preparation to modeling and screening the competency of detection by varying machine learning algorithms. There are five modules of architecture, which include: Data Ingestion, Preprocessing, Feature Selection, Data Splitting, and Evaluation.


CSV
Data Ingestion
(Read CSV & Verify data)
Data Preproces:sing
Removing duplicates
Handling infinite and missing values
Fixing class imbalance using SMOTE
Feature Selection
Drop highly correlated features (correlation > 0.90)
Normalize the data using StandardScaler
Data Split
Training Set
Testing Set
Model Training
Random Forest (RF)
Artificial Neural Network (ANN)
K-Nearest Neighbors (KNN)
Evaluation Module
Accuracy, F1-score, Precision, Recall
Confusion Matrix
ROC curves
Fig 6. Overall architecture of proposed IDS framework for IoT.

4.1 Data Ingestion Module

The first and the foremost step in the process would be to upload the raw dataset to Google Drive. The drive could then be mounted straight in the Google Colab environment where you could have direct access to the files. The data is loaded into memory with the help of the pandas.read_csv() function, which obtains the resultant Pandas Data Frame (data). Prior verification is also made to find out whether the data is loaded correctly. This step is to provide a stable pre-environment for further preprocessing.

4.2 Preprocessing Module

Processes of data cleaning are used to eliminate inconsistencies and make things consistent:

- **Infinite values handling:** The columns that contain infinite values are limited at the peak finite value being observed in that column.
- **Missing values imputation**: the mean of the related column to the NaN entry is inserted instead.
- **Feature engineering**: An attribute, Flow KBytes/s,is derived based on Flow_Byts/s.
- **Filtering classes**: The data is confined to a number of main classes: Mirai, Scan, DoS, Normal and MITM ARP Spoofing.

- **Class balancing:** There is a problem of class imbalance where the number of the dominant Mirai class is cut down to 20 of its initial size and all classes were made balanced based on a variant of the SMOTE method, giving the final balanced DataFrame.

4.3 Feature Selection Module

All the numeric features are used in computation of the Pearson correlation matrix to minimize redundancy and maximize efficiency. Those attributes that have correlation coefficients of above 0.90 are eliminated. Pearson correlation matrix may be determined with the following formula.

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

Where:

- r_{xy} = Pearson correlation coefficient between variables *x* and *y*
- x_i, y_i = individual sample points
- $\bar{x}$, $\bar{y}$ = mean of *x* and *y* respectively
- n = number of observations

The rest of features will be standardized with the help of StandardScaler of Scikit-learn. The StandardScaler may be estimated as following:

$$z = (x - \mu) / \sigma$$

Where:

- x = original feature value
- μ = mean of the feature
- σ = standard deviation of the feature
- z = standardized feature value

It makes features on a similar scale, which shows advantages to distance-based models as well as neural networks.

4.4 Data Split Unit

The standardized data is divided into training and validation sets via train test split and sklearn module by a ratio of 70 and 30. Stratification (stratify = y1) is done to produce proportions of each stratum the same in both samples.

- Training data: Xos_train, yos_train
- Testing data**:** Xos_test, yos_test

5 Implementation

The implementation phase of the proposed study suggests the development and the usage of ML models that will be deployed in categorising the forms of IoT networks that are being attacked through sizeable volume of data. The IDS was constructed with the architecture in Design Specification and the research design in the Research Methodology.
This was coded up in Google Colab with python 3.8 and GPU accelerator enabled. Jupyter notebook that contained all the code was version-controlled with Git to replicate and produce collaborative transaction.

Data Preprocessing and feature selection entailed cleaning, addressing the missing / infinite values, balancing with sure orthogonal method understating and pruning features through the correlation-filtering as explained in Research Methodology. Training of the models was done as RF, KNN, ANN on Scikit learn and TensorFlow/Keras. In Research Methodology, all the models were trained using the hyper parameters that were identified during the cross-validation process. The scripts used to evaluate were designed to generate the values of accuracy, precision, recall, F1- score and ROC-AUC, confusion matrices, and ROC curves. These turns out to be the result and were plotted using Matplotlib and Seaborn.

