Optimizing Intrusion Detection through Web Traffic Using the CIC UNSW-NB15 Augmented Dataset

Maya Shah
UIN: 731001284
mayashah.tam@tamu.edu
Department of Computer Science and Engineering
Texas A&M University
College Station, TX, 77840

Gabriel Wild
UIN: 731003602
gabew2024@tamu.edu
Department of Computer Science and Engineering
Texas A&M University
College Station, TX, 77840

***Abstract*—As the internet continues to expand, cyberattacks have become an increasingly sophisticated and prevalent threat. Cyberattacks leave a footprint in the form of web traffic. Network flow statistics like packet count, byte count, and duration can be indicative of a cyberattack. The sheer amount of web traffic flowing through the network on a daily basis makes it impossible to manually look at every traffic record and determine its risk. The CIC UNSW-NB15 Augmented Dataset created by the Canadian Institute for Cybersecurity (CIC) in collaboration with the University of New South Wales (UNSW) is a labeled dataset of roughly 450,000 samples of malicious and benign web traffic network flow statistics. In this study, we evaluate multiple machine learning models trained on this dataset and optimize the highest performing model through hyper-parameter tuning to identify and categorize benign and malicious web traffic.**

***Index Terms*—Network Intrusion Detection Systems (NIDS), Machine Learning, Cybersecurity, Threat Detection**

**** You can find the code and video at [Code][Video]**

I. Introduction

In an increasingly interconnected and technology-driven world, the prevalence of cybercrime has reached unprecedented levels, posing significant threats to individuals, businesses, and governments alike. Recent statistics underscore the gravity of this issue. In 2023 alone, over 2,365 cyberattacks were reported, affecting more than 343 million victims globally [1]. Notably, this represents a 72% increase in data breaches compared to 2021 [1]. The financial repercussions are equally alarming, with damages from cybercrime projected to reach $10.5 trillion by 2025 [1]. These trends highlight the urgent need for robust and accurate real-time intrusion detection systems, which are essential for mitigating risks and safeguarding sensitive information in the digital age.

Recent cyberattacks are becoming increasingly sophisticated, often evading existing intrusion detection systems through the use of advanced techniques. A significant driver of this is the incorporation of machine learning (ML) and artificial intelligence (AI) by malicious actors. In a 2024 interview with Wall Street Journal, CJ Moses, Amazon's Chief Information Security Officer, stated that the company detects approximately one billion cyber threats daily [2]. Moses also highlighted how generative AI technology has expanded the capabilities of cybercriminals, enabling individuals without formal software engineering or cybersecurity knowledge to orchestrate cyberattacks [2]. While it's clear these emerging technologies pose new threats, they also offer opportunities for enhancing cybersecurity. Moses went on to tell Wall Street Journal that Amazon has created a graph database with information on interactions occurring in their environment [2]. Machine learning models can then utilize this vast amount of data to identify attacker patterns in real-time and perform predictive analysis, potentially mitigating threats before they occur.

The primary objective of this paper is to advance the understanding of machine learning (ML) applications in network intrusion detection by building upon prior research. A recent study [3] evaluated the performance of various ML models using the UNSW-NB15 dataset, a well-established benchmark in cybersecurity research. This study extends that work by employing the CIC UNSW-NB15 Augmented Dataset, a recently published dataset based on the original UNSW-NB15 that incorporates enhanced preprocessing techniques. Given its novelty, minimal research has been conducted on this dataset, providing an opportunity for meaningful exploration.

To address this gap, a two-phase experimental approach was adopted:

- Evaluate the efficacy of multiple ML algorithms on the augmented dataset, mirroring the methodology of [3].
- Conduct a comparative analysis to the models in [3] and an assessment of whether preprocessing techniques influence the effectiveness of specific ML models.
- Utilize hyper-parameter tuning on the best-performing ML model to refine its accuracy and uncover nuances that impact performance.

This dual-phase approach not only identifies an optimal Network Intrusion Detection System (NIDS) model for the CIC UNSW-NB15 Augmented Dataset but also provides insights applicable to similar datasets, enhancing the practical relevance of this research.

II. Related Works

There are multiple previous works that build the foundation for this particular research. As previously stated, the CIC UNSW-NB15 Augmented Dataset [5] used in this study is

based on the UNSW-NB15 Dataset created by [4] in the Cyber Range Lab of UNSW Canberra. In their paper, [4] detail the creation of UNSW-NB15. They utilized the IXIA Perfect Storm tool to provide a hybrid method of traffic generation, incorporating real normal and synthesized abnormal network traffic. Tools such as Argus and Bro-IDS were used to derive features, resulting in a dataset composed of 49 features and 10 labels (1 benign and 9 attack types) mimicking real-world threats. Once completed, a comparative analysis was conducted between the new UNSW-NB15 Dataset and an older KDD99 dataset. It was found that UNSW-NB15 had better overall data comprehensiveness due to more enhanced complexity and variety in the features of the dataset and its representation of various attacks. Thus, it was determined that UNSW-NB15 could serve as a well-constructed dataset for further research in the evaluation of machine learning and Network Intrusion Detection Systems (NIDS).

