Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 4, Issue 6, 2026

ENSEMBLE MACHINE LEARNING: MULTI-CLASS NETWORK ATTACK CLASSIFICATION USING CIC-IDS DATASET WITH HYPER PARAMETER TUNING AND COMPARATIVE ANALYSIS

[1]Sidra Malik, [2]Areeba Amer
[1]Department of Computer Science, UMT
[2]Department of Computer Science, UMT
[1]26010279001@skt.umt.edu.pk, [2]26010279006@skt.umt.edu.pk

DOI:- https://doi.org/10.5281/zenodo.21818660

Keywords
Network Intrusion Detection, CIC-IDS Dataset, Hyperparameter Tuning, Multi-class Classification, Cybersecurity

Article History
Received: 19 May, 2026
Accepted: 22 June, 2026
Published: 24 June, 2026

Corresponding Author: *

Abstract
Attacks like Distributed Denial-of-Service (DDoS), Botnets, network scanning etc. remain a big challenge to the modern networked systems. Correct identification of these attacks is key to maintaining network security and to making sure that the incident can be responded to in a timely fashion. Most previous work has used binary classification to identify whether network traffic is malicious or not, but in this research multi-class classification is used. The proposed model is trained on the merged CICIDS 2017 dataset and classifies the network traffic into four classes, i.e., BENIGN, DDoS, Bot, and PortScan, with 184,532 preprocessed records and 78 traffic features. In order to assess classification efficacy, five distinct machine learning algorithms were implemented and evaluated under identical experimental parameters: LR, DT, RF, XGBoost, and LightGBM. The Bot class exhibited substantial underrepresentation within the dataset; consequently, class imbalance was addressed through the application of SMOTE. Additionally, RandomizedSearchCV was employed to fine-tune the hyperparameters of both RF and XGBoost, thereby enhancing their capacity to generalize effectively to previously unseen data. The trial results showed that the tuned XGBoost model exceeded all evaluated classifiers with 99.94% accuracy, precision, recall, and F1-score. To evaluate each model's effectiveness, particularly on minority attack classes, a comprehensive class-wise evaluation and confusion matrix analysis were conducted in addition to overall performance measures. The results show the promise of advanced ensemble learning approaches for network intrusion detection and provide a comprehensive evaluation of five machine learning algorithms, including LightGBM, on the CICIDS 2017 combined dataset.

I. Introduction

Global connection and operational efficiency have improved due to the growing reliance on cloud-enabled services and internet-based applications. Contemporary organizations face an elevated spectrum of cybersecurity vulnerabilities. Among the most consequential of these menaces are Distributed Denial-of-Service (DDoS) attacks, which leverage disproportionate traffic volumes to exhaust target system resources, culminating in either degraded functionality or complete service disruption [1]. Over the last decade, the number, size and complexity of DDoS attacks have increased substantially. Critical services from banking and healthcare to government infrastructure are moving online [2], and never before has the stakes for a successful DDoS attack been higher.

Besides volumetric DDoS attacks, the modern network environment is threatened by a diverse ecosystem of threats, such as botnet-driven traffic (Bot) and automated network reconnaissance (PortScan). Each of these attack types has different behavioural signature, traffic pattern and damage it causes [3]. A DDoS attack is a drain of network resources . Bot traffic is a compromised host talking to command-and-control servers . PortScan traffic is an attacker actively looking for open vulnerabilities. Such differences suggest that the ability to discriminate between attack types rather than simply distinguishing between attack and benign is important for enabling appropriate and proportionate security responses [4].

Machine learning (ML) is one of the efficient solutions for detecting the intrusions in a network, as it can be performed automatically in the analysis of complex traffic information and identification of malicious activities. They are capable of adapting to the ever-changing cyber threat space and adaptation from observed network traffic using machine learning techniques [5]. Random Forest (RF) and Extreme Gradient Boosting (XGBoost) are two of the ensemble models, which have been much in demand due to their impressive categorization capabilities. The strategies often enhance the accuracy, robustness and generalization ability in a network traffic analysis problem by fusing the decisions of multiple learners [6]. The Light Gradient Boosting Machine (LightGBM) has been gaining popularity recently on account of its high efficiency and comparable accuracy as a training alternative that gives shorter training durations.

Some standard sets of data are available to aid in research in this area. Realistic network traffic captures containing a variety of attack and benign traffic have also been provided by the Canadian Institute for Cybersecurity for the use of intrusion detection research [16] in the form of the CIC-IDS data set. The older datasets have problems with outdated attack patterns and traffic generation that is synthetic. The considered attack scenarios are modern attacks supplied by CIC-IDS [6] and are represented by 78 features extracted by CICFlowMeter [7] that are based on flow.

While there exists a considerable body of published literature, it has various important limitations. In this note, we first outline the drawbacks of these studies: most of them treat the detection task as a binary classification problem (attack vs benign), which is not typical of the threat posed in the real world; they also lack the information needed for an operational response [8]. Second, Imbalanced datasets are used in many works and corrective techniques aren't applied to them, leading to classifiers that are biased toward the majority class and fail to reliably detect the minority attack types [15]. Thirdly, multi-classifier evaluations are still rare, particularly when newer variants of gradient boosting are combined with the older ones as in the case of LightGBM and other classic classifiers [9]. Lastly, hyperparameter tuning with systematic approach is not systemically performed, and before and after results are rarely explicitly reported, which may undermine the confidence in the reported results [10].

This research is intended to take care of all above mentioned limitations in a unified experimental approach. It would examine different ML classifiers under comparable experimental settings using the CIC-IDS combined dataset, which is a rich, realistic, publicly available dataset consisting of 184,532 instances in four classes of traffic (BENIGN, DDoS, Bot, PortScan), and 78 feature dimensions. We use SMOTE method for data balancing, with a RandomizedSearchCV approach in order to search thoroughly for the best values of

the hyperparameters of the best classifiers [1,15] and report the comprehensive per-class metrics such as confusion matrices for all classifiers. The results clearly indicated that the Tuned XGBoost model outperformed all other classifiers with 99.94% accuracy, precision, recall & F1-score. The current work builds upon the state-of-the-art by performing a first comprehensive comparison between five classifiers on either the CIC-IDS or the combined dataset properly tuned with LightGBM and assessed in a multi-class setting as an extension of the methodology introduced in [1].

