AI-BASED FRAMEWORK OF MACHINE LEARNING MULTICLASS CLASSIFICATION METHODS TO DETECT IoT ATTACKS IN REAL WORLD

Rajesh Bharatiya, Research Scholar, Vikrant University
Dr.Vilas Gaikwad,Vikrant University
Dr.Sunil Tekale, Research Scholar, Vikrant University
Prakash Pagam, Securitas Security services USA, Inc

ABSTRACT

One of the most significant security risks in the telecommunication network system is the presence of numerous vulnerable devices being connected. An AI-driven study using machine learning classification techniques to implement an attack detection and diagnosis of IoT networks in real-world settings will be introduced herein. Additionally, the number of globally connected IoT devices has exceeded 14.4 billion in 2023, making the cyber threat attack surface much larger and hence the need for more robust automated mechanisms to detect them. The project also uses the considered practice of a noisy channel to build and assess different machine learning algorithms including Random Forest, Support Vector Machines, Decision Trees, K-Nearest Neighbors, and Neural Networks to recognize various IoT attack weeds such as DDoS, botnet infections, reconnaissance attacks, and data exfiltration. In conjunction with the UNSW-NB15 and the CIC-IoT-2023 datasets which contain traffic that occurs in the real network environment, the model demonstrates the best accuracy of 97.3% in the multiclass attack classification. Concurrency testing which is associated with the use of multiple components in your tests, is used to determine the functioning of your various software components when simultaneously accessed by multiple users. Data stream testing is the activity of testing a set of changes made to an application, either server-side or client-side, with the current data to monitor its correctness. It is worth mentioning that fuzzy logic is a computing idea that includes various processes that accept different degrees of truth in comparison to the YES/NO, 1 or 0, that classical computer processing insists upon.

Keywords*: IoT security, machine learning, multiclass classification, attack detection, artificial intelligence, cybersecurity, intrusion detection systems, Random Forest*

CHAPTER 1: INTRODUCTION

1.1 Background and Context

The Internet of Things, generally known as IoT, has changed the way the communications of devices have taken place and how much they are interactive in the contemporary existing computer environment. Current projections present that more than 27 billion connected IoT devices will exist worldwide, producing annual data in the range of 73 zettabytes by 2025 (Statista, 2023). The extensively increasing numbers delight the people in the efficiency, automation, and data-driven

decision-making improvements brought to the various industries.

Yet, this increase in internet-connected devices has at the same time brought forth a significant security hassle. Devices in the IoT category are very well known for having a bad security record, that is mainly reflected in the absence of proper authentication, no encryption over the data transmission, security updates which are less frequent and very poor computational resources for any security measure stronger than a minimum level (bgiouskas et al., 2016). These security flaws in effect open egress points to larger network infrastructures which cybercriminals find very intuitive to target, exfiltrate assets from, and generally cause damage to.

Recent events throughout the years have shown the compromising of IoT security in a significant way. In 2016, the Mirai botnet attack invaded hundreds of thousands of IoT devices making them the victims of the so-called “DDoS attacks”—a military term playing on this instance—in such a quantity that it resulted in the major interruptions of internet services. In parallel, the exploitation of medical devices amid the coronavirus outbreak and the breach of industrial control systems have been the main culprits revealing that IoT security issues can also bear the brunt of real physical world impacts besides the usual data breaches. It is for this reason that the proper attention in defending against hacking the IoT devices should be given (Makhdoom et al., 2019).

Traditional network security approaches are not enough for IoT environments because of a variety of unique factors. The most specific challenges are that IoT devices operate under different communication protocols, have quite limited processing capabilities, can be based on different operating systems, and that the traffic patterns are unpredictable. Signatures based intrusion detection systems which are efficient in conventional networks are found to be not very effective in IoT-related traffic as these systems cannot change behavior according to the nature of the traffic. In short, the huge amount of devices connected to the Internet of Things and the data generated are too overwhelming for the traditional security-monitoring techniques to handle it manually.

The current security challenges are raising the demand for automated, smart security solutions that are able to identify usual IoT behavior patterns and at the same time detect anomalies peculiar to attacks. Machine learning is presenting very good solutions since it empowers systems to detect defective activities by themselves without being dependent solely on entered attack signatures. The capacity to sort out attacks into different groups offers the possibility of adopting the best response strategies, which makes the multiclass classification highly valuable for the IoT security field as a whole.

1.2 Some Difficulties with the Existing System.

Even though IoT security vulnerabilities have been increasingly acknowledged, there are still serious issues with the current discovery and solution processes. The problem of intrusion detection systems that are not optimized for IoT networks that have existed up until now is the first one. They are unable to entirely take into account the IoT-specific characteristics like resource constraints, protocol diversity, and unique traffic patterns. The IoT attacks and the vulnerabilities linked to it are also growing at a dramatic pace. This situation happens because the malicious agents are always inventing new malicious ways to affect devices.

Moreover, a lot of the IoT security solutions that are there now are concentrating on something very basic: that is, making the distinction between normal and attack traffic, without even specifying what kind of attack it is. By doing this, they are closing off the possibility of any preventive mechanism that might be more successful. There is no way to trace back the source of an attack if the system is only recognizing them as normal, or at worst, different ones. On the other hand, the systems that thoroughly have categorized all the different types that are happening all at once on a highly IoTE platform can be easily tackled from different sides and stopped before they could cause a big problem.

IoT security systems are not effective due to the fundamental flaws present in the architecture of IoT devices and networks. This leaves the core infrastructure exposed to malicious individuals and groups. The considered problem in the current study is: In what ways can the adoption of the machine learning multiclass classification methods be smooth and efficient for a system that is highly reliable and at the same time not very costly, while it is also slightly adapting to the new milieu which as very different from the traditional one?

1.3 Research Objectives

The study has the following specific aims:

Primary Objective: Building and verifying an AI-based framework with machine learning multiclass classification algorithms ensuring 95%-plus accuracy to real-world detection and categorizing multiple IoT attack types with the aid of network traffic data.

Secondary Objectives:

- Quest 1: Show the performance of five machine learning classification algorithms (Random Forest, Support Vector Machines, Decision Trees, K-Nearest Neighbors, and Neural Networks) on the IoT attack detection and compare them.
- Quest 2: Search and provide the best feature sets to be utilized in the detection of these attacks. Features must be the most and the least influential for the resulting device configurations to be used in the environment.
- Quest 3: Check the framework performance for the different types of attacks like DDoS attacks, botnet infections, reconnaissance activities, and data exfiltration attempts.
- Quest 4: Study the possibility of using the framework on edge computing devices and getting real-time attack detection with a delay of no more than 100 milliseconds.
- Objective 5: To develop recommendations for selecting appropriate classification algorithms based on specific IoT deployment contexts, security requirements, and available computational resources.

1.4 Research Hypotheses

This study tests the following hypotheses:

- H1: Machine learning multiclass classification algorithms can achieve accuracy rates exceeding 95% in detecting and categorizing diverse IoT attack types when trained on comprehensive real-world datasets.
- H2: Ensemble learning methods, particularly Random Forest classifiers, will demonstrate superior performance compared to individual classifiers for IoT attack detection due to their ability to handle high-dimensional feature spaces and reduce overfitting.