The CIC UNSW-NB15 Augmented Dataset [5] uses UNSW-NB15 as its foundation. However, it utilizes the CICFlowMeter Tool to extract flow-level features (packet counts, byte counts, packet states) from the raw UNSW-NB15 dataset. This is a different preprocessing method than the original dataset. We've concluded that the previous research of [4] which determined UNSW-NB15 as a comprehensive dataset, combined with the new derivatives provided by the CIC Augmented version by [5] would serve as a good dataset for this particular project. Furthermore, a study by [3] aimed to increase accurate detection of intrusions by comparing the efficacy of various machine learning algorithms against the UNSW-NB15 dataset. The models tested included logistic regression, support vector machine (SVM), decision tree, and random forests.

The results of their experiments and performance evaluation metric calculations determined that Random Forest was the most effective baseline model, with an accuracy of 99.42%, precision of 99.71%, recall of 99.63%, F-1 score of 99.67%, and a false alarm rate (FAR) of 2.04% [3]. After calculating the baseline model results, [3] did some hyper-parameter tuning within their own work as well, on the decision tree model and the random forest model. From this, it was determined that the Random Forest model also utilizing Feature Selection was the overall highest performing model, with an accuracy of 99.45%, precision of 99.72%, recall of 99.65%, F-1 score of 99.65%, and a false alarm rate (FAR) of 1.94% [3].

However, it remains unclear from the paper whether the models were trained as multi-class or binary classifiers. Multi-class models classify inputs as benign or a specific attack type, whereas binary models simply distinguish between benign and malicious traffic without identifying the attack type. In Section 3 ("Proposed Research Methodology") of the paper [3], the authors mention that "these algorithms have been selected for various reasons, including their wider applications for binary classification and existing applications in machine learning-enabled IDS" [3]. However, they later discuss the potential of SVMs to be expanded for multi-class classification and highlight how decision trees and random forests are "suitable for finding non-linear relationships...[and]...relevant features for classification" [3]. This suggests the possibility of both binary and multi-class training approaches being considered, but further clarification is needed.

This study focuses on building multi-classification models to not only distinguish benign traffic from malicious traffic, but also to categorize the specific type of attack occurring. The ability to identify the attack type is critical, as different attacks may require different mitigation strategies. This real-time categorization provides valuable insights for intrusion detection systems (IDS). Furthermore, [3] highlights the potential for future work to incorporate neural network models, prompting this study to include a convolutional neural network (CNN) as part of its experimentation. While feature selection has previously been demonstrated as an effective method of hyper-parameter tuning [3], this study also seeks to explore additional hyper-parameter tuning methods for the best-performing model, aiming to identify the most effective approach.

III. Methodology

The following subsection provides details about the dataset and its various attack types. The subsequent subsection describes the five machine learning algorithms evaluated for their intrusion detection system (IDS) capabilities. Four of these algorithms are based on models from [3] and leverage the most effective hyperparameter configurations identified in that study. The fifth is a convolutional neural network model that was developed specifically for this study. The third subsection outlines the evaluation metrics used to determine the highest-performing model among the five. Lastly, subsection four discusses the hyperparameter tuning techniques applied to the best-performing model to further optimize its performance.

A. *Dataset Properties*

As previously mentioned, the CIC UNSW-NB15 Augmented Dataset [5] is a network traffic dataset containing 447,915 records. The dataset has 10 different target classifications, making this suitable for a multi-class classification model. The corresponding Label.csv file for this Data.csv, contains numbers from 0-9, each representing a benign record (0) or an attack type (1-9). The attack types are listed in detail below [5]:

1) **Fuzzers:** The sending of random/unexpected input into an application to determine how it will respond. The technique helps malicious users discover vulnerabilities that it can exploit.
2) **Analysis:** The collection of data for various analysis methods (traffic, cryptographic, code, data, protocol, etc.) in order to gain insights or extract sensitive data for malicious use.
3) **DoS:** An attack type which overwhelms the target application with excessive amounts of request which makes it crash or slow down.
4) **Backdoor:** Insertion of malicious code or modification of existing code through unauthorized access to a network.

5) **Exploit:** Exploit attacks refer to exploitation of computer systems, applications, or network protocols to perform various malicious activities through the execution of commands.
6) **Generic:** Attacks against cryptography systems.
7) **Reconnaissance:** The collection of intelligence to gain an understanding of the application infrastructure, vulnerabilities, and potential entry points.
8) **Shellcode:** A payload that is injected into a vulnerable program in order to gain access and control over the system.
9) **Worms:** A type of cyberattack that can independently replicate and move through an entire network without any user interaction.