Each classifier or preprocessing step can be removed and replaced without disrupting the rest of the pipeline courtesy of modular implementation. The system has the capacity to internally execute the dataset in proper manner, train models and produce evaluation outputs may be analysed in Evaluation.

6 Evaluation

This section presents the performance evaluation of the three implemented strategies, namely RF, KNN, and ANN, on validation and test datasets. Results are reported in terms of accuracy as the primary evaluation metric, in line with the study's focus. All training, parameter tuning, and preprocessing steps were described in *Research Methodology*.

6.1 Model Training

In the proposed study, three ML models RF, KNN, and ANN, were trained and implemented for IoT Intrusion Detection. GridSearchCV was utilized to tune exhaustively and improve the performance of the predictor, thus preventing the problem of overfitting. In this strategy, every combination of values of the chosen hyperparameters is comprehensively tested using cross-validation and the configuration generating the best performance on the validation set is chosen.

Hyperparameters that were considered in each model are shown below:

Random Forest

- n_estimators: [50, 100, 150]
- max_depth: [None, 10, 20]
- min_samples_split: [2, 4]
- bootstrap: [True, False]

K-Nearest Neighbors

- n_neighbors: [3, 5, 7, 9]
- weights: ['uniform', 'distance']
- metric: ['euclidean', 'manhattan']

Artificial Neural Network

- hidden_units: [64, 128]
- dropout_rate: [0.0, 0.2]
- optimizer: [Adam, RMSprop]
- batch_size: [32, 64]
- epochs: [50, 100]

Optimal parameters of each model on the basis of their application of GridSearchCV are provided as in Table 2 below.

Table 2. Summary table of the three models' key hyperparameters and their selected best values

Model	Best Value(s)
RF	n_estimators=50, max_depth=None, min_samples_split=4, bootstrap=False
KNN	n_neighbors=5, weights='distance'
ANN	hidden_units=128, dropout_rate=0.2, optimizer='adam', batch_size=32, epochs=50

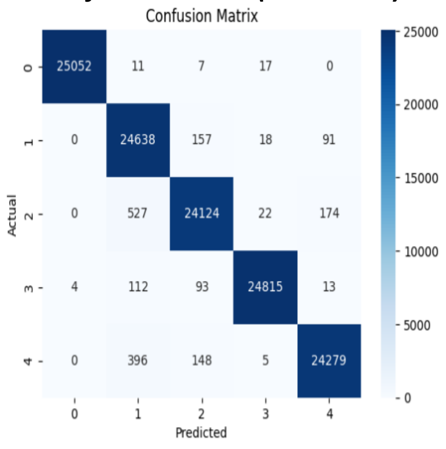
6.2 Random Forest

RF attained high accuracy on validation dataset but performed well on the test dataset. The corresponding precision, recall and F1-scores at the class level were also provided in Table 3, but it is important to note that two scores at the overall level were preferred.

Table 3: RF Performance on Validation and Test Data

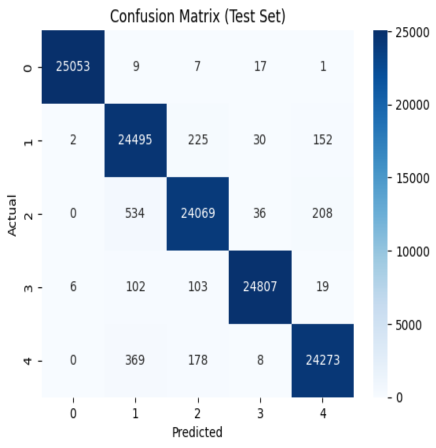
Class	Precision (Val)	Recall (Val)	F1 (Val)	Precision (Test)	Recall (Test)	F1 (Test)
DoS	1.00	1.00	1.00	1.00	1.00	1.00
MITM ARP Spoofing	0.96	0.99	0.97	0.96	0.98	0.97
Mirai	0.98	0.97	0.98	0.98	0.97	0.97
Normal	1.00	0.99	0.99	1.00	0.99	0.99
Scan	0.99	0.98	0.98	0.98	0.98	0.98
Accuracy			**0.9900**			**0.9800**

