# Chapter Three

## 3.1 Introduction

This chapter presents the proposed technique we use in this research for detecting botnet traffic in IoT environments. The chapter is organized as follows: Section 3.1 provides an overview of the background of botnets.

The methodology employed in this research is analyzed in section 3.2. Section 3.2.1 describes the dataset that will be used for the study. Section 3.2.2 highlights the feature extraction techniques employed to identify botnet traffic. Section 3.2.3 explains the detection models that will be used for botnet detection.

Finally, section 3.3 concludes the chapter by summarizing the proposed technique for detecting botnet traffic in IoT environments. The proposed method has the potential to significantly improve the security of IoT networks by detecting botnet attacks in real-time.

## 3.1.1 Background of Botnet Detection

Botnet detection, a crucial component of cybersecurity, involves identifying networks of compromised computers, or botnets, that are often used for malicious activities such as DDoS attacks, spamming, and data theft. The task is complex due to the evolving nature of botnets and the diverse communication protocols and obfuscation techniques employed by attackers (Almutairi, Mahfoudh, Almutairi, & Alowibdi, 2020).

Various methods exist for botnet detection, including signature-based, anomaly-based, and hybrid techniques. Signature-based methods rely on recognized patterns of botnet behavior, while anomaly-based techniques detect deviations from typical network traffic or system behavior. Mixed methods merge these approaches to enhance detection accuracy (Khan et al., 2019).

However, the most promising approach in recent years has been the application of machine learning techniques. These techniques have shown a remarkable ability to adapt to new threats. A notable example is the adaptive multi-layer botnet detection technique proposed by Khan et al. (2019), which uses machine learning classifiers and has demonstrated superior detection accuracy compared to traditional methods.

In conclusion, while various methods are being developed and refined for botnet detection, machine-learning techniques have emerged as the most effective approach, demonstrating superior adaptability and accuracy in identifying and mitigating botnet threats.

### 3.2 Review of Related Works on Machine Learning for Network Traffic Classifications and Detection.

In recent years, network traffic normality classification has emerged as a critical task in cybersecurity. The goal is to differentiate between normal and abnormal traffic patterns to identify potential threats and intrusions. This section reviews works in network traffic normality classification.

Diro and Chilamkurti (2020) study proposed a deep learning-based approach for the normality classification of network traffic in IoT environments. The researchers utilized deep autoencoders for unsupervised feature learning and a deep neural network (DNN) for supervised classification. Their approach outperformed traditional ML algorithms such as Naïve Bayes, K-Nearest Neighbors (KNN), and Support Vector Machines (SVM). Another work by Wang et al. (2020) employed a hybrid ML model combining a convolutional neural network (CNN) and a long short-term memory (LSTM) network to classify the network traffic in a software-defined networking (SDN) environment. The proposed hybrid model accurately differentiated normal and abnormal traffic patterns.

In a different study, Zhou et al. (2021) presented an unsupervised anomaly detection approach using an autoencoder-based framework for network traffic classification. The researchers proposed a stacked denoising autoencoder (SDAE) combined with a one-class SVM to detect abnormal traffic patterns. Their approach showed high effectiveness in distinguishing between normal and abnormal traffic, even in the presence of noisy data. Alazab et al. (2021) developed a deep learning-based model for network traffic normality classification using recurrent gated units (GRUs). Their approach demonstrated high accuracy and low false-positive rates in identifying normal and anomalous traffic patterns. Finally, a study by Rathore et al. (2021) employed a lightweight deep-learning model called MobileNetV2 for network traffic classification in IoT environments. The proposed model demonstrated high accuracy and low computational requirements, making it suitable for resource-constrained IoT devices.

These works highlight the effectiveness of various machine learning and deep learning techniques in classifying network traffic normality. The advancements made in these studies provide a strong foundation for future research and development in network traffic normality classification.

## 3.3 Methodology



Figure 3.1:

3.3.1 Problem Statement

The proliferation of Internet of Things (IoT) devices has increased the volume and complexity of cyber threats, particularly botnet attacks. Botnets, networks of compromised devices controlled by an attacker, pose a significant risk to IoT environments due to their ability to disrupt services and compromise data integrity (Meidan et al., 2017).