The dataset mimics real-world network traffic, comprising 80% benign records and 20% malicious records spanning various attack types [5]. However, this distribution introduces an inherent class imbalance, which can adversely affect model performance. Traditional approaches to address class imbalance include oversampling the minority class or undersampling the majority class. While effective in some cases, these techniques can introduce challenges such as overfitting (in the case of oversampling) or loss of valuable information (in the case of undersampling), both of which can compromise model efficacy and accuracy. To address these limitations, this study employs the `class_weight` attribute, which dynamically adjusts the weight assigned to each class based on the inverse of its frequency in the training data [6]. The formula used by scikit-learn classifiers to compute class weights is as follows:

$$x_i = \frac{n_{samples}}{n_{classes} \times np.bincount(y)} \tag{1}$$

Aside from this class balancing, the CIC UNSW-NB15 Augmented Dataset has already been preprocessed for the desired use case. It contains two separate csv files, one containing all the features and one containing all the labels, offering the data in a ready-to-use format for the models.

B. Model Generation

The CIC UNSW-NB15 Augmented Dataset was divided into training and testing sets using an 80/20 split, with 80% of the data allocated for training and 20% reserved for testing. Figure 1 illustrates the process of dataset utilization for training and testing each model.

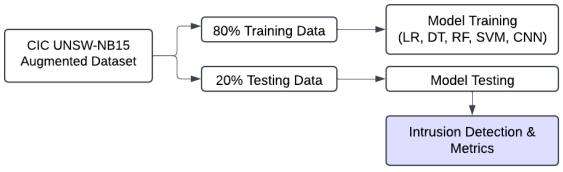


Fig. 1. Dataset Utilization for Machine Learning Models

Each model was trained with the parameters found most effective from the previous study [3] and are described in detail in the subsequent sections.

*1) **Logistic Regression**:* The logistic regression model was implemented following the step-by-step methodology outlined in [3]. However, the initial attempt using a multi-class classification approach yielded poor results, even after extensive hyperparameter tuning of the alpha and penalty parameters as suggested in [3]. Consequently, the model was reconfigured to perform binary classification instead. In this adaptation, the target labels (commonly referred to as `y_train` and `y_test`) were transformed into binary arrays (`y_train_binary` and `y_test_binary`), where the label 0 remained as 0, while labels 1 through 9 were consolidated into the binary value 1. This transformation allowed the model to focus on determining whether a sample was benign (0) or represented an attack (1). The modified logistic regression model retained the following key parameters:

- `alpha`: A constant multiplied against the regularization function. The larger the value, the stronger the regularization [11].
- `penalty`: The regularization function to be used. The value was set to `l2`, which is the standard regularization function for linear support vector machine models [11].
- `max_iter`: The number of epochs run with the training data [11].
- `loss`: Specifies the type of loss function to use – in this case `log_loss` represents the use of a logistic regression function [11].

Additionally, 10% of the training data was reserved for validation, and early stopping was enabled with a tolerance of 1e-4, halting training after 10 consecutive iterations with no improvement.

*2) **Decision Tree**:* The decision tree model in this study was constructed using the optimal parameters identified in [3], which were shown to enhance the performance of this model type on the original UNSW-NB15 dataset. These parameters include `min_samples_leaf=9`, `min_samples_split=6`, and `max_depth=6` [3]. The definitions and significance of these parameters are detailed below.

- `min_samples_leaf`: The minimum number of nodes that must be leaves for there to be any consideration of further splitting of the tree [7].
- `min_samples_split`: The minimum number of nodes that must exist at an internal node in order for it to be eligible for splitting [7].
- `max_depth`: The maximum number of levels the decision tree can be split down to [7].

*3) **Random Forest**:* Similar to the decision tree model, the random forest was constructed using the parameters identified as optimal in [3]. These included `criterion="gini"`, `max_depth=22`, `min_sample_split=6`, `n_estimators=300`, and `n_jobs=-1`. Parameters `max_depth` and

min_sample_split are the same as defined in the above decision tree section. The remainder are defined below.

- criterion: This parameter defines what measure to utilize when determining if a split should occur within the tree. The value of "gini" means the measurement is based on the Gini Impurity – a value between 0 (pure) and 1 (impure) [8].
- n_estimators: The number of trees that comprise the random forest [8].
- n_jobs: This parameter determines the number of jobs that can be run in parallel. Setting the value to -1 means all available processors are utilized [8].