RF Confusion Matrix (Validation):



(a)

***RF Confusion Matrix (Test)*:**



(b)

Fig 7: RF Confusion Matrix on both (a)Validation Dataset and (b)Test Dataset

RF ROC Curve:

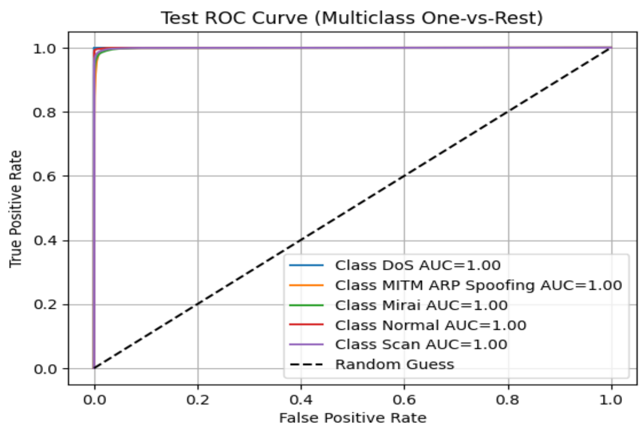


Fig 8: ROC Curve of RF Classifier

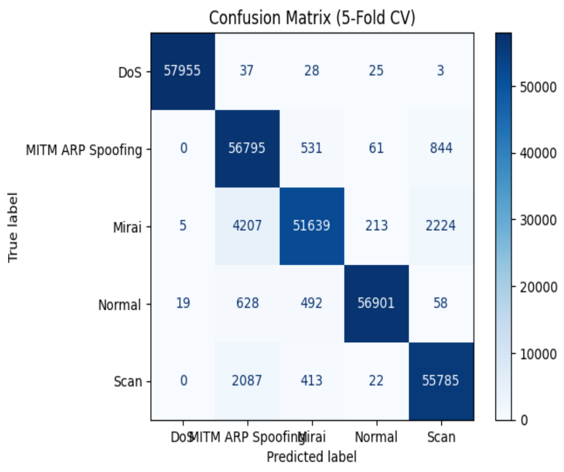
6.3 Artificial Neural Network

ANN recorded a mildly lower accuracy than RF and KNN nonetheless showed great capability of generalization on the test data. The elaborate class-level performance is given in Table 4.

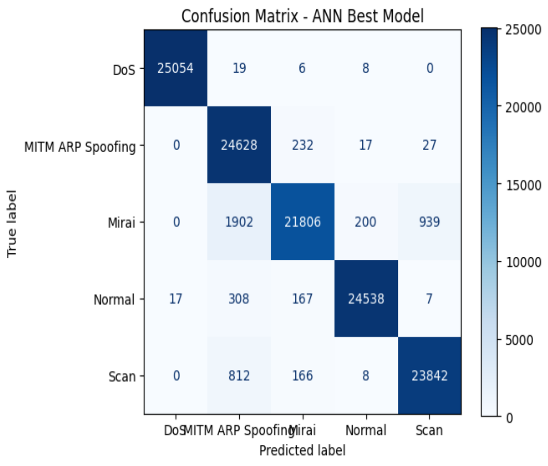
Table 4: ANN Performance on Validation and Test Data

Class	Precision (Val)	Recall (Val)	F1 (Val)	Precision (Test)	Recall (Test)	F1 (Test)
DoS	1.00	1.00	1.00	1.00	1.00	1.00
MITM ARP Spoofing	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	0.956
Accuracy			**0.9691**			**0.9617**

ANN Confusion Matrix (Validation): ***ANN Confusion Matrix (Test):***



True \ Predicted	DoS	MITM ARP Spoofing	Mirai	Normal	Scan
DoS	57955	37	28	25	3
MITM ARP Spoofing	0	56795	531	61	844
Mirai	5	4207	51639	213	2224
Normal	19	628	492	56901	58
Scan	0	2087	413	22	55785