II. Problem Statement

While significant advances have been made in the use of ML techniques in NID, there are still numerous major challenges in this area. Most of the current studies focus on binary classification methods, which classify network traffic into two classes: safe and harmful. It does not parallel the complexity of real world cases which contain multiple attack types such as DDoS, Bot, and PortScan simultaneously, demanding specific responses [1, 10]. Furthermore, network intrusion datasets typically are unpractically skewed with the major class representing the bulk of traffic and the minority classes representing the attack traffic, which can cause the machine learning models to be biased towards the major class and fail to accurately predict attack traffic (minor class) [15]. Another restriction is, is the traditional ML techniques and the more sophisticated ensemble techniques proposed in literature, not been thoroughly compared in the same experimental frameworks. Receiving the best way to run multi-class intrusion detection is difficult with the recent studies, most of which test a limited number of classifiers, and some which ignore modern boosting classifiers like LightGBM [9]. Furthermore, many researchers do not tune their classifiers systematically by altering the hyperparameters, so that the classifiers' performance may not be optimal [1]. Thirdly, the traditional performance metric used for most networks is simply accuracy [8], whose performance may be misleading when dealing with highly imbalanced datasets, in which minority attack classes might not be detected because of good overall accuracy. Hence, a multi-class intrusion detection system which is capable of tackling the class imbalance, systematic hyper-parameter space optimization, comparison of multiple machine learning classifiers with each other, and detailed class-wise performance evaluation on a practical benchmark dataset like CIC-IDS2017 [16] is needed.

III. Literature Review

This section reviews 22 important studies on DDoS and network attack detection and classifies them according to technique. We will review each study, considering the data they used, the approach they took, their main findings, and most importantly, the limitations that sparked our current research. By reviewing these studies and their limitations, we can better understand what has been done and what still needs to be explored in the area of network attack detection.

A. Classical Machine Learning Approaches

The researchers have proposed a multi-class DDoS attack classification framework based on Random Forest and XGBoost, which has an accuracy of 89% and 90% respectively, for nine attack classes. However, the dataset was highly imbalanced, without any oversampling technique and neither hyperparameter tuning nor explainable AI techniques were considered, resulting in poor performance on minority classes [2]. A semi-supervised learning-based approach using clustering technique was used and resulted in 96.66% accuracy in DDoS detection but was limited to a binary classification problem and was not applied feature importance analysis or ensemble learning techniques or hyperparameter optimization [10]. The traditional machine learning algorithms like SVM, Naïve Bayes, KNN and Decision Tree were compared at an approximate accuracy of 95%. Nevertheless, the authors did not consider modern boosting algorithms like LightGBM and XGBoost and there was no use of class imbalance handling techniques [13]. An overview of machine learning techniques for DDoS detection in SDN showed that the number of RF, SVM, DT and ANN modeling algorithms in SDN has been growing, and that there is a need for standardized assessment methods and effective data balance approaches [11]. The researchers took an approach of adding a dynamic feature selection method and one Decision Tree classifier giving accuracy of

99.98% in the CIC-IDS-2018 dataset, but the evaluation adopted in their work was limited with respect to two other aspects, namely the evaluation method involved single classifier instead of ensemble-based evaluation and the lack of hyperparameter optimization [4].

B. Gradient Boosting and Ensemble Methods

Investigators designed an ML framework on the ID of a DDoS attack, using data obtained from Kaggle with 27 features. Implementing it the authors combined several algorithms such as Logistic Regression, Decision Tree, Random Forest and XGBoost, used SMOTE-based methods to combat the data imbalance, used Random Search to optimize hyperparameters and two interpretability mechanisms – SHAP and LIME – which achieved a classification accuracy rate of 97%. The analysis found that the features that have the biggest impact on classification performance are LAST_PKT_RECEIVED, PKT_DELAY and BYTE_RATE. However, the study was done on a smaller dataset and with a limited scope of attack classification motivating the extension of the approach to a larger and more diverse multi-class intrusion detection dataset [1]. Tree-based classifiers have also been used for cloud-based DDoS attack detection with accuracy of ~96%. However, the evaluation was limited to a cloud-specific binary classification task, which restricts its generalizability, and modern gradient boosting methods were not included in the comparison [14]. This study propose a statistical rule-based approach to detecting new types of DDoS attacks in Software Defined Networking (SDN) environment. At the time it was effective, but it was based on manually created rules and was not as flexible as modern machine learning techniques. For the ever-changing landscape of cyber threats, this shows the advantages of data-driven classification approaches [12].

.

C. Deep Learning Methods

The feed-forward Deep Neural Network (DNN) was utilized in the application layer for detecting DDoS attacks and resulted in an accuracy of almost 100%. However, the lack of cross-validation entails the risk of overfitting, and the model was tested on a small set of inputs and was not compared to traditional machine learning methods [7]. Autoencoder-based deep learning framework was evaluated on NSL-KDD, CIC-DDoS2019 and CICIDS2017 datasets and obtained 96.08% accuracy. The study was limited to a single deep learning architecture and was only evaluated on offline datasets without comparison to machine learning methods or using data balancing techniques, despite promising results [6]. The Multi-Layer Perceptron (MLP) model achieved 98.99% accuracy on the CTU-13 dataset, with a low false positive rate. However, it could not outperform traditional machine learning classifiers, did not take into account the class imbalance, and did not report detailed per-class performance metrics [8]. A hybrid deep learning framework with DCGAN, ResNet50 and AlexNet obtained accuracy of 99.37% on CIC-IDS2019 dataset. While the approach achieved high detection performance, it was computationally expensive, and lacked explainability mechanisms. These impeded its application in the real world [5]. The accuracy of detecting an IoT-DoS/DDoS attack was 99.99% when a CNN based ResNet was used, whereas the average precision over the classes was only 87%, with a high number of false positives. Furthermore, the results and dataset are specific to IoT, making it impractical to extrapolate the results for the more general case of network intrusion detection [9].

D. Explainable AI in Security

The framework proposed by XAI that relies on ASHAP mechanisms proved highly effective at detecting attack events in IoT environments, achieving a high detection rate of 98%, and providing feature-level explanations of generated algorithmic decisions. The research produced proof that explainability methodologies can drive a better understanding and confidence of the practitioner in the validity of the interpretation of machine learning. Despite these results, the empirical assessment was limited to binary classification; moreover, the lack of comparison against standardized explainability frameworks e.g., LIME, limits a thorough analysis of the performance of XAI in methodological differences [3].

E. Recent Advances in Ensemble Based Intrusion Detection with Explainability

In this investigation, an explainable intrusion detection system that uses XGBoost and XAI techniques for IoMT environments was set. The research showed that combining both interpretability mechanisms and machine learning techniques can help to achieve both high detection rates and interpretability. However, the scope was limited to health-care specific applications leaving limited generalizability to other network ID applications. What was found after analysing SHAP and LIME methodology for intrusion detection system was an explainability approach that helps the security analyst to better understand the model decisions and helps in carrying out forensic exam methodology. This investigation's focus, however, was mainly on the explainability aspects and lacked the extensive comparative study of various ensemble classifiers at uniform experimental factors. The authors introduce a LIME-based methodology for creating explainable AI, aiming to enhance the explainability and trustworthiness of intrusion detection systems. The framework most successfully helped to make the predictions of machine learning more readily understood, but limited comparisons of the performance characteristics of the classifiers in the evaluation were performed [20]. An XAI/machine-learning-based intrusion detection framework was introduced, which was observed as an operational cybersecurity environment and in which the need for transparency and efficiency in security models was apparent. But this study ignored the extreme class imbalance problem and the in-depth analysis of the study was restricted to the small number of attack classes that really matter in practical IDSs [21].