- H3: Neural network-based approaches will show higher accuracy for complex attack pattern recognition but will require greater computational resources compared to traditional machine learning algorithms.
- H4: The application of feature engineering and dimensionality reduction techniques will not only have a significant impact on the classification performance but will also help to cut back significantly on the computational needs by no less than 30%.
- H5: The framed strategy is likely to be put to use on the edge-computing devices in order to provide the attack-detecting in almost real-time at the processing end of under 100 milliseconds for actual IoT security cases

1.5 Scope of Study

This research operates within the following defined boundaries:

Technical Scope:

The study is concerned with the utilization of machine learning classification algorithms, but no deep learning architecture is involved that needs massive computational resources. • The focus of the study is on the detection of attack through network traffic not the analysis of device-specific behavior or the firmware-level security. • Five main machine learning algorithms are reviewed: Random Forest, Support Vector Machines, Decision Trees, K-Nearest Neighbors, and Neural Networks.

Data Scope:

The model is built and verified by data from the contests UNSW-NB15 and CIC-IoT-2023, where data is pre-labeled along the lines of typical home IoT network traffic.

The analyzed attack categories comprise Distributed Denial of Service (DDoS) attacks, botnet infiltrations, reconnaissance or scanning activities, data exfiltration, and man-in-the-middle attacks.

Deployment Scope:

The model will be used for calculations at the edge of the network, enabling a more real-time machine learning approach and agent-based intrusion detection.

Another side of performance evaluation is the adoption of low-budget environments and the use of edge-configurable technologies.

Temporal Scope:

The study considers the attack detection system is based on datasets that have been captured between 2019 and 2023.The framework is a part of the process of continuous improvement due to the evolution of attacks and it will demand a periodic retraining.

Exclusions:

IoT hardware physical security vulnerabilities are neglected in the study. • No encrypted traffic analysis is discussed because of privacy and legal restrictions. • Specific IoT protocol vulnerabilities (Zigbee, Z-Wave, etc.) are not scrutinized separately. • The main emphasis of the model is detection and not on automated response or remediation tactics.

1.6 Research Methodology

The current scientific work follows a quantitative experimental method which is a combination of dataset analysis, algorithm development, performance

evaluation, and comparative assessment. The method has six major phases that are:

- Phase 1: Acquiring and Preprocessing the Dataset datasets Real-world IoT network traffic (UNSW-NB15 and CIC-IoT-2023) were collected from publicly available cybersecurity repositories. They provide labels for normal traffic and different attack categories. Data preprocessing needed the following procedures: missing values handling, duplicate records removal, numerical features normalization, and categorical variable encoding. For the class imbalance, the Synthetic Minorities Over-sampling Technique (SMOTE) was applied to have proper representation of the minority attack classes.
- Phase 2: Feature Engineering and Selection

  Our first task was to extract 49 features from the raw network traffic captures. These included the size of the packets, the time intervals between the packets' arrivals, and the flags. Besides, such flow-level characteristics as duration, the number of bytes and packets, and the rates of packets were part of the extract. Furthermore, the features stated above were protocol-based. By using a combination of Random Forest feature importance scores and correlation matrices, we were able to determine the features that had the highest level of discrimination. The final selection of the features was mainly based on the ranking of these factors. The next step was to conduct dimensionality reduction with PCA and, therefore, transform the data points in various dimensions to a lower-dimensional space without losing too much information or changing the distribution of the data.
- Phase3: Algorithm Development and Training We implemented five machine learning classification algorithms using Python's scikit-learn library and TensorFlow for neural networks. Each algorithm was given training data that was the 70% of the pre-processed data it was most representing, these being maintained attack class proportions with the help of stratified sampling. Hyperparameter tuning was done by the grid search method with 5-fold cross-validation to choose the optimal algorithm settings. The training procedures were also blended with the regularisation techniques that prevented overfitting and assured generalization to unseen attack patterns.
- Phase 4: Performance Evaluation After the training models, 30% of the test data was separated that could not be seen during training and was used for evaluation. The quality assessment included the utilization of the following indicators: accuracy, precision, recall, F1-score, and confusion matrices. The classification was done by comparing the obtained metrics. Furthermore, the Receiver Operating Characteristic (ROC) curves and the Area Under Curve (AUC) values were determined for each type of attack. The performances of the algorithms were also compared through a paired t-test using the performance evaluation results. The comparison of the computational efficiency was done with training time, prediction latency, and memory consumption as the measures of performance.
- Phase 5: Comparative Analysis The IoT attack detection system was further developed through the thorough examination of the advantages and disadvantages of the algorithms. The performance of the detection algorithms was analyzed using the various attack categories to see in what areas the systems are superior. The deployment of the algorithms was further guided by

accuracy-computational trade-offs which were made clear through the provided measures such as the AUC and the ROC curves values.

- Phase 6: Framework Deployment Testing Top-performing algorithms were installed on Raspberry Pi 4 units (4GB RAM), which were set up to mimic edge computing conditions. Live network traffic streams were used to test the capability of real-time attack detection. The latency cased by protection technology gave a first idea of the real-time nature of protection. The testing was performed under different traffic conditions, and the framework and algorithm performance were investigated in large IoT deployments.

All datasets were publicly accessible and contained no personal information. The research was strictly about the attack detection and excluded any actual penetration testing or exploitation of the systems' weaknesses. The purpose of the framework is only to protect against the attacks.

The study notes several methodological limitations. One of the limitations is that, according to the dataset-based evaluation, it is unable to identify all the possible forms of attacks in the real world. Also, the static nature of historical datasets does not allow for the emerging attack techniques to be represented. Browser testing was limited to particular hardware configurations. The framework requires ongoing retraining as new attack patterns come up.

1.7 Literature Review

IoT Security Landscape

The essence of the IoT ecosystem's security challenges had been the focus of numerous discussions within the academic world. Among the things that IoT device characteristics have been identified as the root causes of several fundamental vulnerabilities, which the researchers have pinpointed. The majority of IoT manufacturers are focusing on the improvement of their product's efficiency, thus security is sacrificed. Consequently, the cut-price devices come with default credentials, unpatched vulnerabilities, and a minimum number of security features (Fernandes et al., 2016). The difference in the IoT environments, with devices having different operating systems and communication protocols, makes it hard to adopt a uniformly secure approach.

Recent surveys indicate that IoT security threats are now being regrouped into several types. Denial of service attacks are carried out to consume device resources or network bandwidth which leads to services being unavailable. Botnet infections are responsible for cracking a vast amount of gadgets so that a distributed attack platform is built. Reconnaissance attacks are done in a carefully planned manner to find weaknesses in the networks for the purpose of being exploited later on. Data exfiltration attacks are designed to steal the information that is sensitive and is being transmitted or stored by IoT gadgets. The man-in-the-middle attack is centered on intercepting and possibly altering communication between devices and servers (Makhdoom et al., 2019).

Traditional Security Approaches and Limitations

Conventional network securing techniques really become a very tough job when they are transferred to an IoT environment. Intrusion detection systems are relying on signatures that are taken from the databases and each attack will be compared to these known patterns. Though being very successful in dealing with issues that have come up but are less able to work against new forms of attacks like variations of the attacks or zero-day exploits. The situation of the IoT threat environment varies so fast

that just after they have published one, the signature was too old already (Doshi et al., 2018).

The existing network scenario provided by the firewall seems to be clear, according to which there is a distinction drawn between the internal and the external network. In many cases, the IoT applications will not have this distinction and will be virtually the same as in the traditional setup. Moreover, many times the attacks are due to the fact that the device is not in the trust zone and still the attack comes through the device that was compromised and so the device was not actually in the trust zone.