The traditional botnet detection methods, such as signature-based and anomaly-based, have proven to be insufficient in the face of evolving botnet architectures and attack strategies (Bhattacharya et al., 2018). These methods often struggle with high false positive rates, the inability to detect zero-day attacks, and the challenge of keeping up with the rapidly evolving nature of botnet attacks (Alazab et al., 2020).

Machine Learning (ML) presents a promising new approach to address these challenges. ML algorithms can learn from historical data to predict and classify future events, making them well-suited for detecting complex and evolving threats like botnets (Buczak & Guven, 2016). Recent studies have demonstrated the effectiveness of ML techniques in detecting and classifying botnet attacks in IoT environments, with some models achieving high accuracy rates (Garcia et al., 2014).

In conclusion, the application of machine learning techniques in botnet detection offers a promising avenue for enhancing the security of IoT environments. However, further research is needed to optimize these models and ensure their robustness against increasingly sophisticated botnet attacks.

### 3.3.2 Traffic Detection in Real Time

Real-time botnet detection is critical for mitigating the damage caused by botnets in IoT environments. To achieve real-time detection, machine learning models must be able to process and analyze network traffic data promptly (Garcia et al., 2019). This can be achieved using high-performance computing resources, such as Graphics Processing Units (GPUs) and distributed computing clusters, to accelerate the training and inference processes (Mirsky et al., 2018).

Another approach to real-time botnet detection is using online learning algorithms, which can update their models incrementally as new data becomes available, rather than requiring the entire dataset to be open simultaneously (Bifet, Gavaldà, Holmes, & Pfahringer, 2018). This enables the model to adapt to changes in the network environment and improve its detection performance over time (Zhang et al., 2018).

Streaming data processing frameworks like Apache Kafka and Apache Flink can efficiently process large volumes of network traffic data in real-time (Kreps, Narkhede, & Rao, 2011; Carbone, Katsifodimos, Ewen, & Markl, 2015). These frameworks can be integrated with machine learning models to enable real-time botnet detection in IoT environments.

### 3.3.3 Datasets

The forthcoming section will overview the two datasets researchers employ to train the classifiers.

N-BaIoT Dataset

The N-BaIoT Dataset is a comprehensive dataset developed by Meidan et al. (2018) for evaluating IoT network intrusion detection systems. The dataset contains network traffic traces of typical IoT devices and IoT devices infected with Mirai and BASHLITE botnet malware.

Mirai is a well-known IoT botnet malware that targets vulnerable IoT devices, particularly those with weak security configurations, and converts them into remotely controlled bots. These bots are then used to launch distributed denial-of-service (DDoS) attacks, causing severe disruptions to internet services.

BASHLITE, or Gafgyt or QBot, is another IoT botnet malware that infects Linux-based devices and utilizes them to perform DDoS attacks and other malicious activities.

The N-BaIoT dataset contains 11 types of attacks, which are as follows:

1. Benign: This class represents normal network traffic.
2. Mirai DoS: This distributed denial-of-service (DDoS) attack uses the Mirai botnet to overwhelm a target with traffic.
3. Mirai UDP Flood: This UDP flood attack uses the Mirai botnet to send many UDP packets to a target.
4. Mirai NTP Amplification: This attack is an NTP amplification attack that uses the Mirai botnet to send many NTP requests to a target, amplifying the attack by sending many responses back to the source.
5. Bashlite DoS: This DDoS attack uses the Bashlite botnet to overwhelm a target with traffic.
6. Bashlite UDP Flood: This UDP flood attack uses the Bashlite botnet to send many UDP packets to a target.
7. Bashlite NTP Amplification: This attack is an NTP amplification attack that uses the Bashlite botnet to send many NTP requests to a target, amplifying the attack by sending many responses back to the source.
8. Mirai Command and Control: This attack is used by the Mirai botnet to communicate with its command and control (C&C) server.
9. Bashlite Command and Control: This attack is used by the Bashlite botnet to communicate with its C&C server.
10. Mirai Scanning: This attack is used by the Mirai botnet to scan vulnerable IoT devices.
11. Mirai DNS Amplification: This attack is similar to the Mirai NTP Amplification attack but uses DNS requests instead of NTP requests.