True \ Predicted	DoS	MITM ARP Spoofing	Mirai	Normal	Scan
DoS	25054	19	6	8	0
MITM ARP Spoofing	0	24628	232	17	27
Mirai	0	1902	21806	200	939
Normal	17	308	167	24538	7
Scan	0	812	166	8	23842

(a) (b)

Fig 9: ANN Confusion Matrix on both (a)Validation Dataset and (b)Test Dataset

ANN ROC Curve:

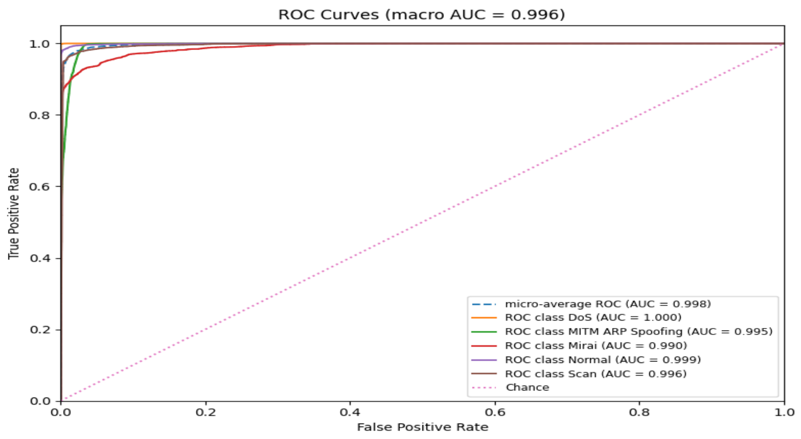


Fig 10: ROC Curve of ANN Classifier

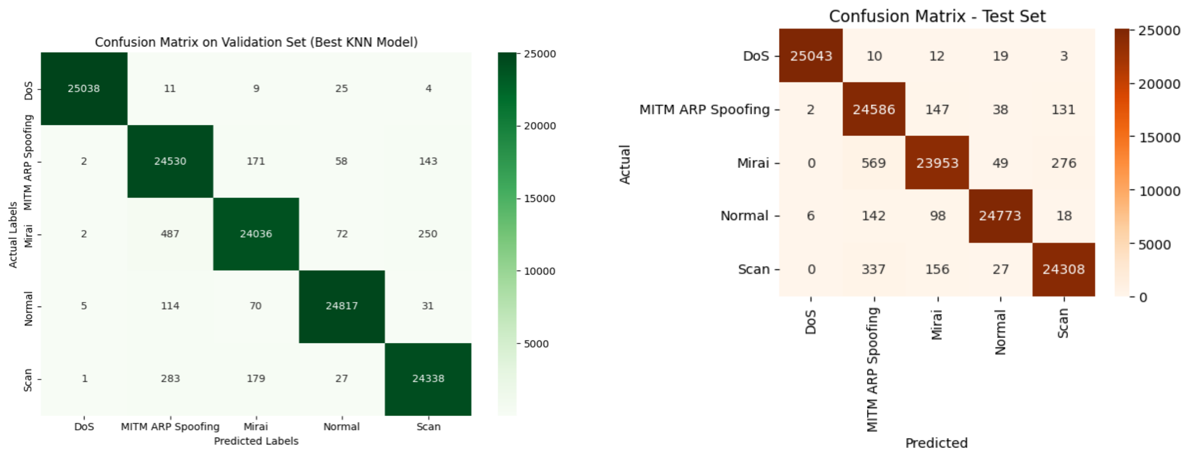
6.4 K-Nearest Neighbors

KNN performed equally well as RF in test accuracy and achieved high accuracy in both data sets but has a slightly lower validation accuracy compared to that of RF. Supporting metrics are given in table 5.