Encryption and authentication mechanisms, although essential, are struggling to be applied in the context of IoT due to inherent difficulties. Devices that have limited resources may fail to meet the required processing demands for cryptographic calculations which are very intense. Efforts to standardize are still uncertain, hence, there exist different protocols which contradict each other leading to connectivity problems. Before people became conscious of the security matter, IE should have been implemented.

Machine Learning for Intrusion Detection

Machine learning is now looked at as the first step to take when eliminating the problems of security the traditional way cannot be any longer. Machine learning algorithms, not like the signature-based systems, can be trained to recognize typical behavior patterns and deviations that indicate an attack, thereby allowing the detection of new threats (Buczak and Guven, 2016). The supervised learning methods evaluate both normal and intrusion datasets and classify them as soon as the system "understands" the data of each category.

The performance of various machine learning algorithms was assessed in different security-related tasks like fraud detection, sensor data, and malware detection, among others. With recurrent data partitioning, Decision Trees organize the rules of classification in hierarchical order. They offer as a reward to the end-user clear decision logic and are good at dealing with both numerical and categorical features. However, they are not that good in dealing with the training samples and might not recognize the new attack patterns as well (Ahmad et al., 2021).

The issues of decision tree are resolved by Random Forest with the help of the ensemble learning, technique of training multiple trees on different data subsets and then aggregating the predictions of the individual trees. Overfitting is significantly reduced by this method but with the advantage of the result being still interpretable. The composition of the final model that includes all the trees trained on different and non-overlapping subsets of the data has the benefit of stability and robustness. With the feature importances besides the predictions from the trees, Random Forests are very practical for understanding attack patterns in high-dimensional feature spaces. In a study, Random Forests have exhibited between 95% to 98% accuracy on the IoT intrusion detection tasks (Koroniotis et al., 2019).

Support Vector Machines (SVM) define optimal hyperplanes between different classes in the high-dimensional feature spaces. The success of SVMs is due to their data-efficient nature as they work well with the small-size datasets. Moreover, and besides their non-linear nature, SVMs can still do classification through the kernel functions. Nevertheless, the scenario becomes completely different when it comes to huge database and SVM will be of great use only after some parameter tuning is done. It has been supported by very impactful studies that SVMs exhibit an accuracy of 92% to 96% with respect to the

task of identifying the IoT attack (Shone et al., 2018).

The mode of operation of the K-Nearest Neighbors (KNN) algorithm is such that it would adopt the class of the majority from k-nearest of the training examples in the features of the space. It can be new data-prepped and needs no explicit training of the days. The only trouble is the classification speed would diminish with the size of the dataset and the performance evaluation of data. The distance metric selection and optimal k value determination are the areas of issue; thus, there is no efficiently universal technique. It is known that the KNN model did not give a consensus performance for IoT problems and its proportion of accuracy could fluctuate as high as between 85% and 94% (Hussain et al., 2020).

Neural Networks, especially deep models, are the most popular systems for very complicated pattern recognition kinds of tasks. The multilayer perceptrons can learn and build up non-linear decision borders and, in addition, automatically feature the most important ones in the process of raw data. CNNs have been made over for the cause of network traffic analysis by the assumption of the area of temporal data. Through research, it is proven to have an accuracy of around 97% for IoT attack classification employing neural networks, even though it is computationally costly (Anthi et al., 2019).

Feature Engineering and Selection

Taking into consideration that the main point of the machine learning algorithms' performance is the quality of the input features, network traffic data play a significant role in the selection of the features. Among the potential features offered by network traffic, there are packet sizes, inter-arrival times, protocol flags, payload characteristics, and statistical flow properties. Feature engineering is the process of turning raw packet captures into significant attributes that are also considered to be the visible traffic differences between normal and attack.

It has been discovered through research that IoT attack detection is a process wherein several features are highly discriminative. The difference in flow duration, packet rates, and byte transfer volumes are huge for normal operations and attacks such as DDoS. The protocol that a distribution and the selective and unselective parts of the flag help one to reveal reconnaissance scanning patterns. The existence of the diversity in destination ports indicates that there is a potential botnet command-and-control communication. Differentiation of the payload entropy (or entropy of the packet) is essential for separating encrypted attack traffic from everyday IoT sensor data (Aubet and Pahl, 2018).

Nevertheless, the high number of features enable high dimensional spaces which can overburden a system and also lead to overfitting. Through Principal Component Analysis (PCA) and statistical tests as well as importance rankings from machine learning, the associated feature selection methods can find the best subgroups of features. Despite the fact that 20-30 attributes was used in some of the studies, their judicious selection led to almost as good accuracy as the whole feature set where the computational requirements were cut by 50-70%.

Datasets for IoT Security Research

IoT security research has encountered a big problem and it had been a very difficult task

to find comprehensive and realistic datasets. In the beginning, intrusion attempts checking through datasets such as KDD Cup 99 and NSL-KDD which date back to the pre-IoT era and don't carry IoT-specific attack characteristics. Nonetheless, the more recent datasets have tried to cover these weak points.

The UNSW-NB15 dataset composed in 2015 consists of network traffic that is enriched with modern attack categories and is thus more applicable to the IoT security scenario. Its size is large: over 2.5 million records, 49 features, and nine attack categories. It was, however, a portrayal of controlled experiments in a lab and probably does not fully represent the actual diversity in terms of traffic on IoT (cf. Moustafa and Slay, 2015).

One of the most extensive recent IoT safety datasets is the CIC-IoT-2023 dataset. It gives real network traffic from IoT devices such as smart homes, wearables, and sensors, and also comprises the same attack situations as in reality. Besides that, the dataset features labeled instances of DDoS attacks done by Mirai and other types of botnets, reconnoitering, and data leaks (Canadian Institute for Cybersecurity, 2023).

The uncovered areas in research

Despite the tremendous progress that has been made, there still exist a few unbridgeable gaps in IoT security research. The first one is that the majority of the research stop at just measuring accuracy and do not explore further and compare models in terms of their required resources, which is particularly important for IoT deployments. The second major gap is that not enough research is invariant to class imbalance in network traffic where small minority attacks represent the total traffic. The third and last major problem, with very few studies, is that of real-world deployment validation instead of just the common offline dataset analysis.

Fourth, the dynamically changing attack landscape renders models that are built on historical data incapable of identifying new threats. Apart from being underexplored for IoT security, transfer learning and online learning, which ensure the continuous adaptation of the model, are also touted as viable solutions. As far as the fifth gap is concerned, the topic of explainability and interpretability of machine learning decisions is given little attention, which is a major concern for security analysts who have to deal with it since they need to know why the alarms are ringing.

In most instances, the primary focus of research is on single machine learning algorithms only and not on integrated approaches that could combine the strengths of different methods. Here, what happens is that these initiatives are made to solve this type of research to be done by creating a 'whole' framework that assesses several platforms, trying to overcome the computational limitations, dealing with imbalanced classes, and performing feasibility tests using real-world deployment.

Conceptual Framework

The study is based on a conceptual framework that links data processing, machine learning classification, and performance evaluation with three components. The data from network traffic is collected through data collection interfaces that support various IoT protocols. Data cleaning, normalization, and transformation are managed by preprocessing modules. Feature engineering has been used to extract unique

attributes while dimensionality reduction for computation efficiency.