*4) **Support Vector Machine (SVM)**:* Support Vector Machines (SVMs) classify data by finding a hyperplane that separates classes with the maximum margin. As described in [3], the linear SVM is the specific type of SVM employed in this study. Using the scikit-learn library, two approaches are available for training a linear SVM. The first approach involves using the SVC model with the kernel set to "linear". This method relies on an implementation based on libsvm, and has a training time that scales at least quadratically in the best-case scenario [9]. Consequently, SVC can become computationally expensive and impractical for datasets larger than tens of thousands of samples [9]. The second approach utilizes the LinearSVC model, which is built on liblinear, a more computationally efficient implementation, making it better suited for large-scale datasets [10]. Given the size of the CIC UNSW-NB15 Augmented Dataset, which contains hundreds of thousands of samples, and the computational limitations of running the experiments on Google Colab, the LinearSVC model was selected as the more efficient option. No additional hyperparameters were specified for the model.

*5) **Convolutional Neural Network (CNN)**:* A 1D Convolutional Neural Network (CNN) applies sliding filters (kernels) over a one-dimensional input sequence to identify relationships and extract patterns that contribute to the activation of neurons. In this approach, each record is treated as a distinct one-dimensional input, with each column representing a separate feature. The CNN architecture was implemented using the TensorFlow Sequential API [12] and consisted of three convolutional layers, each followed by batch normalization and max-pooling operations. These layers were designed to progressively extract hierarchical features from the data. Following the convolutional layers, the output was flattened and passed through two fully connected dense layers, enabling the network to learn class-specific features. The final output layer comprised 10 neurons, each corresponding to one of the classification labels. The model was compiled with the Adam optimizer, configured with a learning rate of 0.0001, and employed a sparse categorical cross-entropy loss function suitable for multi-class classification with integer-encoded labels. The choice of a lower learning rate facilitated more controlled parameter updates during training, helping the model focus on meaningful patterns and reduce sensitivity to noise. To address class imbalance in the dataset, class weights were computed to assign higher importance to underrepresented classes during training. Lastly, to mitigate over-fitting, 10% of the training data was set aside as a validation set, and early stopping was employed.

C. Highest Performing Model Selection Criteria

The highest performing model was selected based on its overall performance on the following metrics. Note that TP, FP, TN, and FN stand for True Positive, False Positive, True Negative, and False Negative, respectively.

$$Precision = \frac{TP}{TP+FP} \tag{2}$$

$$Recall = \frac{TP}{TP+TN} \tag{3}$$

$$F-1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} \tag{4}$$

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \tag{5}$$

In evaluating model performance, it is important to recognize that accuracy may not be the most reliable metric when dealing with an imbalanced dataset. As previously discussed, the dataset used in this study reflects a real-world scenario, where 80% of the data is benign, and 20% corresponds to various attack types. Given this imbalance, accuracy can be misleading. To address this issue, the F1-score was employed, as it provides a more balanced measure by considering both precision and recall. Precision, which reflects the positive predictive value, indicates the proportion of predicted positives that are actually true positives, while recall (sensitivity) measures the model's ability to identify all positive instances. Consequently, the F1-score, which balances the trade-off between precision and recall, is given the highest priority in performance assessment, followed by precision, recall, and accuracy.

Additionally, model performance was assessed through graphical evaluations, including the Receiver Operating Characteristics (ROC) curve and the Precision-Recall Curve (PRC). However, similar to the limitations of certain metrics, the ROC curve may not accurately reflect performance on imbalanced datasets. In such cases, the False Positive Rate (FPR) can still stay low since the number of True Negatives will outweigh the number of misclassifications of positive samples of the minority class. However, the PRC provides a clearer representation of the model's effectiveness in the model's ability to handle performance on the positive class (the minority class) and maintain a balance between the false positives and false negatives. As such, the PRC is weighted more heavily in the performance evaluation compared to the ROC curve.

D. Hyper-Parametrization of Highest Performing Model

To optimize the highest performing model, three systematic hyper-parametrization approaches were conducted: Grid Search, Optuna, and Hyperopt. The best hyper-parametrization of the model is decided by weighted F-1 Score. Each method is given values within these ranges for 4 the following parameters:

- n_estimators: [100,500]

- `max_depth`: [10,50]
- `min_samples_split`: [2,10]
- `min_samples_leaf`: [1,5]

Grid Search performs an exhaustive search over the list of hyper-parameters, training models with the different number of trees, maximum tree depth, and features considered for splitting. The grid search algorithm was implemented using the sci-kit function GridSearchCV with 3 fold cross validation to improve speed and ensure robust evaluation [13]. This exhaustive approach will ensure that there are samples of hyper parametrization with all combinations of high, medium, and low values for each parameter.