Table 5: KNN Performance on Validation and Test Data

Class	Precision (Val)	Recall (Val)	F1 (Val)	Precision (Test)	Recall (Test)	F1 (Test)
DoS	1.00	1.00	1.00	1.00	1.00	1.00
MITM ARP Spoofing	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**

KNN Confusion Matrix (Validation): ***KNN Confusion Matrix (Test):***



(a) (b)

Fig 11: KNN Confusion Matrix on both (a)Validation Dataset and (b)Test Dataset

KNN ROC Curve:

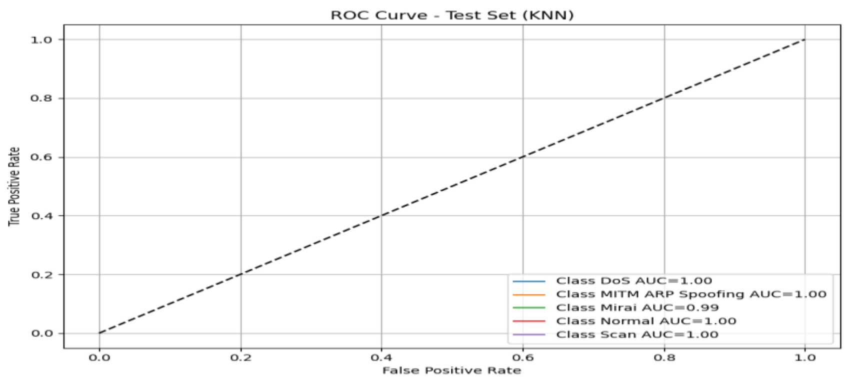


Fig 12: ROC Curve of KNN Classifier

6.5 Comparison with Previous Work

Table 6 includes comparisons of the accuracy of the proposed (RF) model against two other studies that also used the RF on the IoTID20 dataset, but more recently. Our model was 98.00 % which is higher than Al-Hawawreh et al. [26] who was 96.80 % and Kasongo & Sun [27] 97.30 %. Even though the same data set and measure were employed in the three works, methodological discrepancies have an influence on the findings.

Class imbalance was also an issue that Al-Hawawreh et al. did not comprehensively resolve and mentioned it as a restrictive effect of RF. Kasongo & Sun also used unbalanced data, and they concentrated on multiclass detection performance. Intended in contrast, we fitted the dataset with down sampling and SMOTE along with hyperparameter optimization and used RF as part of a modular IDS pipeline. To be fair, there is only one metric that is compared to each other which is accuracy because it was always listed in the studies. The findings indicate that balanced data preparation along with fine-tuning could be a major improvement on RF performance in regard to IoT intrusion detection.

Table 6: Accuracy Comparison with Related Works (IoTID20)

Ref	Study / Approach	Dataset	Accuracy
	Proposed Framework	IoTID20	**0.9800**
[26]	Al-Hawawreh et al. (2022)	IoTID20	0.9680
[27]	Kasongo & Sun (2023)	IoTID20	0.9730

6.6 Discussion

An unbiased comparison of the three strategies indicates that RF indicated the best validation accuracy (0.99) having enjoyed a good performance on the test set (0.98). KNN performed as well as RF in terms of the test accuracy, and a little less in terms of the validation accuracy (0.9820). ANN performed with the lowest accuracy as compared to the other two getting 0.9691 on the validation set and 0.9617 in the test set. Nevertheless, the performance of ANN is still competitive, and it might still be applicable to those cases in which flexibility or certain limiting considerations are the desired aspects. The findings verify that RF surpasses in validation performance efficiency, whereas KNN performed as well as other generalization of data on unknown data.

Table 7 – Model Accuracy Comparison

Model	Accuracy (Val)	Accuracy (Test)	Precision (Val)	Precision (Test)	Recall (Val)	Recall (Test)	F1 (Val)	F1 (Test)
RF	0.9900	0.9800	0.99	0.99	0.99	0.98	0.99	0.98
KNN	0.9820	0.9800	0.98	0.98	0.98	0.98	0.98	0.98
ANN	0.9691	0.9617	0.96	0.96	0.96	0.96	0.96	0.96