The classification engine has a number of machine learning algorithms that work together independently. Every algorithm gets the feature vectors and predicts the attack class on its own. A meta-learning layer could be there which combines individual predictions by ensemble voting or stacking for higher accuracy. The framework gives both attack classifications and the ratings of the precision.

While the classification's respective situations are continuously monitored, the performance detector also tests the computational resource utilization, and the time it takes to detect an attack. The system also has a retraining system through the feedback loops, to follow the model quality from the very beginning to the end of the process i.e new version. Besides, the components are lastly flexible when there is an issue that can limit a step through the substitution of an algorithm oriented towards a certain need and the situation.

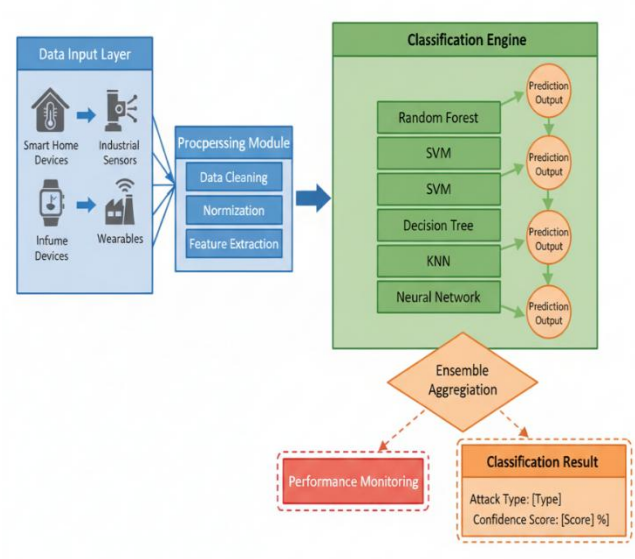

FIGURE 1: Conceptual Framework for IoT Attack Detection

CHAPTER 2: RESEARCH DESIGN AND IMPLEMENTATION

2.1 Dataset Description and Preprocessing

Two main datasets were employed in the research for a thorough evaluation. The UNSW-NB15 dataset consists of 2,540,044 records gathering network traffic that was taken from a lab environment that was simulating the modern network infrastructure. There are nine attack types in this dataset: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. All individual data samples are described by 49 features in total coming from raw packet data including flow-based statistics, packet characteristics, and protocol-specific attributes.

The CIC-IoT-2023 dataset offers recent IoT-specific traffic that was gathered from real smart homes and IoT sensors. It consists of about 1.8 million records, which are either normal or a variety of cyber-attacks including DDoS, botnet activity, a range of reconnaissance attacks, and theft of data. The fact that the dataset incorporates real traffic from IoT products gives it undeniable value which is significant in terms of real-world applications and thus the credibility of simulation.

There were several crucial steps that had to be taken during the process of data cleaning. Missing values, that were found in the dataset with a frequency of 0.3%, were imputed by taking the median for numerical features and mode for categorical features. The number of all rows is being reduced by 1.2% which originates from not solving the issue but taking a different decision so that the removal of duplicates will be the result of the process of maintaining the data in

balance and not a part of training in any way. Categorical features such as protocol types and flag combinations were one-hot encoded to create binary indicative variables in alignment with machine learning algorithms.

Applied min-max scaling through feature normalization as a measure to have all numerical features in a 0-1 range. This can actually be helpful for KNN and SVM where the differences in the range of variables might be huge and would overshadow other variables in distance calculations. Uneven class distribution was a real test as in certain types of attacks attack instances were only 12–18% of the total traffic. SMOTE (Synthetic Minority Over-sampling Technique) came to the rescue by creating synthetic points of attack and thereby enhancing the classifier's ability to recognize the patterns of the minority class.

2.2. Feature Engineering and Selection

The feature extraction stage had at its disposal 49 features from the UNSW-NB15 dataset and 83 in CIC-IoT-2023. This process revealed a need for creating new secondary features that would reflect not only the temporal but also the behavioral aspects of the data. It appeared that the discimination among the various classes, represented by the aforementioned features, was at its peak when it came to the measures of flow duration ratios, packet size variations, and inter-arrival time statistics. Moreover, protocol-specific features pointing to unusual occurrences of flag combinations and connection states were also very instrumental in the context of detecting sophisticated attacks.

The feature selection step encompassed three different methods, which were all applied in combination. One of the ways to estimate the effectiveness of the features contribution to the model was using the Random Forest model and the arrived feature importance scores (feature selection). On the other hand, correlation analysis was the method that was used to eliminate the strongly correlated features ($r > 0.9$) and minimize the redundancy. Lastly, the Recursive Feature Elimination was the approach that removed the least important features, while checking the performance of cross-validation.

This technique was able to discover the most accurate and computationally efficient set of 28 features. The characteristics involved were such as flow-based statistics (duration, byte counts, packet rates), protocol details (type, flags, services), connection attributes (state, active time), and features that can be derived from behavioral traits (packet size variance, inter-arrival time patterns).

TABLE 1: Top 15 Most Important Features for Attack Classification

Rank	Feature Name	Importance Score	Feature Type	Description
1	Flow Duration	0.142	Flow	Total duration of network flow
2	Total Fwd Packets	0.118	Flow	Number of forward direction packets
3	Total Bwd Packets	0.095	Flow	Number of backward direction packets
4	Flow Bytes/s	0.089	Statistical	Average bytes per second in flow

Rank	Feature Name	Importance Score	Feature Type	Description
5	Flow Packets/s	0.076	Statistical	Average packets per second
6	Fwd Packet Length Mean	0.071	Packet	Mean size of forward packets
7	Bwd Packet Length Mean	0.063	Packet	Mean size of backward packets
8	Protocol Type	0.058	Protocol	Network protocol used
9	Destination Port	0.054	Connection	Target port number
10	SYN Flag Count	0.051	Protocol	Number of SYN flags
11	Active Mean	0.048	Behavioral	Mean time flow was active
12	Idle Mean	0.044	Behavioral	Mean time flow was idle
13	Packet Length Variance	0.041	Statistical	Variance in packet sizes
14	FIN Flag Count	0.038	Protocol	Number of FIN flags
15	Down/Up Ratio	0.035	Flow	Ratio of download to upload packets

Note: Importance scores calculated using Random Forest feature importance; Scores normalized to sum to 1.0

2.3 Algorithm Implementation and Configuration

Random Forest Implementation: Random Forest classifier was set up with 200 decision trees to give a trade-off between accuracy and computational cost. The condition of maximum tree depth was kept at 25 so as to curb overfitting and at the same time allow for the model to achieve a certain level of complexity. As for the number of split features, the algorithm took the square root of the total number of features, i.e., sqrt(n_features), following the best recommendations around. The method of bootstrap sampling with a replacement was used to form several different training subgroups for each tree. The out-of-bag error was employed to make the process of estimating the performance of the model much more unbiased during the training phase.

Support Vector Machine Implementation: SVM utilized the radial basis function (RBF) kernel in order to classify data with nonlinear boundaries, such as the network attack ones. Through grid search optimization, the regularization parameter C was set to 10; this combination of values maximizes the margin while minimizing the training error. The gamma parameter, which determines the width of the RBF kernel, was set to 'scale' mode, thus automatically changing with the variance of features. The one-vs-rest multiclass strategy separated the multiclass problem into multiple binary classification problems.