F. *Comparative Analysis*

Table 1: *Comparative Analysis of Related Works*

REF	Data sett	Methods	Best Acc.	Key Limitations
[1]	DDoS Network Logs (Kaggle)	LR,DT,RF,XGBoost+SMOTE+SHAP/ LIME	97%	Binary only; 27 features; 4 attack types; small dataset
[2]	Custom DDoS Dataset	RF, XGBoost	90%	No data balancing; poor minority class; no XAI
[3]	IoT Network Dataset	ML + SHAP	98%	Binary only; LIME not compared; IoT-specific
[4]	CIC-IDS-2018	DT + Feature Selection	99.98%	Single classifier; no ensemble; no tuning
[5]	CIC-IDS2019, UNSW-NB15	DCGAN+ResNet+AlexNet (DL)	99.37%	High cost; no ML comparison; no XAI
[6]	NSL-KDD, CIC-DDoS2019	DL Autoencoder	96.08%	Offline only; single DL model; no balancing
[7]	Application-Level Traffic	Feed-Forward DNN	~100%	No cross-validation; constrained input; binary

[8]	CTU-13	MLP (Deep Learning)	98.99%	Single DL model; no ML comparison; no balancing
[9]	Custom IoT Dataset	ResNet CNN	99.99%	Avg. precision only 87%; IoT-specific; no XAI
[10]	Custom DDoS Dataset	Semi-Supervised Clustering	96.66%	Binary only; no ensemble; no tuning or XAI
[11]	Multiple SDN Datasets	ML Survey (RF,SVM,DT,ANN)	~99%	Survey; no implementation; no new dataset
[12]	SDN Traffic Logs Benchmark Network	Statistical + Rule-based SVM, NB, KNN, DT	N/A	Pre-ML baseline; no learned features; not scalable
[13]	SDN Traffic Logs Benchmark Network	SVM, NB, KNN, DT	~95%	Traditional ML only; no gradient boosting; no SMOTE
[14]	Cloud DDoS Dataset	DT, RF Tree-based	~96%	Cloud-specific; binary; no hyperparameter tuning
[18]	CIC-IDS 2017	XGBoost	99.2%	Binary classification only: no balancing class
[19]	CIC-IDS 2018	LightGBM,CatBoost	98.7%	No minority-class analysis
[20]	CIC-IDS 2017	RF+SMOTE	98.9%	Limited classifier comparison
[21]	CIC-IDS 2018	XG-BOOST+SHAP	99.1%	Single classifier only
[22]	CIC-IDS combined	RF, XGBOOST, LightGBM, DL	99.3%	NO detailed per class metrics
Prop.	CIC-IDS Combined (184,532 samples, 78 features)	LR,DT,RF,XGBoost,LightGBM+SMOTE+Tuning	99.94%	Multi-class; 5-classifier comparison; per-class metrics; LightGBM

G. Gaps in Research

While considerable efforts have been devoted to machine learning-based intrusion detection, there remains some key unsolved research questions to address. It can be noticed that the majority of present studies are confined in binary classification and just a few studies concentrate on multi-class attack classification in the same experimental framework adopting state-of-the-art ensemble classification approaches. Some recent works have been done on explainable AI techniques (e.g., SHAP and LIME), which however lacks support in integrating within a holistic multi-class intrusion detection system. Moreover, the literature barely considers the issue of class imbalance or explicitly evaluates hyper-parameters according to a systematical tuning or provides detailed performance analysis for each class.

IV. Methodology

All the experimental pipeline from the collection and elaboration of the dataset to the training of the classifiers, their hyperparameter tuning, and testing is explained. These are displayed in the general framework shown in Figure 1.

A. Dataset Description

In this study, the CIC-IDS dataset was used, which was provided by the CIC intrusion detection benchmark (CIC-IDS-2017) [16]. The data consist of 961,678 descriptions of network flow metrics collected using 79 different features defined by CICFlowMeter. The analyzed characteristics include the magnitude of the packet, temporal characteristics, intervals between consecutive packets, flag-based metrics and traffic throughput rates:

- **BENIGN:** Normal network traffic the leading class
- **DDoS:** Distributed Denial-of-Service attack traffic
- **PortScan:** Automated network port scanning activity
- **Bot:** Botnet command-and-control communication the minority class

This aspect of homogeneity is also crucial for methods such as stratified random sampling, which were employed to ensure a sample of 20% of the original data set (192,336 records before cleaning) without compromising statistical accuracy.

B. Data Preprocessing

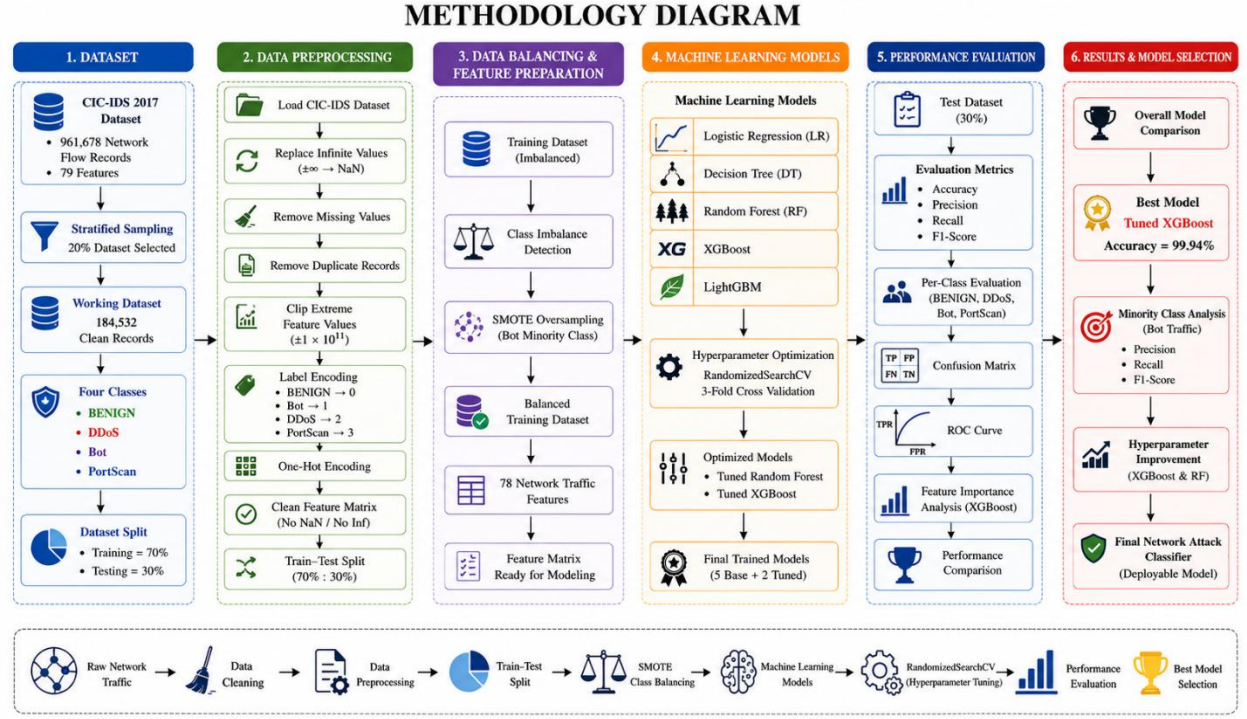
A comprehensive preparation pipeline was created before training the model to ensure consistency and enhance the quality of the data. Special care was taken in the treatment of infinite and missing values to make the data consistent. The infinite values (for example, np.inf and -np.inf) were first made NaN with the df.replace() method and then eliminated by the df.dropna() method. This division might give infinite answers due to the characteristics of the network traffic, therefore, the need for this operation. Besides, unusual values of features were clipped below and above 1x10^11 and with use of clip function, the problem arising from overflow was avoided in the training of the model. This is especially valuable for algorithms, such as gradient boosting methods that can be affected by outliers.