7 Conclusion and Future Work

This study aimed at establishing which supervised-learning paradigm, either Random Forests, feed-forward Artificial Neural Networks and distance-weighted K-Nearest Neighbours, would provide the best and deployable results in the flow-based intrusion detection of modern IoT traffic. We started with a balanced multi class dataset that we then curated, we divided it into a same preprocessing pipeline and then applied systematic hyper-parameter tuning and five-fold cross-validation followed by evaluation of each model family on a complete separate test set. The results were conclusive with a Random Forest model with 50 trees and arbitrary depth achieved an F1-score of 0.98 on the test data. It also recorded a validation accuracy of 0.9900 and a mean AUC of 0.9993. This performance outperformed both the KNN and ANN models. The KNN achieved a validation accuracy of 0.9820 and a test accuracy of 0.9800, securing second place. It surpassed the ANN, which achieved a validation accuracy of 0.9691 and a test accuracy of 0.9617. This aligns with previous research that frames ensemble trees as a reliable benchmark on tabular intrusion-detection problems, but this also highlights that there are caveats to this position. Excellent cross-validation results, excessive model size, and the use of a single, homogenous dataset hinder the extent of external validity of the study since there is no temporal hold-out testing or runtime profiling to report resilience to concept drift or fitness in edge-deployment targets. However, it is proven that, in controlled settings, tree-based ensembles are capable of achieve state-of-the-art accuracy with minimal overhead in terms of feature-engineering.

Our future validation procedure will additionally involve carrying out cross-dataset validation to ensure that our results can be more generalizable and robust. It will imply the testing of the suggested models on diverse IoT traffic data that will be accumulated under the visions of various network settings, the types of devices, and the time. In this way, we will have the potential to garner a more informed appreciation of the capacity of the models to withstand alterations in the data overtime, conform to various traffic conditions, and scale on various IoT infrastructures. We will also test them to see how they perform in various operating conditions to gauge whether they can be used in real-life applications under edge-computing and limited resource conditions. This will assist in overcoming the limitations that exists at present where only one dataset, and a uniform one, will be used and it will enhance the external validity of our results.

References

[1] H. Rajadurai and U. D. Gandhi, "A stacked ensemble learning model for intrusion detection in wireless network," Neural Computing and Applications, pp.1–9, 2020.
[2] Kurniabudi, D. Stiawan, Darmawijoyo, M. Y. Bin Idris, A. M. Bamhdi, and R. Budiarto, "CICIDS-2017 Dataset Feature Analysis With Information Gain for Anomaly Detection," IEEE Access, vol. 8, pp. 132 911–132 921, 2020.
[3] Statista Research Department. "Internet of Things (IoT) connected devices installed base worldwide from 2019 to 2025 (in billions)." Statista, updated February 2025.