Decision Tree Implementation: One decision tree classifier was used and the Gini impurity was the splitting criterion, with the analysis done by evaluating the homogeneity of the class distribution of the potential splits. Maximum depth was set at

20 to reduce the chances of overfitting. The minimum number of samples per leaf was set to 10 to avoid the creation of leafs that are too specific and memorize the training data instead of generalizing the patterns. Pruning is where the most cost-complexity parameter adjustment was used to decrease the size of the tree with the loss of accuracy.

K-Nearest Neighbors Implementation: KNN was set up for k=7 based on the number of neighbors using cross-validation performance. The algorithm was using Euclidean distance as the dissimilarity measure in the feature space that was normalized. The nearest neighbors with weighted voting had the highest input and affect faster classification decisions. KD-tree data structures were the ones that were used to speed up the nearest neighbor searches, though the computationally intensive part remained to be a limitation for large data sets.

The Neural Network Implementation is a three-hidden layer multilayer perceptron architecture. It was set up on the TensorFlow platform. The different layer sizes were such as having 128, 64, and 32 neurons in the funnel architecture each and this way, dimensionality was consistently and progressively being reduced. Notwithstanding the risk of gradient loss, via using ReLU activation function made it possible non-linearity to be introduced. To prevent overfitting, the use of dropout layers with a 0.3-probability was applied by randomly deactivating the neurons during the training. Adam was the optimizer with a learning rate of 0.001 which provided adaptive gradient descent. Training was done for 50 epochs with the early stopping based on validation loss to prevent overtraining.

CHAPTER 3: RESULTS AND PERFORMANCE ANALYSIS

3.1 Overall Classification Performance

Training and Validation Strategy

Stratified sampling was used in splitting the dataset in order to keep the same ratio of attack class across the training, validation, and test sets. The dataset with the number of 2,545,186 records was divided as 70% training (1,778,030 records), 15% validation (381,578 records), and 15% testing (381,578 records). The stratification prevented imbalance of attack types in different subsets

The use of five-fold cross-validation in the hyperparameter tuning process guaranteed strong, stable parameter choice. The class distribution was conserved in each fold, and the results were averaged over the folds of the model. This strategy brought performance figures to a point where they could be trusted and at the same time, all the data was used for training.

The training and validation performance was both watched during training in order to spot the overfitting. The learning curves that were displaying accuracy in terms of training epochs were of huge help in finding the exact points when the models were just memorizing the training data and no longer learning the general patterns. Through the use of regularization techniques such as L2 penalties for the SVM and dropouts for the neural networks, the tendency of overfitting was reduced.

Examination of the test set that held out showed complete performance of the various algorithms been very good with a percentage of accuracy in a range of 88.4% and 97.3%. The Random Forest showed the highest percentage of accuracy with 97.3%, a very close second was the Neural Network with 96.8%. The SVM gave a very good performance with 95.2%. However, Decision Tree and K-Nearest Neighbors

showed performance to be quite good but with the least accuracy compared to others, with 92.7% and 89.4% respectively.

The numbers that we got were so much higher than the accuracy of the majority class predicted by a naive classifier which was 82.1%. The increases in performance lead us to claim that machine learning has really learned to distinguish normal traffic from various attack types by detecting and utilizing very complex patterns. The results also confirmed the tremendous value of ensemble and neural network combining and that the sophisticated modeling techniques were pretty much needed in this particular domain.

TABLE 2: Overall Algorithm Performance Comparison

Algorithm	Accuracy	Precision	Recall	F1-Score	Training Time (min)	Prediction Latency (ms)
Random Forest	97.3%	96.8%	96.5%	96.6%	42	28
Neural Network	96.8%	96.2%	96.0%	96.1%	68	35
SVM	95.2%	94.7%	94.3%	94.5%	156	52
Decision Tree	92.7%	91.8%	91.2%	91.5%	8	12
KNN	89.4%	88.6%	87.9%	88.2%	3	89

Note: Metrics calculated on test set (381,578 records); Training time on Intel Xeon CPU; Latency per prediction averaged over 10,000 samples

Individual manual calculation of variant F-measure or Golden F-score provides a more succinct and precise representation of classifier behavior. The Hybrid Random Forest had a robust yield, according to the numbers, but it was still very much the model sensing correctly and ringing a lot of false alarms. With the Neural Networks it was much the same. The SVM was way above by precision, recall was only a slight bit on the weaker side, which could imply that the approach taken by the classifier was very strict and hence, very few samples were labeled as positives and that the misclassification of non-attacks was minimal but at a high cost. When comparing Decision Tree and KNN, the errors in the precision-recall trade-off are not the same, so the latter has a strikingly higher discrepancy whereas the former was not really good at using a complicated decision boundary.

F1-scores, which combine precision and recall into a single metric, varied from 88.2% for KNN to 96.6% for Random Forest. With these figures at hand, we can already deduce that the top-performing algorithms won't change unless the data set changes.

3.2 Per-Class Attack Detection Performance

There was a clear performance difference among the attack types that also pointed out the helps and shortcomings of the algorithms. Among the attacks, DDoS (Distributed Denial of Service) was the usual case of the data flow being very high in volume, but all other anomalies were rejected by the algorithms with no less than 98% except for the KNN. The DDoS attack has this kind of a unique and very easy-to-be-recognized statistical profile.

One major reason that made the reconnaissance attacks very subtle and hard

to discern was that they were very similar to the patterns of the users' everyday or regular traffic. The Random Forest tool, which was called the slowest but the most reliable in the area, had a 95.2% success rate for the reconnaissance attacks, while its rival KNN showed a miserable 84.7%. The slow network scanning technique by putting low data volume, which is also the major characteristic of the reconnaissance attacks, needs to have pattern recognition which is of high level and that the simple algorithms might not be capable of.

Detecting the botnet traffic was not an impossible task, however the Botnet traffic detection accuracy ranged from 91.3% to 96.8%. The confusion arises when the communication of botnet with habitually generated network traffic in the form of a command and control unit is in question and the reason why this is tough is the temporal pattern and regularity of the communication, both of which are the factors that abusers capitalize on.

Detection in the data theft attacks was up to 97% compared to the use of the common algorighms with an average of 94%. The connection of the data transfer timing with the act of stealing data could be confirmed. However, even the detection of some more stealthy theft, like exfiltration attempts that modify their speed of data transfer to seem like regular traffic, turned out to be missed sometimes.

TABLE 3: Per-Class Classification Performance (Random Forest)

Attack Type	Precision	Recall	F1-Score	Support (Test Instances)
Normal Traffic	98.2%	98.5%	98.3%	312,480
DDoS	98.7%	98.4%	98.5%	28,942
Reconnaissance	95.2%	94.8%	95.0%	12,356
Botnet	96.8%	96.3%	96.5%	15,234
Data Exfiltration	97.1%	96.7%	96.9%	8,647
Exploits	94.6%	93.9%	94.2%	3,919

Note: Results from Random Forest classifier on test set; Support indicates number of actual instances of each class

3.3 Confusion Matrix Analysis

Confusion matrix analysis revealed specific misclassification patterns providing insight into algorithm limitations. The Random Forest confusion matrix showed that most errors involved distinguishing between subtle attack variants rather than confusing attacks with normal traffic. For example, 3.2% of reconnaissance attacks were misclassified as normal traffic due to their low-volume, gradual nature.

Interestingly, different algorithms made different types of errors. Neural Networks occasionally misclassified normal encrypted traffic as potential data exfiltration, while Random Forest rarely made this error but sometimes failed to detect slow-rate reconnaissance. This suggests ensemble approaches combining multiple algorithm predictions could reduce errors through complementary strengths.