Optuna expands the search space to the in-betweens of the bounds of the GridSearch. Optuna is a state-of-the-art optimization framework that utilizes a tree-structured Parzen estimator (TPE) sampler to intelligently navigate the hyper-parameter space [14]. The TPE sampler is a probabilistic model that builds a distribution for each hyper-parameter and samples it to suggest higher performing configurations, with the proceeding searches prioritizing regions where previous trials have shown success. This reduces computational costs, while increasing the likelihood of finding optimal hyper-parameters.

Similar to Optuna, Hyperopt is a framework designed to use a Bayesian probabilistic approach to hyper-parameter tuning [15]. This Hyperopt function employs a probabilistic model that uses simulated annealing for searching for an optimization between the bounds. Simulated Annealing mimics metallurgy which heats and cools materials to find a state with minimal entropy by exploring parameters to find an optimal model by measuring the entropy gained and loss from trial to trial. Each configuration's performance was evaluated using cross-validation, and the best configuration was selected across 20 trials.

IV. Results & Discussion

This section below presents a detailed evaluation of the performance of the models developed in this study. It also compares their performance both within the context of this study and against the results reported in [3]. The discussion aims to provide insights into the strengths and limitations of each model, offering a comprehensive understanding of their relative effectiveness.

A. Model Evaluation

Table 1 below presents the performance metrics used to evaluate the effectiveness of the machine learning models analyzed in this study.

TABLE I
Model Evaluation Metrics

Model	Precision	Recall	F-1	Accuracy
Logistic Regression	0.8267	0.6752	0.7076	0.6752
Decision Tree	0.9493	0.8773	0.9037	0.8773
Random Forest	0.9451	0.9325	0.9354	0.9325
SVM	0.9177	0.8868	0.8959	0.8868
CNN	0.7830	0.8205	0.8205	0.7918

*1) **Logistic Regression**:* The logistic regression model was the least successful among the models evaluated. As discussed earlier, it required conversion from a multi-class classification framework to binary classification to achieve any level of performance. Despite this adjustment, it yielded a precision of 0.8267, recall of 0.6752, accuracy of 0.7076, and an F1-score of 0.6752, significantly lower than the other models, which mainly achieved metrics in the upper 80s to mid-90s. This combination of poor performance metrics and its inability to function effectively as a multi-class classification model rendered logistic regression unsuitable for further hyperparameter tuning or consideration as the top-performing model.

*2) **Decision Tree**:* The decision tree model ranked as the second-best performing model overall. It achieved the highest precision among all models, with a value of 0.9493. However, its recall and F1-score were both 0.8773, ranking third overall for these metrics. Its accuracy was slightly higher, at 0.9037, placing it second in this category. This combination of metrics, particularly the exceptional precision, secured its position as the second-best model in the study.

*3) **Random Forest**:* The random forest model emerged as the highest-performing model among those evaluated in this study. Although its precision (0.9451) was slightly lower than that of the decision tree (0.9493), the difference was minimal. Moreover, the random forest outperformed all other models across the remaining metrics. It was the only model to achieve both a recall and F1-score above 0.90, with these metrics recorded at 0.9325. Its accuracy was also notable at 0.9354. As the only model with all key performance metrics exceeding 0.90, the random forest demonstrated superior performance.

This outcome aligns with how a random forest algorithm works. By constructing an ensemble of decision trees, random forests enhance overall accuracy while mitigating the risk of overfitting—a common limitation of individual decision trees. Given that the decision tree model ranked second overall, the strong performance of the random forest is consistent with expectations, as it builds upon the decision tree framework to achieve greater robustness and predictive power.

*4) **Support Vector Machine (SVM)**:* The linear support vector machine ranked as the third-best performing model in this study. It achieved a precision of 0.9177, surpassing the 0.90 benchmark. However, its other metrics fell short of this threshold, with a recall and F1-score of 0.8868 and an accuracy of 0.8959. While the model demonstrated relatively strong performance, it was outperformed by other models evaluated in this study.

*5) **Convolutional Neural Network (CNN)**:* The convolutional neural network (CNN) model had not been tested on the UNSW-NB15 dataset in the previous study [3]. Consequently, its performance metrics are only available for the CIC UNSW-NB15 Augmented Dataset. The CNN model exhibited the lowest precision among all models, with a value of 0.7830, performing worse than the logistic regression model. However, its F1-score of 0.7918 surpassed that of logistic regression. Both recall and accuracy were recorded at 0.8205. Overall,

these results position the CNN model as the second-lowest performing model in this study and therefore, further hyper-parameter tuning of this model was not considered.

B. Comparative Analysis against UNSWB-N15 Models

The evaluation metrics for the models in this study were lower than those reported in the previous study [3], where all models achieved metrics between 0.9879 and 0.9995. This discrepancy can be attributed to several factors. First, [3] applied extensive data preprocessing to the UNSW-NB15 dataset, which included data cleaning, exploratory data analysis to identify and remove highly correlated features, and feature engineering to standardize the data. While the CIC UNSW-NB15 Augmented Dataset utilized preprocessing techniques such as feature extraction with CICFlowMeter, these methods appear to have been less effective in achieving the high-performance metrics observed in [3].