[4] Verizon. “2025 Data Breach Investigations Report (DBIR)”. Verizon Enterprise Solutions, May 2025, pp. 28–32.
[5] IBM Security & Ponemon Institute. “Cost of a Data Breach Report 2025”. IBM Corporation, Jul 2025, pp. 14–17.
[6] M. Airowaily, F. Alenezi, and Z. Lu, "Effectiveness of machine learning based intrusion detection systems," in International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage, 2019,pp. 277–288.
[7] S. Shafieian and M. Zulkernine, "Multi-layer stacking ensemble learners for low footprint network intrusion detection," Complex & Intelligent Systems, pp. 1–13, 2022.
[8] Z. Chiba, N. Abghour, K. Moussaid, O. Lifandali, and R. Kinta, "Review of recent intrusion detection systems and intrusion prevention systems in IoT networks," in 2022 International Conference on Software, Telecommunications and Computer Networks (SoftCOM), 2022, pp. 1–6.
[9] H. Hellström, J. M. B. da Silva Jr., M. M. Amiri, M. Chen, V. Fodor, and H. V. Poor, “Wireless for Machine Learning: A Survey,” *Foundations and Trends in Signal Processing*, vol. 16, no. 3–4, pp. 290–399, 2022.
[10] ouhail Meftah, Tajjeeddine Rachidi, and Nasser Assem. Network based intrusion detection using the UNSW-NB15 dataset. International Journal of Computing and Digital Systems, 8(5):478–487, 2019.
[11] Mohammed Alrowaily, Freeh Alenezi, and Zhuo Lu. Effectiveness of machine learning
based intrusion detection systems. In International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage, pages 277–288. Springer,2019.
[12] ayu Adhi Tama, Marco Comuzzi, and Kyung-Hyune Rhee. TSE-IDS: A two-stage classifier ensemble for intelligent anomaly-based intrusion detection system. IEEE access, 7:94497–94507, 2019.
[13] Hoang Ngoc Thanh and Tran Van Lang. Evaluating Effectiveness of Ensemble Classifiers When Detecting Fuzzers Attacks on the Unsw-Nb15 Dataset. Journal of Computer Science and Cybernetics, 36(2):173–185, 2020.
[14] Azriel Henry, Sunil Gautam, Samrat Khanna, Khaled Rabie, Thokozani Shongwe, Pronaya Bhattacharya, Bhisham Sharma, and Subrata Chowdhury. Composition of hybrid deep learning model and feature optimization for intrusion detection system. Sensors, 23(2):890, 2023.
[15] Tianbo Gu, Zheng Fang, Allaukik Abhishek, Hao Fu, Pengfei Hu, and Prasant Mohapatra. Iotgaze: Iot security enforcement via wireless context analysis. pages 884–893.IEEE INFOCOM 2020 - IEEE Conference on Computer Communications, 2020.
[16] Mingzhe Chen, Deniz G¨und¨uz, Kaibin Huang, Walid Saad, Mehdi Bennis, Aneta Vulgarakis Feljan, and H. Vincent Poor. Distributed learning in wireless networks: Recent progress and future challenges. IEEE Journal on Selected Areas in Communications,39(12):3579–3605, 2021.
[17] Yogendra Kumar and Siba K Udgata. Machine learning model for iot-edge device based water quality monitoring. pages 1–6. INFOCOM, 2022.
[18] Zouhair Chiba, Noreddine Abghour, Khalid Moussaid, Oumaima Lifandali, and Rachid Kinta. Review of recent intrusion detection systems and intrusion prevention systems in iot networks. In 2022 International Conference on Software, Telecommunications and Computer Networks (SoftCOM), pages 1–6, 2022.
[19] Miao Xie, Jiankun Hu, Xinghuo Yu, and Elizabeth Chang. Evaluating host-based anomaly detection systems: Application of the frequency-based algorithms to ADFA-LD. In International Conference on Network and System Security, pages 542–549. Springer, 2015.
[20] Basant Subba, Santosh Biswas, and Sushata Karmakar. Host based intrusion detection system using frequency analysis of n-gram terms. In TENCON 2017 - 2017 IEEE Region 10 Conference, pages 2006–2011, 2017.
[21] Bhavesh Borisaniya, Dhiren Patel, et al. Evaluation of modified vector space representation using adfa-ld and adfa-wd datasets. Journal of Information Security, 6(03):250, 2015.

[22] Nam Nhat Tran, Ruhul Sarker, and Jiankun Hu. An approach for host-based intrusion detection system design using convolutional neural network. In International conference on mobile networks and management, pages 116–126. Springer, 2017.
[23] Dainius ˇCeponis and Nikolaj Goranin. Evaluation of deep learning methods efficiency for malicious and benign system calls classification on the awsctd. Security and Communication Networks, 2019, 2019.
[24] Belal Sudqi Khater, Ainuddin Wahid Bin Abdul Wahab, Mohd Yamani Idna Bin Idris, Mohammed Abdulla Hussain, and Ashraf Ahmed Ibrahim. A lightweight perceptron-based intrusion detection system for fog computing. Applied Sciences, 9(1):178, 2019.
[25] Oluwagbemiga Ajayi and Aryya Gangopadhyay. DAHID: Domain Adaptive Host-based Intrusion Detection. In 2021 IEEE International Conference on Cyber Security and Resilience, pages 467–472, 2021.
[26] M. Al-Hawawreh, N. Moustafa, and E. Sitnikova, "A hybrid deep learning approach for intrusion detection in IoT networks," *Computers & Security*, vol. 120, p. 102934, 2022.
[27] G. Kasongo and Y. Sun, "Ensemble-based intrusion detection for IoT networks using IoTID20," *IEEE Internet of Things Journal*, 2023.