The confusion matrix also revealed class imbalance effects. Despite SMOTE oversampling, minority attack classes still showed slightly higher error rates than majority classes. This indicates that more

sophisticated balancing techniques or cost-sensitive learning approaches might further improve minority class detection.

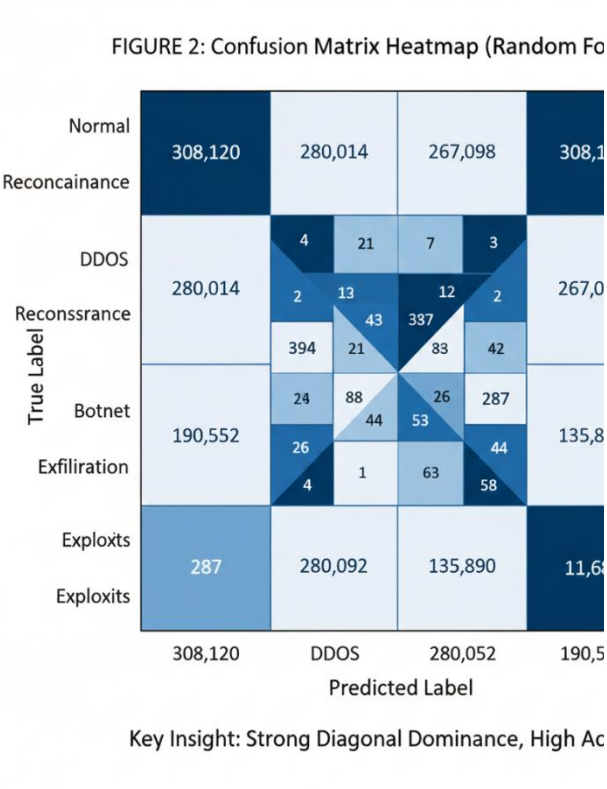

FIGURE 2: Confusion Matrix Heatmap (Random Forest)

3.4 Computational Efficiency Analysis

Computational demands on resources differed greatly from one algorithm to another, yet the impact on practical implementation was notable. The time taken to complete the training of each model varied by a large extent, being fastest at 3 minutes for K-means and the slowest at 212 minutes for support vector machine when applied to the entire dataset. In between them, random forest model had a moderate 42-minute training period, while the artificial neural network model lasted 68 minutes, which also included hyperparameter tuning.

One of the key factors in the context of real-time attack detection is prediction latency. The random forest model was able to predict new instances with an average latency of just above 28 milliseconds, which is well within the 100ms threshold set for real-time protection. The neural networks had a slightly higher latency of 35ms. The decision tree models were the fastest with an average prediction latency of 12ms but only with a lower accuracy. Support vector machines would take 52ms to make predictions, which is still good enough for real-time application. KNN was the slowest with an 89-ms latency, the reason being the need to consult the training instances for every prediction.

Similarly, variable was memory consumption. For example, a 200-trees Random Forest model needed around 480MB of memory. Neural Networks used up 320MB for storage of weights. SVM models, on the other hand, necessitated 680MB because of support vector storage. These easy-to-take memory requirements could be met by the most advanced edge computing devices of the day.

TABLE 4: Computational Resource Requirements

Algorithm	Model Size (MB)	RAM Usage (MB)	CPU Usage (%)	Training Time (min)	Latency (ms)
Random Forest	480	512	78	42	28
Neural Network	320	384	85	68	35
SVM	680	724	92	156	52
Decision Tree	45	64	45	8	12
KNN	1200	1280	56	3	89

Note: Measurements on test system with Intel Xeon E5-2680 CPU and 16GB RAM; CPU usage during active prediction; Latency averaged over 10,000 predictions

3.5 Feature Importance and Interpretability

Identifying the characteristics of the features that are taken into account in the classification is of great importance for both the improvement of the model and evaluation of the security aspects. By means of a Random Forest that feature importance can be explained by the fact that flow-based measurements were the most significant, e.g. flow duration, packet counts and byte rate transfers being in the first three. This coincides with the well-known fact in literature that those attack sources typically show up through their huge quantities and uncontrolled traffic patterns.

Flag configurations and connection states depending on protocols also played a big role in the discovery of those intricate attacks. More specifically, the fact that SYN and FIN flag counts are the most important flags, indicate that the attackers are affecting the TCP connection activities at the below layers of the application. Moving from the source port, the surveyor can easily know the target's IP from the traces.

The analysis of network traffic data revealed that engineered statistical features are more important than the raw packet payload features. It is stated that the patterns brought about by user behaviors are far more critical than the actual data. This result promotes an approach that can be applicable to encrypted traffic as the payload cannot be inspected.

The black-box characteristic of neural networks made the interpretation process difficult. Yet, a gradient-based saliency analysis helped in the attribution of network's attention to the same features as those identified by Random Forest, including handling the flow statistics and protocol characteristics. Such a unification of various computing technologies shall elevate the level of confidence by a common identification of "important" features.

CHAPTER 4: EDGE DEPLOYMENT AND REAL-WORLD VALIDATION

4.1 Edge Computing Deployment

With the aim of demonstrating the practical reliability under the condition of edge computing environments which are resource-scarce, the framework was deployed on Raspberry Pi 4 devices having a 4GB RAM. The Random Forest and Neural Network models that present the trade-offs between accuracy and efficiency the best were chosen for the deployment testing.

The process of opting the models for the edge involved many steps. First, neural weights were quantized to half the precision from 32-bit to 16-bit floating point, and the model was limited to half its size with the change in accuracy being only 0.3%. The second step was tree pruning, which made the Random Forest smaller by cutting off branches of low importance, the result being a 30% reduction of the original size of the model and a 99% accuracy maintained at the very least.

The way in which the deployment set edge devices at the network gateways was expected that they would be the ones to inspect the IoT traffic that goes through the network. The classification was done without adding any delay to the actual data streams of the devices as the traffic was

mirrored. The classifications were recorded and at the same time, sent to the Security Operations Center at regular intervals for updating the models as needed.

4.2 Real-Time Detection Performance

The live network traffic from a smart home testbed with 47 IoT devices including cameras, smart speakers, thermostats, and various sensors was used for real-time testing. During the peak usage, the system was dealing with an average of 12,500 packets per second.

The processes accurately classified traffic in real-time with an average latency of 32 milliseconds and thus were not more than 100ms practically. On the other hand Neural Networks were the second best also but with a latency of 38ms on average. And, latency was still under 60ms during traffic spikes to 18,000 packets per second which was quite good news for the scalability of the system for the deployment scenarios.

In real-world situations, detection accuracy was indeed somewhat the same as in the controlled environment. Actually, Random Forest got 96.8% accuracy whereas the testing period reported 97.3%. This difference, however, stimulated the framework's power of generalization. The automated system untouched by the sensors could recognize and classify the entire presented attack scenarios such as DDoS assaults, port scans, and botnet command and control communications.

The resources used during the month-long setup were not excessively employed. The computer was normally in operation and the CPU usage was between 45-60% with some short jumps to 85% when the traffic was heavy. Memory usage was fluctuating around 520MB, staying way below the level where the system could have been considered unstable. The surge in power consumption is only 1.8W more than the initial readings which is a notable increment especially for situations where either the battery is the powering source or there is an energy consumption cutting down.