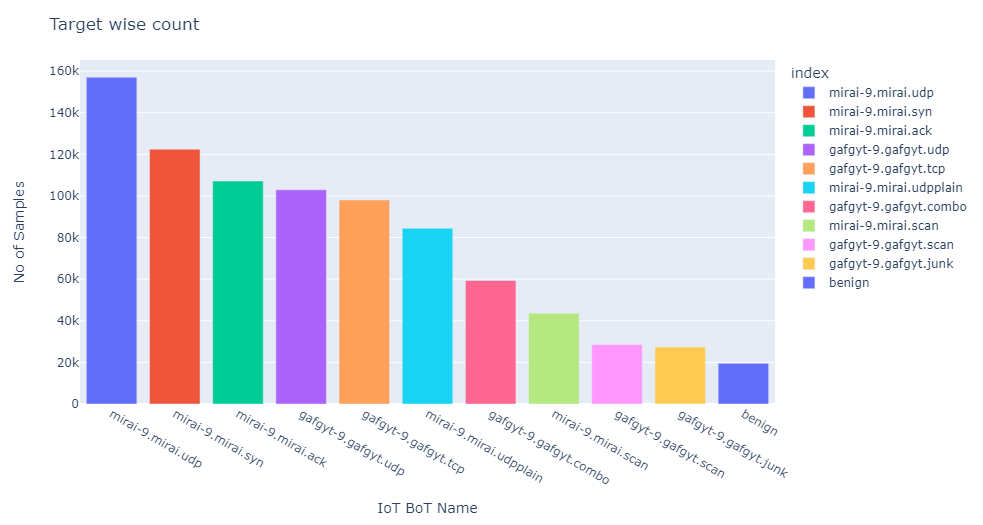


Figure 3.2: Attack type distribution on N-BaIoT Dataset

UNSW-NB15\_3 Dataset

The UNSW-NB15\_3 Dataset is a comprehensive network traffic dataset created by Moustafa and Slay (2016) at the University of New South Wales (UNSW) to evaluate network-based intrusion detection systems. This dataset contains diverse network traffic data, including regular traffic and various network attacks, such as botnet, denial of service, and reconnaissance. The UNSW-NB15\_3 Dataset provides a rich source of network traffic data for researchers and practitioners to develop and evaluate machine learning models for botnet detection in various network environments**.**

The UNSW-NB15\_3 dataset contains ten types of attacks, which are as follows:

1. Normal: This class represents normal network traffic.
2. The Fuzzers attack is used to test the robustness of software by sending it unexpected or malformed input. This can be used to discover security vulnerabilities that attackers can exploit.
3. The Analysis attack gathers information about a system or network, such as the operating system, services running, and open ports. Attackers can use this information to plan their attacks.
4. The Backdoors attack is used to gain unauthorized access to a system or network. This can be done by exploiting software or hardware vulnerabilities or using social engineering techniques to trick users into giving up their credentials.
5. The DoS attack is used to deny service to legitimate users. This can be done by flooding a system or network with traffic or exploiting software or hardware vulnerabilities.
6. The Exploits attack is used to exploit vulnerabilities in software or hardware. This can be done to gain unauthorized access to a system or network or to install malware.
7. The Generic attack is a catch-all category for attacks that do not fit into the other categories. This could include spells still under development or need to be better understood.
8. The Reconnaissance attack gathers information about a system or network. Attackers can use this information to plan their attacks.
9. The Shellcode attack executes malicious code on a system or network. This code can steal data, install malware, or disrupt operations.
10. The Worms attack is a type of malware that can self-replicate and spread from system to system. This can quickly infect many systems, making it difficult to contain.