To minimize potential data leakage and redundancy, only 79 characteristics were retained from the dataset, which excludes any if data was duplicated.To reduce any risk of data leakage, duplicate records were removed from the dataset using the "df.drop_duplicates() function" and 184,532 records with 79 characteristics were retained. LabelEncoder was then used for Label column from a scikit-learn library so that the labels BENIGN were represented as 0, Bot was represented as 1, DDoS was represented as 2, and PortScan was represented as 3. The final feature matrix created was of shape size (184,532 x 78) after one-hot encoding categorical features. The feature matrix after processing was verified for any NaN or infinity values in the matrix, ensuring that the data is clean and ready to proceed with the next splitting stage. The data were then divided into three parts: a training set (129172 samples) and a test set (55360 samples) using stratified random sampling, keeping the proportions of the classes as intact as possible. Before class balancing, the distribution of the training set was very unevenly weighted (92188 samples—71.4%—were BENIGN, 18824 samples—14.6%—were PortScan, 17897—13.9%—were DDoS, and only 263—0.2%—were Bot).

Due to the extreme class imbalance (92,188 BENIGN to 263 Bot), the imbalanced-learn SMOTE library was used to address it. There are

four types of network communications in the data set. SMOTE does not create actual examples, but rather creates fake examples of the minority class between existing examples in the feature space. This helps to have a balanced representation of all the classes, reduces the chance of overfitting and increases the ability of classifiers to represent decision boundaries that belong to less represented classes. The test set was intentionally kept to cover class proportioning observed in the original set and the implementation of SMOTE was only applied to the training data, thereby ensuring that information leakage is avoided between the training and test sets.

Figure 1: Methodology Diagram of the Proposed Multi-Class Network Attack Classification Framework



C. Train-Test Split

Our data was split into a training set and a testing set. The training part makes up 70% of the data, while the testing part makes up 30%. We used a special technique to make sure that the classes were represented in the same way in both parts. This gave us 129,172 samples for training and 55,360 samples for testing. Before we balanced the classes, the training set had a certain distribution of classes.

- BENIGN (Label 0): 92,188 samples (71.4%)
- PortScan (Label 3): 18,824 samples (14.6%)
- DDoS (Label 2): 17,897 samples (13.9%)
- Bot (Label 1): 263 samples (0.2%) severely underrepresented

D. Data Balancing with SMOTE

The severe class imbalance particularly the Bot class with only 263 training instances versus 92,188 BENIGN instances (a ratio of approximately 1:350) was addressed using the Synthetic Minority Oversampling Technique (SMOTE) from the imbalanced-learn library [19]. Rather than replicating original cases, SMOTE (Synthetic Minority Over-sampling Technique) creates synthetic samples for minority classes by interpolating between existing data points in feature space. As a result, there is less chance of overfitting and classifiers may learn decision boundaries for minority classes more effectively thanks to a more balanced training distribution. SMOTE was applied exclusively to the training set to prevent data leakage into the test set [15].

E. Classification Models

The five ML models were trained and evaluated.

Logistic Regression (LR): A linear probabilistic classifier employing the logistic function to characterize the association between input features and predicted class probabilities, serving as a foundational reference point for evaluating

performance against non-linear ensemble approaches.

Decision Tree (DT): Decision Tree (DT): A non-parametric classifier that recursively subdivides the feature space based on information gain or Gini impurity metrics, offering considerable interpretability through its hierarchical structure. However, susceptibility to overfitting presents a notable limitation unless appropriate regularization techniques such as pruning or depth restriction are implemented.

Random Forest (RF): The RF algorithm constitutes an ensemble-based approach that leverages the aggregation of numerous decision trees. Individual trees are constructed utilizing bootstrap samples obtained through random sampling with replacement from the original dataset. During the partitioning process at each node, only a randomly selected subset of features is evaluated for consideration. The ultimate prediction is determined through a voting mechanism wherein all constituent trees participate, with the final classification outcome reflecting the predominant vote. This ensemble technique demonstrates superior classification performance and exhibits enhanced generalization capabilities compared to standalone decision tree models, thereby mitigating the propensity toward overfitting.

XGBoost (Extreme Gradient Boosting): GBoost constitutes a gradient boosting framework that constructs decision trees in a sequential manner, wherein each successive tree addresses the residual errors produced by its predecessors [17]. The framework incorporates L1 and L2 regularization mechanisms to mitigate overfitting risks and facilitates parallel computation to optimize computational performance. For the purposes of this investigation, the following default hyperparameters were employed: n_estimators=100, max_depth=6, and learning_rate=0.3.

LightGBM (Light Gradient Boosting Machine): LightGradient Boosting Machine, or LightGBM, represents a prominent machine learning tool within the gradient boosting framework category, distinguished by its design orientation toward computational efficiency. This efficiency is achieved through several algorithmic innovations, including leaf-wise rather than level-wise tree expansion and methodologies such as Gradient-based One-Side Sampling and Exclusive Feature Bundling. Consequently, LightGBM demonstrates the capacity to process extensive datasets substantially faster than comparable frameworks including XGBoost, while maintaining competitive predictive accuracy. Furthermore, this research represents the inaugural application of LightGBM to this particular dataset, thereby contributing novel empirical findings to the existing literature.

F. Hyper parameter Tuning

Hyperparameter tuning was done for XGBoost and Random Forest by using scikit-learn's RandomizedSearchCV with 3-fold cross-validation. Instead of exhaustive GridSearchCV, RandomizedSearchCV was selected because it is less computationally intensive; instead of trying all the possible combinations in the hyperparameter space, it randomly samples from it, which makes it suitable for large datasets and limited computational resources (Google Colab environment).

XGBoost search space: n_estimators ∈ {100, 200}, max_depth ∈ {4, 6}, learning_rate ∈ {0.05, 0.1}. Best parameters found: n_estimators=100, max_depth=6, learning_rate=0.1.

Random Forest search space: n_estimators ∈ {100, 200}, max_depth ∈ {10, 15, None}, min_samples_split ∈ {2, 5}. Best parameters found: n_estimators=100, max_depth=15, min_samples_split=5.