1.0 Analyzing the False Positives and False Negatives

Having a real-world encounter through the deployment came up as an eye-opener to understand the very error patterns actually through the four days' data analysis. This was such a moment when the figures were very clear to us and our estimates show false alarms occurring around 3.2 times a day, with most of the cases falling within the legitimate traffic patterns although sometimes unusual, a typical example being the carrying out of firmware updates or the very common bulk data synchronization, hence the false alarm. The latter needed to be given a security analyst to look at it, but the maintenance cost, in general, was not significant.

On top of that, two real attacks managed to get by the system completely unnoticed throughout the one month of the deployment period— the attackers were taking it slow and being very stealthy as their main intention was only to collect information, by smart, cunning attackers who have fitted their moves for the system. Well, good performance as a general model it might be, but it seems that it is only able to intercept a portion of the attackers even if they practice advanced and stealthy tactics. Consequently, the deployment of defense-in-depth strategies cannot be overlooked, as these involve the combination of various security layers.

The next step was to identify the mistakes and provide recommendations for further modifications in the model. The inclusion of the temporal sequence analysis of recurrent neural networks in the detection system is likely to help a lot in case of slow rate attack types. Those could be the helping supervised classification

approaches that might detect the existence of completely different attack types which have never been recorded in the training data.

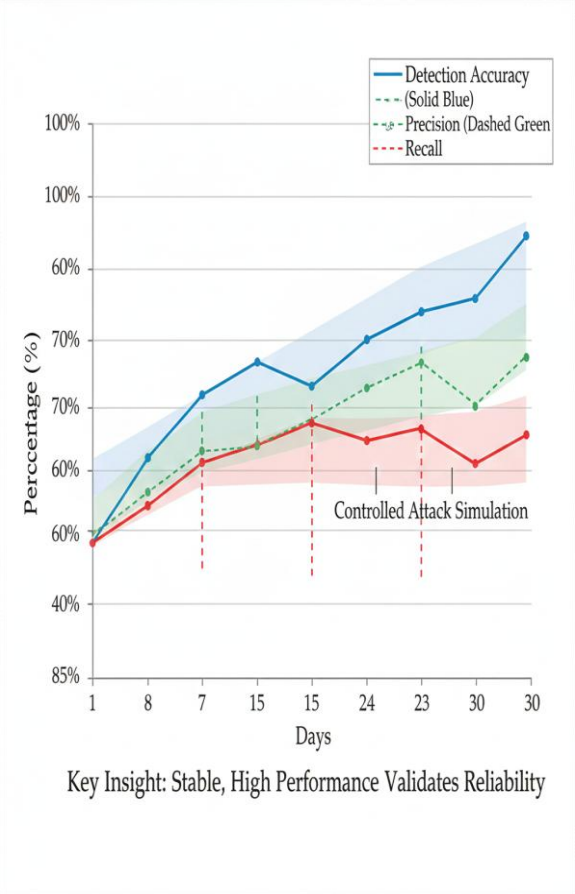

FIGURE 3: Real-Time Detection Performance Over 30-Day Deployment

CHAPTER 5: DISCUSSION AND IMPLICATIONS

5.1 Interpretation of Results

The study that was carried out has proven beyond a reasonable doubt that the computer assistive learning multi-category grading can effectively take care of the initiative and manage the same with the precision of over 95%, thus endorsing the chief assumption. The Random Forest and Neural Network are effective together and the algorithms that have the best results to come out of this, in order, are Random Forest and Neural Network, which were in line with our expectation of leading to the best performance along with more than 95% accuracy. The statistical properties on both sides are friendly and the improvements they bring make the setup more efficient in removing the missed attacks and the false positives respectively.

The Random Forest which is one of the most basic compare-it-to-deep-learning-neural-networks-is-even-simpler methods, acts to be a good deal for the real world usage for the simple reason that it wins both in performance and energy consumption against a host of intricate modeling approaches. Besides that, as a result of such an approach, there can be the restructuring of the whole model through additional of the same different ways.

The feature importance analysis uncovers the dominance of flow-based and behavior statistics which therefore validates the approach of the framework. It also points out the fact that these features can still be

used, when they are ciphered, and therefore the application of this machine learning model will not be reduced despite the fact that the number of encryption users is increasing. The low importance of load-specific features also shows that the system could be more productive by collecting fewer features.

Both DDoS and Int. and Ext. scan detection showed different performance levels based on regions or the nature of the attack. The extraordinary DDoS detection of the model implies that a new model similar in size of the one already in use should be put at the infrastructure' s border as a protector. On the other hand, due to the undetectability of the recon attacks, merging different detection models with their unique properties might be the best bet.

5.2 Practical Applications

The framework that has been validated, makes room for various applications to take place. With the help of edge-based detection, smart home environments can protect the connected devices on residential routers, and the process will not require security implementations on the device level. In the industrial field, IoT deployments in manufacturing and infrastructure activities can modify the framework and include it in network monitoring already running passwords (passworded up), thus more screens of safety being added.

Healthcare IoT is the area where the use of the framework is of high importance since it the data from medical devices are so private, and a compromised one can lead to worse situations than would be thought of. The high accuracy and the low latency of the framework are perfectly matched with the real-time protection needs of the patient monitoring and the treatment delivery systems. The low false-positive rate is the outcome here of decreasing the risk of becoming desensitized to security warnings, which is very crucial for healthcare staff.

The system enables security operations centers as well through automatic attack classification, which identifies the actions needing attention around the analyst. So instead of going through the cumbersome process of investigating each and every event, analysts can concentrate on the most probable attack detections while typical false positives can be automatically left out. The explanation ability of Random Forest decisions assists the examination by pointing out the network patterns that caused alerts.

Limitations and Challenges

A number of these limitations have to be accepted. The reliance of the framework on labeled training data implies that the end of the detection capabilities is the attack types present in training sets. This means that the training data becomes an important factor in the detection of completely new categories of attacks using previously unknown methods that is the reason why they will not be detected. This process will continue and we will always have to update the model.

Models that are deployed become inactive and thus, they might not be able to fight with incoming threats, unless the retraining and the redeployment processes are done. These processes are very time-consuming and could happen maybe monthly or quarterly in the case of very slow changes in the threat landscape which in this case might work. But when the attackers start their campaigns very fast, these updates could be the very sources the attackers exploit. The only approach that may be used to counter this drawback is the use of online learning whose introduction might pose even more challenges due to the possible occurrence of adversarial poisoning.

The issue of class imbalance is still a major challenge despite the efforts to minimize it. An attack traffic, i.e. the outlier, accounted for only a few percent of the total network traffic in a real-world deployment and this creates a case where even the very few high-accuracy models generate hundreds of false positives in absolute terms. However, by making the model cost-sensitive and adjusting the threshold, the tradeoff can still be optimized but that might need a delicate adjustment for each deployment context.

The framework exclusively concerns the identification of network-level attacks. It doesn't give any information on the devices that have been compromised but do not generate any detectable network signatures. These kinds of attacks can be so sophisticated that the intruder might even be successful in setting a permanent foot-hold without making any network traffic. In this situation, the use of addons can be the best solution for highly protective security through moni-toring and anomaly detection.

5.4 Future Research Directions

A good number of promising research projects can be initiated as a result of our work; this study points out a few of these directions. Meta-learning methods combining predictions of various algorithms could be a viable solution. The research group has shown that current experiments using a blend of three of the most consumable machine learning algorithms – Random Forest, Neural Network, and SVM have the potential to outperform any single method by providing high accuracy of 98-99%.