Despite these differences, the performance hierarchy of the models remained relatively consistent between the two studies. Both identified the random forest model as the highest performing, followed by the decision tree. However, a key divergence was observed in the lowest-performing model: [3] identified the linear support vector machine (SVM) as the weakest model, whereas this study found the logistic regression model to perform the worst. Notably, the convolutional neural network (CNN) model was not part of [3]'s original study but was suggested as a direction for future work and thus, incorporated into this study.

TABLE II
MODEL FALSE ALARM RATE

Model	False Alarm Rate (FAR)
Logistic Regression	0.3672
Decision Tree	0.0284
Random Forest	0.0259
SVM	0.0282
CNN	0.0235

An additional point of comparison arises in the false alarm rate (FAR) metrics, as shown in Table 2. In this study, the CNN model achieved the best FAR; however, its poor performance across other evaluation metrics disqualified it from being considered a high-performing model. The random forest model, identified as the top-performing model, had the second-best FAR. Although its FAR was slightly higher than the 0.0204 reported in [3], the difference was minimal. Interestingly, apart from the logistic regression model, all other models in this study exhibited substantially lower FARs than their counterparts in [3]. For example, the decision tree model in this study achieved a FAR of 0.0284 compared to 0.0575 in [3], and the SVM model recorded a FAR of 0.0282, significantly lower than the 0.0848 reported in [3]. The logistic regression model, however, had an exceptionally high FAR of 0.3672 in this study, compared to the more typical 0.0576 found in [3].

These findings suggest that the preprocessing techniques applied to the CIC UNSW-NB15 Augmented Dataset may be more effective in reducing false alarm rates. This observation could inform the development of future intrusion detection systems (IDS), where minimizing false alarms is a critical consideration.

C. Hyper-Parameterized Model Evaluation

Table 3 shows the values of the hyper-parameters selected through the optimization processes compared to the original RFC.

TABLE III
OPTIMIZATION PARAMETER VALUES

Model	max_depth	min_samples_leaf	min_samples_split	n_estimators
Random Forest	22.0	1.0	6.0	300.0
GridSearch	40.0	2.0	2.0	200.0
Optuna	20.0	3.0	7.0	195.0
Hyperopt	16.0	1.0	5.0	222.0

Table 4 shows the metrics of the models trained on the best found hyper-parameters compared to the original RFC.

TABLE IV
OPTIMIZATION MODEL EVALUATION METRICS

Model	Precision	Recall	Accuracy	F-1 Score	FAR
Random Forest	0.9451	0.9325	0.9354	0.9325	0.0259
GridSearch	0.9352	0.9294	0.9294	0.9304	0.0212
Optuna	0.9452	0.9377	0.9377	0.9371	0.0175
Hyperopt	0.9409	0.9340	0.9340	0.9326	0.0176

*1) **Grid Search**:* The best hyper-parameters found by GridSearchCV were `max_depth` = 40, `min_samples_leaf` = 2, `min_samples_split` = 2, and `n_estimators` = 200. The model trained with these hyper-parameters achieved an accuracy of 0.9294, precision of 0.9352, recall of 0.9294, and an F-1 Score of 0.9304. The false alarm rate for the GridSearchCV-optimized model was 0.0212.

*2) **Optuna**:* The best hyper-parameters found by Optuna with a Tree-structured Parzen Estimator (TPE) were `max_depth` = 20.0, `min_samples_leaf` = 3.0, `min_samples_split` = 7.0, `n_estimators` = 195.0. The model that was trained with these hyper-parameters had an accuracy of 0.9377, precision of 0.9452, recall of 0.9377, and an F-1 Score of 0.9371. The Optuna optimized model also had a low false alarm rate of 0.0175.

*3) **Hyperopt**:* The best hyper-parameters found by Hyperopt with simulated annealing were `max_depth` = 16.0, `min_samples_leaf` = 1.0, `min_samples_split` = 5.0, `n_estimators` = 222.0. The model that was trained with these hyper-parameters had an accuracy of 0.9340, precision of 0.9409, recall of 0.9340, and an F-1 Score of 0.9326. The Hyperopt optimized model also had a low false alarm rate of 0.0176.