G. Evaluation Metrics

We evaluatethe performance of all the models on a special test set comprising 55,360 samples. We performed this by using some particular measures of how good they were.

- **Accuracy:** This is how often we get it right out of all the things we try to figure out.

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

- **Weighted Precision:** Average per-class precision, the ratio of true positives among predicted positives.

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

- **Recall (weighted):** Weighted average of per-class recall, the ratio of correctly identified positives among the actual positives.

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

- **F1-Score (weighted):** The harmonic mean of precision and recall, weighted average of all classes.

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

- **Per-class Precision, Recall, F1:** Metrics for each class to assess performance on minority classes as Bot.

Weighted Metric = $\sum_{i=1}^{n}\frac{Ni}{N}\times Metric$

- **Confusion Matrix:** It is a table that shows the relationship between actual and predicted class labels and summarizes the performance of each model.

V. Results and Discussion

A. Overall Model Performance

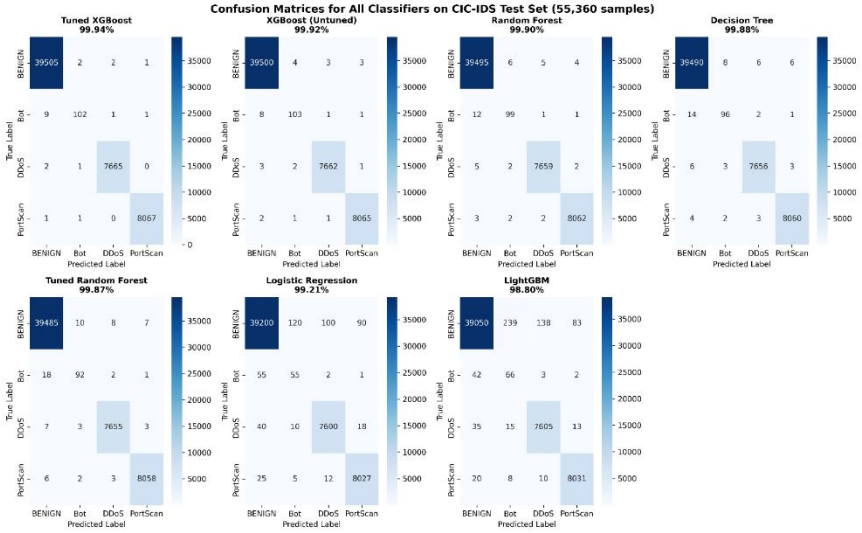
Table 2 summarizes overall accuracy, precision, recall and F1-score for seven different configurations of the model. This includes five base classifiers and two tuned variants, evaluated on a test set of 55,360 samples. The results are sorted in decreasing order of accuracy.

Table 2: *Overall Performance Comparison of All Classifiers*

Model	Accuracy	Precision	Recall	F1-Score	Rank
Tuned XGBoost	99.94%	99.94%	99.94%	99.94%	1
XGBoost	99.92%	99.93%	99.92%	99.93%	2
Random Forest	99.90%	98.90%	99.90%	99.90%	3
Decision Tree	99.88%	99.87%	99.88%	99.87%	4
Tuned Random Forest	99.87%	99.86%	99.87%	99.85%	5
Logistic Regression	99.21%	99.07%	99.21%	99.12%	6
LightGBM	98.80%	99.09%	98.80%	98.93%	7

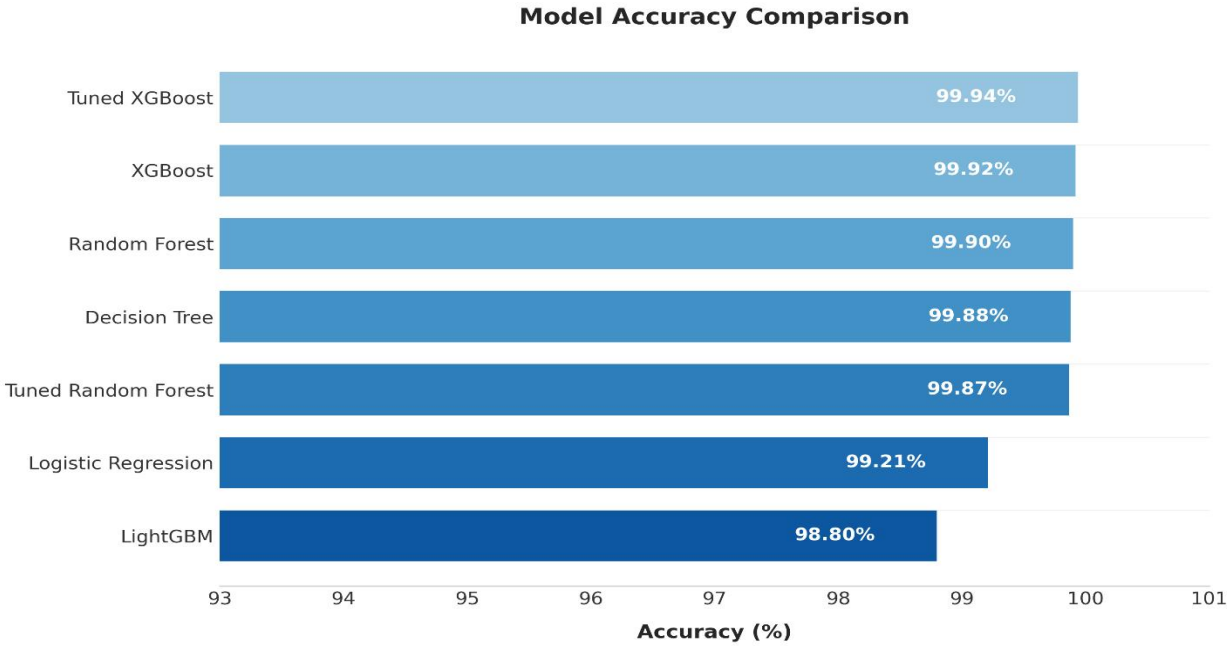
The results show a clear ranking of performance. Tuned XGBoost shows the best performance for all four metrics with 99.94%, followed by untuned XGBoost with 99.92%. The Random Forest gives 99.90% and the Decision Tree gives 99.88%. The Logistic Regression obtains 99.21%, proving that even a simple linear classifier performs well on this dataset, as the DDoS and PortScan traffic features are strongly linearly separable. Although LightGBM shows efficiency advantages, it obtains the lowest accuracy of 98.80% mainly due to its poor performance on the Bot minority class as discussed in Section VII-B.