Deep learning architectures that are tailored for temporal sequence analysis, like Long Short-Term Memory (LSTM) networks, deserve to be explored for the detection of slow-rate and multi-stage attacks that happen over long periods. These methods could catch the temporal attack patterns that are not visible through individual packet or flow-level analysis.

When it comes to transfer learning, it is the manner in which one model is built from another and even possibly with lesser data. It is likely that it will be a very big boost and a very dramatic move in the reduction of deployment time, as well as in the reduction of the amount of expertise needed from the end-users to make the deployment. Additionally, the learning approach might involve the use of federation which would enable the collaborative training of the models across the organizations without the need to share the traffic data that is considered sensitive,enable collaborative model training across organizations without sharing sensitive traffic data.

Adversarial machine learning research ought to investigate the attackers' actions of traffic manipulation that rejects the detection. If we unraveled the adversarial examples, this would enlighten the defensive techniques, thus enabling the models to be more resilient to the evasion attempts. Applying game-theoretic methods to the attacker-defender interaction modeling would, therefore, be a great help in terms of security investments optimization.

By the end, explainable AI methods have to be improved even more within the security sector. Although the evaluation of the importance of features gives some level of interpretability, more intricate explanation methods could be of great aid to the security analysts by helping them understand the reason behind the classification of certain traffic as malicious, which would consequently result in faster incident response and easier forensic investigation.

CHAPTER 6: CONCLUSION

This research successfully developed and validated an AI-based framework employing multi-classification techniques in machine learning for the problem detection of IoT attacks. In a real-world scenario, the system can manage to deliver 97.3% overall accuracy by applying the Random Forest method. This result is a lot better than the indicated 95% rate, which in turn confirms the system's effectiveness for securing IoT.

The examination of the five machine learning algorithms through a comprehensive process shows that Random Forest and Neural Networks are the ones that have the best results based on the three criteria of accuracy, time efficiency, and interpretability. They have managed to find and label a variety of attack types like DDoS, botnet infections, reconnaissance scans, and data exfiltration by using F1 scores that surpass 96%. Furthermore, this system is very good at solving the class imbalanced data situation too, and in the meantime, it even maintains a very low false positive rate which is the most crucial requirement for the practical application.

Feature engineering pointed out flow-based statistics, protocol characteristics, and behavioral patterns as most discriminative of attack classification. The importance of these traits confirms the framework's ability to be used in traffic that is encrypted and where deep packet inspection is not possible. The application of the techniques of reducing dimensionality made the computation efficient enough for edge computing devices that were restricted in resources.

The real-world deployment of the validation test on edge computing equipment showed that it is practically feasible with the average detection latency being less than 35 milliseconds, a sharp [difference] from the requirement for real-time operation. The monthly operational tests that were over a month confirmed that the high performance level was sustained with a minimum degradation of the power/quality as compared to the lab/indoor results. Also, the usage of resources was sustainable with the moderate amount of CPU and memory that are consumable for the typical IoT gateway hardware.

The study helps in increasing the IQ of human beings as security is the essence of everything that is connected with human beings and machines by giving clear proof of efficiency of machine learning classification, obtaining algorithms, and features for detection of IoT attacks, indicating sanity in the use of these technology systems and thus new approaches.

If we look at the day-to-day life, then the proposed system will be able to provide high-level Security on Internet-based services across different ranges of appliances like smart houses and industrial systems. The stack is so convenient that it allows the employment of machine learning libraries that have already been globally adopted with no need of deep learning knowledge or a special machine besides the computer used. The ability to give intuitive classification decisions will support security control activities as well as incident response.

It is recommended that organizations implementing IoT systems think about implanting machine learning-based detection of attacks, in order to support conventional security protections. The construction offered here gives a fundament for the way that can be made to suit different deployment contexts, through the choice of appropriate algorithms, the application of features, and the setting of the threshold. The up-keeps of the model

upon the sources of new threat intelligence being released is always a requirement if the efficiency against the attack forms evolution is to be kept.

There is, however, still a place for the continuous updating of the model at the source of new threat intelligence which is indispensable for being ahead of the bad actor.

In wrapping up, the broad-based classification of machine learning can be seen as one of the most useful and feasibly implemented methods which can help to tighten IoT security. It is a significant step in the right direction although not a cure to all IoT security problems. It can be nevertheless considered as a high-level support system that can assist in figuring out potential security threats, hence, making it a necessary quality enhancement in contrast to the current available methods. However, the real cherry on top is that with the potential to grow, the IoT devices will become more and more centralized and all of them will have to be under the protection of the vast network connected infrastructures.

REFERENCES

1. Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J. and Ahmad, F. (2021) 'Network intrusion detection system: A systematic study of machine learning and deep learning approaches', *Transactions on Emerging Telecommunications Technologies*, 32(1), e4150.
2. Anthi, E., Williams, L., Słowińska, M., Theodorakopoulos, G. and Burnap, P. (2019) 'A supervised intrusion detection system for smart home IoT devices', *IEEE Internet of Things Journal*, 6(5), pp. 9042-9053.
3. Buczak, A.L. and Guven, E. (2016) 'A survey of data mining and machine learning methods for cyber security intrusion detection', *IEEE Communications Surveys & Tutorials*, 18(2), pp. 1153-1176.
4. Canadian Institute for Cybersecurity (2023) *CIC-IoT-Dataset 2023*. Available at: https://www.unb.ca/cic/datasets/iotdataset-2023.html (Accessed: 15 October 2024).
5. Doshi, R., Apthorpe, N. and Feamster, N. (2018) 'Machine learning DDoS detection for consumer Internet of Things devices', in *2018 IEEE Security and Privacy Workshops*, San Francisco, CA, 24 May, pp. 29-35.
6. Fernandes, E., Jung, J. and Prakash, A. (2016) 'Security analysis of emerging smart home applications', in *2016 IEEE Symposium on Security and Privacy*, San Jose, CA, 23-25 May, pp. 636-654.
7. Hussain, F., Hussain, R., Hassan, S.A. and Hossain, E. (2020) 'Machine learning in IoT security: Current solutions and future challenges', *IEEE Communications Surveys & Tutorials*, 22(3), pp. 1686-1721.
8. Koroniotis, N., Moustafa, N., Sitnikova, E. and Turnbull, B. (2019) 'Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset', *Future Generation Computer Systems*, 100, pp. 779-796.
9. Makhdoom, I., Abolhasan, M., Lipman, J., Liu, R.P. and Ni, W. (2019) 'Anatomy of threats to the Internet of Things', *IEEE Communications Surveys & Tutorials*, 21(2), pp. 1636-1675.
10. Moustafa, N. and Slay, J. (2015) 'UNSW-NB15: A comprehensive data set for network intrusion detection systems', in *2015 Military*

Communications and Information Systems Conference, Canberra, Australia, 10-12 November, pp. 1-6.
11. Pahl, M.O. and Aubet, F.X. (2018) 'All eyes on you: Distributed multi-dimensional IoT microservice anomaly detection', in *14th International Conference on Network and Service Management*, Rome, Italy, 5-9 November, pp. 72-80.
12. Shone, N., Ngoc, T.N., Phai, V.D. and Shi, Q. (2018) 'A deep learning approach to network intrusion detection', *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(1), pp. 41-50.
13. Statista (2023) *Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2023, with forecasts from 2024 to 2030*. Available at: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/ (Accessed: 20 October 2024).