D. Hyper-Parameterized Model Comparison

The Optuna hyper-parametrization outperformed the original Random Forest Classifier (RFC), GridSearchCV, and Hyperopt in all evaluation metrics, establishing itself as the most effective optimization approach. Notably, Optuna

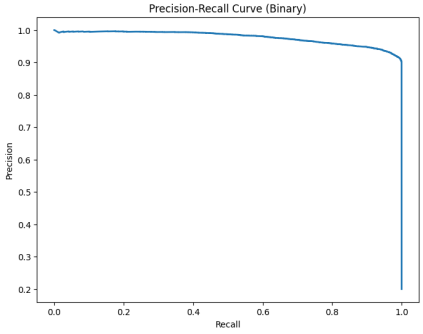
achieved the highest precision, demonstrating its ability to classify minority classes more effectively. Hyperopt emerged as the second-best optimization, surpassing GridSearchCV while maintaining comparable performance to the original RFC. GridSearchCV delivered the weakest results, performing worse than both the original RFC and the other optimization techniques.

Overall, the Bayesian optimization methods — Optuna and Hyperopt— consistently outperformed the exhaustive GridSearchCV approach. The ability of these methods to explore the hyper-parameter space beyond fixed bounds allowed for superior performance. Both Bayesian approaches produced similar configurations, indicating that high-performing hyper-parameter combinations are concentrated within specific ranges. A notable finding was that GridSearchCV selected the maximum value for `max_depth`, nearly doubling the depth used by the other methods. This suggests that the optimal `max_depth` is approximately 20, balancing the prevention of overfitting with maintaining strong predictive performance.

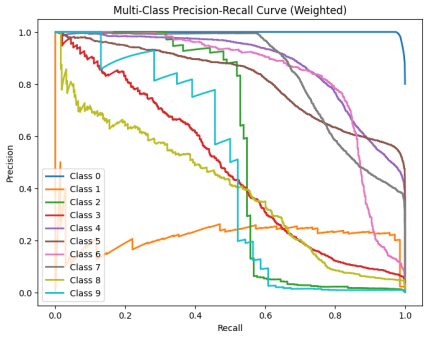
Despite these improvements, the models' overall performance still fell short of the results reported in prior studies [3]. While the hyper-parameter tuning demonstrated measurable enhancements, further refinements in the preprocessing of the CIC UNSW-NB15 Augmented Dataset are likely necessary to achieve comparable or superior results.

E. Optuna Hyper-Parametrized Trained RFC

Figure 2 contains the PRC curves for the binary and multi classifications of the original Random Forest Model.



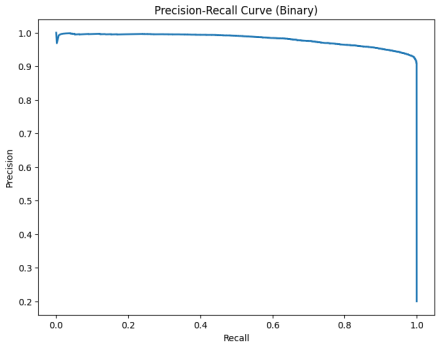
(a) Binary Precision-Recall Curve



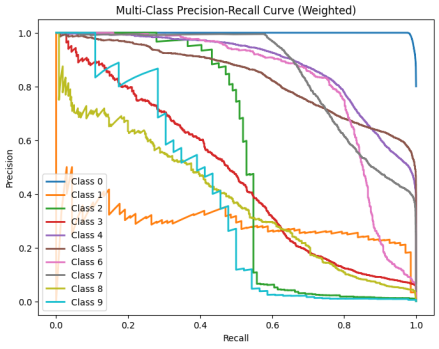
(b) Multi-Class Precision-Recall Curve

Fig. 2. Precision-Recall Curves for Original Random Forest Model

Figure 3 contains the PRC curves for the binary and multi classifications for the Optuna-optimized Random Forest Model.



(a) Binary Precision-Recall Curve



(b) Multi-Class Precision-Recall Curve

Fig. 3. Precision-Recall Curves for Optuna-optimized Random Forest Model

The Optuna-optimized Random Forest Classifier (RFC) demonstrated notable improvements over the original RFC trial. The enhancements across key metrics, coupled with similar precision-recall (PRC) curves, highlight its superior performance, particularly in boosting confidence for minority class classification. The significant improvement in the F-1 Score indicates that the Optuna-optimized RFC achieves a better balance between precision and recall.

Both RFC models performed exceptionally well in the binary classification of benign versus attack traffic. The dataset's flow statistics appear to contain discernible attack patterns that the RFC models effectively leveraged. However, the inherent class imbalance likely contributed to lower performance for certain minority classes in the multi classification PRC curve. For instance, Class 1, representing fuzzer attacks, is characterized by sporadic and random behavior. This nature likely led to correct predictions but with lower confidence.

It is anticipated that improved dataset preprocessing techniques, such as noise reduction and oversampling, could further enhance performance on the smaller classes.