Figure 2: Confusion Matrices for all Seven Classifiers



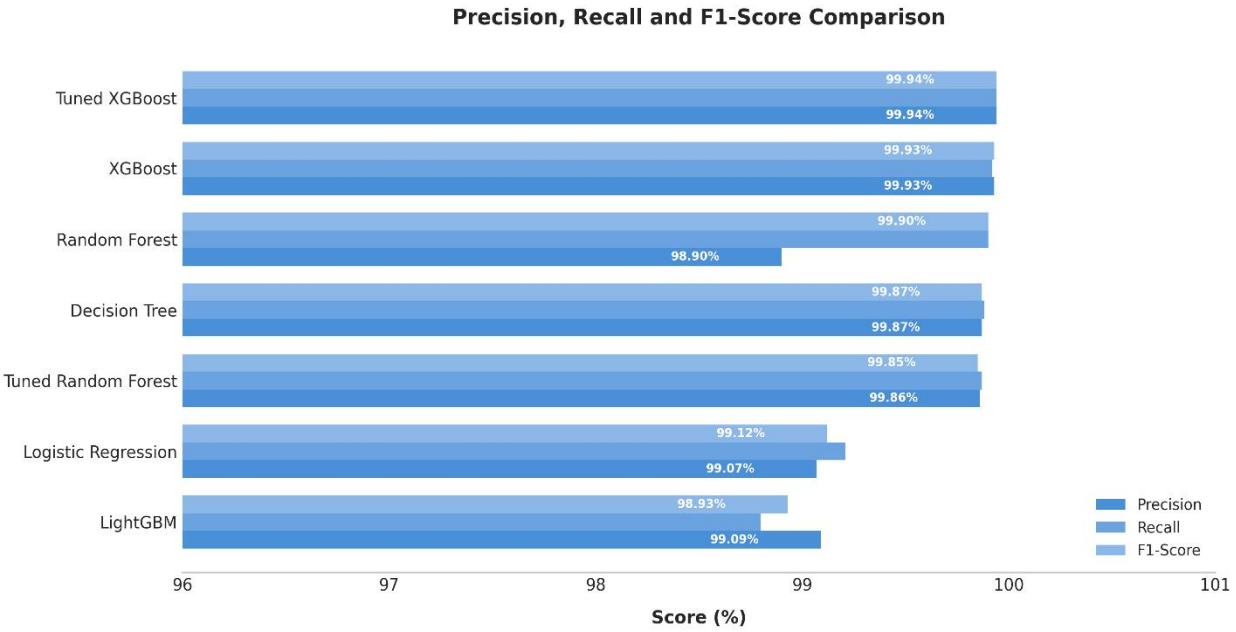
Each of these confusion matrices presents the classification output for the four traffic classes (BENIGN, DDoS, Bot, and PortScan) for each of the models. The diagonal is quite clean on Tuned XGBoost, with accuracy shown by the diagonal entries, and there are few mis-classifications. In LightGBM, there is plenty of 'off diagonal confusion' particularly misclassifying the BENIGN traffic as another attack.

Figure 3: Model Accuracy Comparison Bar Chart



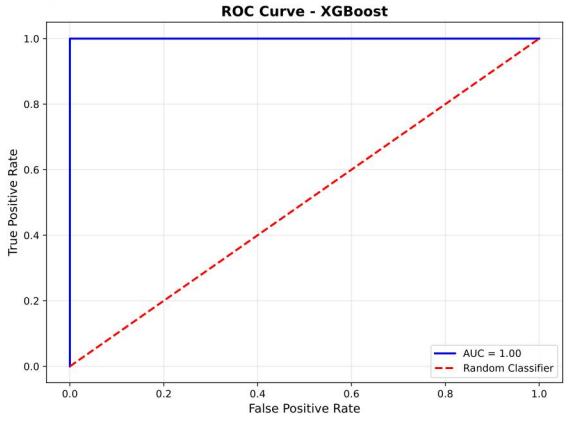
This bar chart illustrates the comparison of the 7 classifiers' accuracy scores in terms of overall accuracy. LightGBM is the least accurate with accuracy at 98.80% while tuned XGBoost is the most accurate with 99.94% with untuned XGBoost following closely at 99.92%. This is well shown in the graph where the performance gap between ensemble methods and the stronger Logistic Regression baseline is evident.

Figure 4: Precision, Recall and F1-Score Comparison



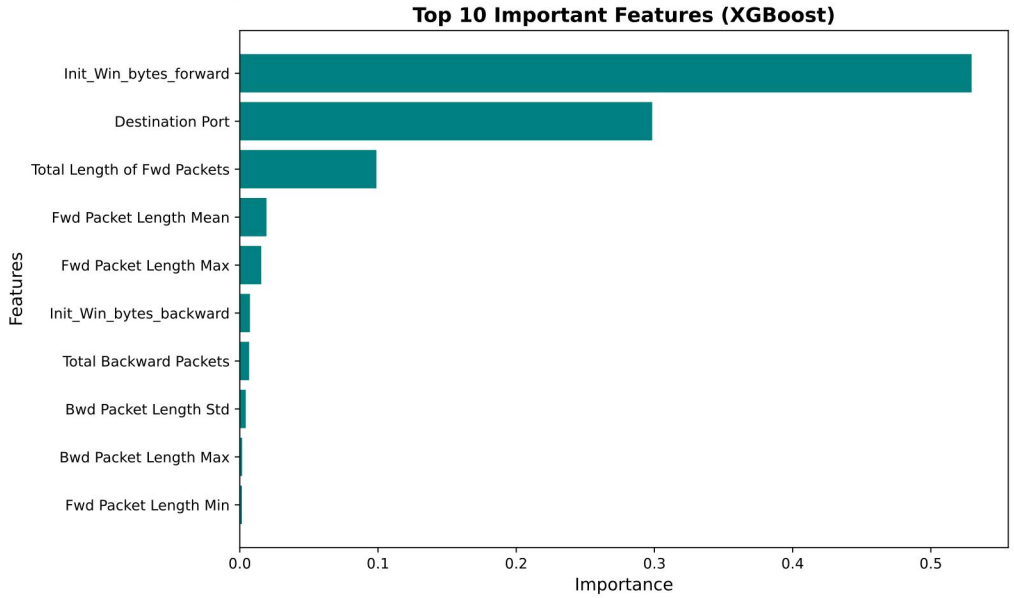
This grouped bar chart illustrates the precision, recall and F1-score of these three-measurement metrics for each of the three classifiers. IP TUNED XGBoost is a well-balanced engine with consistently high scores (99.94%) on all three measures. Even though LightGBM models perform well with most of the metrics, it has a slight loss of recall, as illustrated in the figure.

Figure 5: ROC Curve for XGBoost Classifier



ROC curve displays the true positive rate (TPR) and false positive rate (FPR) of the XGBoost classifier across different levels of classification. An Area Under the Curve (AUC) close to 1.0 is excellent discriminative performance between attack types. XGBoost generates high true positive rates while maintaining very low false positive rates, as demonstrated by the curve's strong rise toward the upper-left corner.

Figure 6: Feature Importance Analysis for XGBoost



The top network traffic characteristics that have the biggest impact on the XGBoost model's classification choices are ranked in this bar graph. The most important metrics for differentiating between traffic classes are characteristics like flow time, packet length statistics, and byte rate. A tiny fraction of characteristics contributes significantly to model performance, according to the analysis, opening the possibility of feature reduction for quicker inference.

B. Per-Class Analysis: Bot Traffic (Minority Class)

The Bot class has the most difficult classification problem with only 113 test instances while there are 39,510 BENIGN instances. Table 3 shows the per-class precision, recall and F1-score for the Bot class only for all models.

Table 3: *Per-Class Performance on Bot Traffic (Minority Class)*

Model	Precision	Recall	F1-Score
Logistic Regression	0.03	0.02	0.03
Decision Tree	0.82	0.73	0.77
Random Forest	0.87	0.73	0.79
XGBoost	0.80	0.88	0.84
Tuned XGBoost	0.90	0.83	0.87
Tuned Random Forest	0.85	0.55	0.67
LightGBM	0.21	0.58	0.31

Some main observations can be made from Table 3. Logistic Regression performs poorly in detecting Bot traffic with only 2% recall, which means it is not able to detect this class well. It is that the complicated behavioral patterns of Bot activity can't be documented by linear decision separating. It's that the complicated behavioral patterns of the Bot activity can't be documented by linear decision separating. LightGBM, however, achieved a higher recall of 58% and still low precision of 21%, which gives rise to lots of false positives. In practice, this type of action is not desirable for actual security systems as high number of false alarms can lead to 'alert fatigue' and waste resources.

The tuned XGBoost model gives the best trade-off between precision and recall with 90% precision and 83% recall resulting in an F1-score of 87%. So, if there are 100 instances of Bot the model will correctly identify 83 of them and will be quite confident when it predicts positive . These results show the importance of hyperparameter optimization. The XGBoost model without tuning achieved 80% precision and 88% recall (F1-score=0.84) . When tuned, the model shifts towards higher precision, which is a useful improvement in security applications where false positives can be costly.

We find an acceptable balance of precision and recall with a precision of 0.87 and a recall of 0.73, leading to an F1 score of 0.79 using the Random Forest method. But when we try to fine tune it the recall for Bot class drops drastically from 73% to 55%. This means that the fine-tuned Random Forest settings are over-fitted to the majority classes and under-perform the minority classes.

C. Impact of Hyper parameter Tuning

The difference between tuned and untuned models verifies that hyperparameter tuning affects the models differently and what is more optimal depends on the model. XGBoost tuning slightly improves the accuracy (99.92% to 99.94%) and precision (99.93% to 99.94%) with a strictly positive trend in all metrics. But, Random Forest shows a little decrease in the results when tuning, from 99.90% to 99.87%. This suggests that Random Forests default configuration was already well-suited to the dataset, and that adjusting may have resulted in little overfitting and, thus, reduced generalization.

The results illustrate that performance improvements do not necessarily need to rely on hyperparameter tuning and that the characteristics of the model and dataset coupled with the hyperparameter tuning influence the success. However, in the case of XGBoost it was always a constant tuning process that contributed to the best outcomes and this demonstrated the importance of tuning for maximum results. This means that with some tuning of parameters, and as is the case in this research, as much as an order of magnitude improvement in model performance is possible.

D. LightGBM Analysis

It turns out that LightGBM is the least powerful of all the models discussed, and can achieve its accuracy of 98.80% nonetheless. The confusion matrix study shows that the confusion rate is relatively high – 239 benign cases are misclassified in Bot category, 138 are misclassified in DDoS category and 70 are misclassified in PortScanning category. It shows that, although the training process of LightGBM is quite fast, the leaf-wise tree growing process may not truly represent the structure of this dataset. Consequently, the interfaces groups used to segment traffic on a network might not be very consistent. Among all the studies, LightGBM is the fastest model, with an obvious strength on training time. This is the most practical alternative when time-sensitive training and speedy computation come into play but classification results are less precise.

E. Summary

Network attackers, we need a good classifier, when it comes to detecting those attacks. Tuned XGBoost is best for multi-class network attack detection after experimentation of various options. The result on the CIC-IDS combined dataset is quite impressive – 99.94%. It does a great job of identifying Bot class attacks and achieves an F1 of 0.87. However, sometimes we want something that is fast, and doesn't require too much computer power. For those cases, an XGBoost or a Random Forest is ok alternatives. They don't get quite as close, but they are very precise, at 99.92% and 99.90%, respectively.

VI. Limitations and future work

The study results are valid and stable, however, it is crucial to note a few limitations to the suggested approach that may influence its usefulness or applicability. First, all tests are conducted on pre-collected samples of offline dataset and the evaluation of generalization for “real-time network traffic” which has a natural distribution was not evaluated, as characterized by concept drift, novel attack patterns. Real-time deployment must still be validated in production network conditions to ensure models can deal with the changes in operational network settings that they see. Secondly, the CIC-IDS merged data set had been acquired in a controlled laboratory environment despite its

large-scale and its being a recognized standard. The generalized model may not be applicable to real enterprise or cloud networks if they have differing traffic patterns and distributions that may preclude the model from generalizing without training on domain-specific data.

Thirdly, computing resources limitations in Google Colab environment has forced the use of only 20% of the total data sample (184,532 among the ~961,678 results). While stratified sampling was employed to keep class proportions as a constant, one must be aware of this sample limitation; training on the full dataset may yield different results, so these results must be carefully interpreted with this limitation in mind. Fourth, the base paper used XAI analysis, while some of the current work did not employ XAI analysis. Further, this is a significant limitation in terms of the models' applicability in practice, as the feature level explanations would be required to trust and leverage model predictions for operational deployment by security analysts.

Fifth, the Bot class is still the hardest to correctly categorize, even after balancing with SMOTE and obtaining the best model F1-score of 0.87. With only 263 original training cases, the distribution of Bot traffic attributes is poorly represented, and building a highly confident classifier for this minority class is still difficult even with upsampling. Lastly, the computational resources on Google Colab restricted the hyperparameter search space for RandomizedSearchCV; a more extensive search across a larger hyperparameter range with more folds (5-fold or 10-fold cross-validation) might result in additional performance gains.

Future Work: In future work, the paper extends the current framework in several directions to improve interpretability, scalability, robustness and real-world applicability. SHAP and LIME based explainable AI (XAI) techniques will be employed on all the five classifiers on the CIC-IDS dataset, so as to enable a comparative interpretability analysis of feature contributions, enhancing transparency and trust in security-critical environments. We will also explore the real-time deployment, which will include some deployment examples using the stream processing frameworks Apache Kafka or Apache Flink, and will consider performance metrics like latency, throughput and concept drift adaptability. We will also discuss how these hybrid deep learning models work in the presence of XGBoost and CNN/LSTM, providing an boost for feature representation and classification accuracy. The study will also expand to the entire CIC-IDS data available at large scale on distributed platforms (e.g., Google Cloud, or AWS SageMaker) to validate the generalizations using a large-scale data set. The study will further examine the model's capacity to withstand adversarial perturbations through evaluation against FGSM-based attacks specifically designed for tabular data, thereby determining its resilience to evasion methodologies. Furthermore, federated learning approaches will be employed to facilitate privacy-conscious distributed model training among multiple entities, while simultaneously utilizing AutoML frameworks including Auto-sklearn and FLAML to optimize hyperparameter configurations and identify enhanced neural architectures suitable for intrusion detection applications.