V. Limitations and Future Works

The class imbalance in the CIC UNSW-NB15 Augmented Dataset posed a significant challenge, particularly for minority attack classes. While the model performed well in binary

classification, the lower confidence in correctly classifying certain attack types, such as fuzzer attacks, highlights the need for strategies like advanced oversampling techniques or class-specific weighting to mitigate this issue.

The limited computational resources of free Google Colab impacted the training process. Grid Search and Bayesian optimization methods, while effective, were restricted in the depth of hyper-parameter exploration due to computational time constraints. Access to more robust computing resources would allow for more extensive hyper-parameter tuning with an expanded search space.

To further this study, techniques such as noise reduction, feature selection, and enhanced representation of attack patterns are expected to reduce the noise-to-signal ratio and improve model generalization. The augmentation of the dataset shows promising results, but is raw and in need of further refinement to produce comparable results to [3].

VI. Conclusion

As the world becomes increasingly interconnected through the internet, cybercrime continues to evolve in both sophistication and frequency. To stay ahead of this growing threat, cybersecurity professionals must continuously adapt. Cyberattacks often leave traces in the form of network traffic statistics, and manually analyzing each record to identify malicious activity is impractical and time-consuming. However, an Intrusion Detection System (IDS) trained on a dataset of both benign and malicious web traffic can efficiently identify and alert on potential attacks at scale. In this study, we proposed a novel approach of training a Random Forest Classifier (RFC) on a diverse dataset that includes both benign and various types of malicious traffic. Our results demonstrated that an RFC with hyper-parameters optimized by Optuna outperformed all other models in terms of performance. We observed that classifiers can effectively distinguish between the characteristics of benign and malicious web traffic. Moving forward, we plan to refine our data preprocessing techniques for the CIC UNSW-NB15 Augmented Dataset, with the goal of further improving the model's accuracy and robustness.

Acknowledgment of Generative AI

We would like to acknowledge the assistance of ChatGPT in refining the professionalism and clarity of this paper. The insights provided helped improve the overall structure and tone, ensuring the language is appropriate for a scholarly audience.

References

[1] M. St. John, "Cybersecurity stats: Facts and Figures You Should Know," Forbes, https://www.forbes.com/advisor/education/it-and-tech/cybersecurity-statistics (accessed Nov. 25, 2024).

[2] J. Rundle, "AI Effect: Amazon Sees Nearly 1 Billion Threats a Day," Wall Street Journal, 2024. Available: http://proxy.library.tamu.edu/login?url=https://www.proquest.com/newspapers/ai-effect-amazon-sees-nearly-1-billion-threats/docview/3132486088/se-2.

[3] S. More, M. Idrissi, H. Mahmoud, and A. T. Asyhari, "Enhanced Intrusion Detection Systems Performance with UNSW-NB15 Data Analysis," Algorithms, vol. 17, no. 2, p. 64, Feb. 2024. Available: https://doi.org/10.3390/a17020064.

[4] Moustafa, Nour, and Jill Slay. "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)." Military Communications and Information Systems Conference (MilCIS), 2015. IEEE, 2015.

[5] H. Mohammadian, A. H. Lashkari, A. Ghorbani. "Poisoning and Evasion: Deep Learning-Based NIDS under Adversarial Attacks," 21st Annual International Conference on Privacy, Security and Trust (PST), 2024.

[6] "compute_class_weight," scikit-learn, https://scikit-learn.org/dev/modules/generated/sklearn.utils.class_weight.compute_class_weight.html (accessed Dec. 1, 2024).

[7] "DecisionTreeClassifier," scikit-learn, https://scikit-learn.org/dev/modules/generated/sklearn.tree.DecisionTreeClassifier.html (accessed Dec. 1, 2024).

[8] "RandomForestClassifier," scikit-learn, https://scikit-learn.org/dev/modules/generated/sklearn.ensemble.RandomForestClassifier.html (accessed Dec. 1, 2024).

[9] "SVC," scikit-learn, https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html (accessed Dec. 1, 2024).

[10] "LinearSVC," scikit-learn, https://scikit-learn.org/dev/modules/generated/sklearn.svm.LinearSVC.html (accessed Dec. 1, 2024).

[11] "SGDClassifier," scikit-learn, https://scikit-learn.org/1.5/modules/generated/sklearn.linear_model.SGDClassifier.html (accessed Dec. 1, 2024).

[12] "Sequential", scikit-learn, https://www.tensorflow.org/guide/keras/sequential_model (accessed Dec. 1, 2024)

[13] "GridSearchCV", scikit-learn, https://scikit-learn.org/dev/modules/generated/sklearn.model_selection.GridSearchCV.html (accessed Dec. 1, 2024)

[14] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In KDD. (accessed Dec. 1, 2024)

[15] Bergstra, J., Yamins, D., & Cox, D. (2013). Hyperopt: A Python library for model selection and hyperparameter optimization. (accessed Dec. 1, 2024)