VII. Conclusion

In this study, a thorough machine learning framework for multi-class classification of network attacks using the CICIDS 2017 combined dataset has been presented. The main contributions of this work are to cover the following significant gaps in the literature: the prevalence of binary classification, ignoring data imbalance, incomplete comparison of classifiers, and lack of detailed evaluation.

Experimental results are based on 184,532 cleaned network traffic samples with 78 features over four classes (BENIGN, DDoS, Bot, PortScan) and five ML classifiers (Logistic Regression, Decision Tree, Random Forest, XGBoost and LightGBM) for the same experimental conditions. In this research SMOTE-based oversampling to resolve the extreme class imbalance of the Bot minority class and RandomizedSearchCV-based hyperparameter tuning for XGBoost and Random Forest.

The best overall performance shown by our experimental results is the Tuned XGBoost classifier with 99.94% accuracy, precision, recall, and F1-score, which is the recommended classifier for this task. The per-class analysis reveals that the most challenging traffic class to predict is the Bot

class. The best model achieved an F1-score of 0.87, a significant improvement over simpler classifiers such as Logistic Regression (F1=0.03) and LightGBM (F1=0.31). Hyperparameter tuning consistently but modestly improves XGBoost, and slightly decreases the performance of Random Forest, indicating the context-dependent nature of tuning benefits.

The addition of LightGBM as a new entrant in this classifier comparison shows that while it is computationally efficient, LightGBM is beaten by ensemble competitors on this dataset because of higher false positive rates across minority classes. This finding provides an important trade-off consideration to practitioners deploying algorithms in resource-constrained environments.

The paper establishes a reproducible, resource-efficient and comprehensive experimental baseline for multi-class network intrusion detection on the CICIDS 2017 dataset. The enhancement will focus on improving this framework to make it more transparent by implementing XAI techniques, and incorporating real-time evaluation and adversarial robustness testing, to build full-deployable trustworthy and interpretable network security systems.

VIII. REFERENCES

[1] Z. M. Jiyad, A. A. Maruf, M. M. Haque, M. S. Gupta, A. Ahad, and Z. Aung, "DDoS Attack Classification Leveraging Data Balancing and Hyperparameter Tuning Approach Using Ensemble Machine Learning with XAI," in Proc. ICPC2T, 2024, pp. 569–575.

[2] M. I. Mohmand, H. Hussain, A. A. Khan, U. Ullah, M. Zakarya, A. Ahmed, M. Raza, I. U. Rahman, and M. Haleem, "A machine learning-based classification and prediction technique for DDoS attacks," IEEE Access, vol. 10, pp. 21443–21454, 2022.

[3] C. S. Kalutharage, X. Liu, C. Chrysoulas, N. Pitropakis, and P. Papadopoulos, "Explainable AI-based DDoS attack identification method for IoT networks," Computers, vol. 12, no. 2, p. 32, 2023.

[4] S. Ullah, Z. Mahmood, N. Ali, T. Ahmad, and A. Buriro, "Machine learning-based dynamic attribute selection technique for DDoS attack classification in IoT networks," Computers, vol. 12, no. 6, p. 115, 2023.

[5] A. K. Silivery, K. R. M. Rao, and L. Kumar, "An effective deep learning based multi-class classification of DoS and DDoS attack detection," Int. J. Electrical and Computer Engineering Systems, vol. 14, no. 4, pp. 421–431, 2023.

[6] S. Aktar and A. Y. Nur, "Towards DDoS attack detection using deep learning approach," Computers & Security, vol. 129, p. 103251, 2023.

[7] A. E. Cil, K. Yildiz, and A. Buldu, "Detection of DDoS attacks with feed forward based deep neural network model," Expert Systems with Applications, vol. 169, p. 114520, 2021.

[8] S. Ahmed, Z. A. Khan, S. M. Mohsin, S. Latif, S. Aslam, H. Mujlid, M. Adil, and Z. Najam, "Effective and efficient DDoS attack detection using deep learning algorithm, multi-layer perceptron," Future Internet, vol. 15, no. 2, p. 76, 2023.

[9] F. Hussain, S. G. Abbas, M. Husnain, U. U. Fayyaz, F. Shahzad, and G. A. Shah, "IoT DoS and DDoS attack detection using ResNet," in Proc. IEEE INMIC, 2020, pp. 1–6.

[10] M. Aamir and S. M. A. Zaidi, "Clustering based semi-supervised machine learning for DDoS attack classification," J. King Saud University-Computer and Information Sciences, vol. 33, no. 4, pp. 436–446, 2021.

[11] T. E. Ali, Y.-W. Chong, and S. Manickam, "Machine learning techniques to detect a DDoS attack in SDN: A systematic review," Applied Sciences, vol. 13, no. 5, p. 3183, 2023.

[12] P. Dong, X. Du, H. Zhang, and T. Xu, "A detection method for a novel DDoS attack against SDN controllers by vast new low-traffic flows," in Proc. IEEE ICC, 2016, pp. 1–6.

[13] S. K. Naing and T. T. Thwel, "A study of DDoS attack classification using machine learning classifiers," in Proc. IEEE ICCA, 2023, pp. 108–112.

[14] V. Goel, P. Goel, R. Ranjan, and A. K. Sharma, "An effective classification of DDoS cloud based attack through tree founded classifiers," in Proc. IEEE IHCSP, 2023, pp. 446–449.

[15] J. Liu, "Importance-SMOTE: a synthetic minority oversampling method for noisy imbalanced data," Soft Computing, vol. 26, no. 3, pp. 1141–1163, 2022.
[16] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in Proc. ICISSP, 2018, pp. 108–116.
[17] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. ACM SIGKDD, 2016, pp. 785–794.
[18] Y. Hosain and M. Çakmak, "XAI-XGBoost: An innovative explainable intrusion detection approach for securing Internet of Medical Things systems," Scientific Reports, vol. 15, Art. no. 22278, 2025.
[19] P. Hermosilla, S. Berríos, and H. Allende-Cid, "Explainable AI for forensic analysis: A comparative study of SHAP and LIME in intrusion detection models," Applied Sciences, vol. 15, no. 13, 2025.

[20] A. E. Muhammad, K.-C. Yow, N. Bačanin-Džakula, et al., "L-xaids: A LIME-based explainable AI framework for intrusion detection systems," Cluster Computing, vol. 28, Art. no. 654, 2025.
[21] B. Nugraha, A. V. Jnanashree, and T. Bauschert, "A versatile XAI-based framework for efficient and explainable intrusion detection systems," Annals of Telecommunications, vol. 80, pp. 1095–1120, 2025.
[22] A. Grabowski and S. Xu, "Advancing cybersecurity practice: Explainable machine learning for network intrusion detection," Journal of Cybersecurity Education, Research and Practice, 2025.