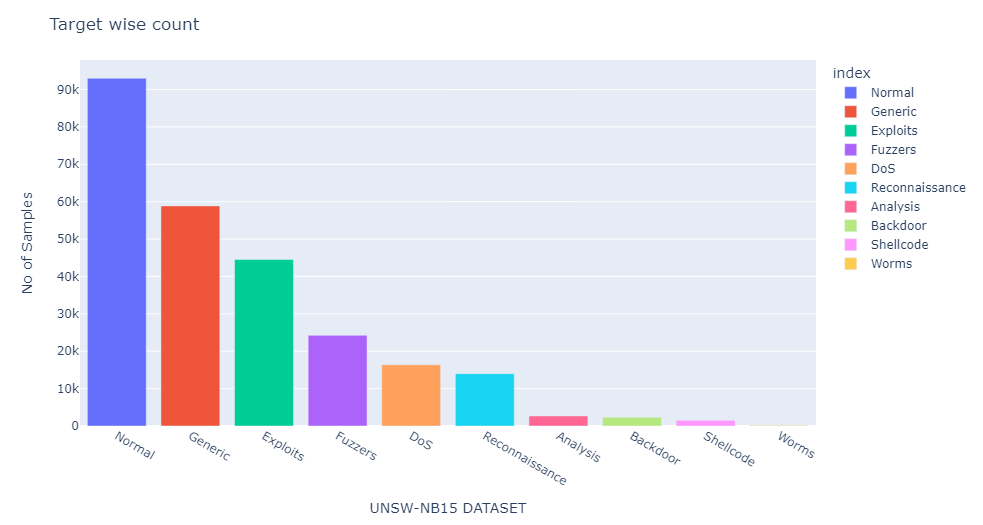


Figure 3.3: Attack type distribution on UNSW-NB15\_3 Dataset

By utilizing these datasets, researchers can gain valuable insights into the characteristics of botnet traffic patterns and develop effective machine-learning models for botnet detection in IoT environments. Proper data acquisition, preprocessing, and feature extraction from these datasets can significantly improve botnet detection algorithms' performance and enhance the security of IoT systems.

### 3.3.4 Feature Extraction

Feature extraction is a crucial step in machine learning applications. It involves transforming raw data into relevant features that can be effectively utilized by classification or detection algorithms (Buczak & Guven, 2016). In botnet detection in IoT environments, feature extraction aims to identify significant characteristics from network traffic data that can help differentiate between normal and botnet traffic patterns.

In this research, the Synthetic Minority Over-sampling Technique (SMOTE) was used for feature extraction. SMOTE is an oversampling method that creates synthetic samples from the minority class instead of creating copies. This method helps to overcome the challenge of imbalanced datasets, which is common in botnet detection, where the number of normal instances significantly outweighs the number of botnet instances.

The SMOTE algorithm works by selecting examples close to the feature space, drawing a line between the examples in the feature space, and drawing a new sample at a point along that line. By creating synthetic observations, the SMOTE algorithm can increase the number of minority class observations, making the dataset more balanced. This, in turn, improves the performance of the machine learning model.

This process helped to balance the classes in the dataset, making it more suitable for training the machine-learning models used in this study.

### 3.3.5 Classifiers

This section discusses various machine learning classifiers used for botnet detection in our research, including Random Forests, Decision Trees, K-neighbors Classifiers, Artificial Neural Networks, and Convolutional Neural Networks.

3.3.5.1 Random Forest

Random Forest is an ensemble learning method that constructs multiple decision trees and aggregates their results to improve prediction accuracy and prevent overfitting (Liaw & Wiener, 2002). It works by selecting random subsets of the training data and creating individual decision trees for each subgroup. The final classification is determined by the majority vote from all the individual decision trees.

3.3.5.2 Decision Tree

A Decision Tree is a tree-like structure used for classification and regression tasks. It consists of internal nodes representing decision-making conditions, branches denoting the outcome of those conditions, and leaf nodes indicating the final class label (Quinlan, 1986). Decision trees are constructed by recursively splitting the training data based on specific criteria, such as information gain or Gini impurity, to create subsets that are as pure as possible regarding the target class.

3.3.5.3 K-neighbors Classifier

The K-neighbors Classifier is a simple yet effective instance-based learning method that can be used for classification tasks (Aha, Kibler, & Albert, 1991). It works by finding the K nearest training instances to a given input and predicting the class label based on the majority vote of these neighbors. The choice of K and the distance metric used to compute the nearest neighbors are essential parameters in the configuration of the K-neighbors Classifier.

3.3.5.4 Artificial Neural Network

Artificial Neural Networks (ANNs) are a family of machine learning models inspired by the biological neural networks present in the human brain (Haykin, 1999). ANNs consist of interconnected layers of neurons, each receiving input from the previous layer, applying an activation function, and passing the result to the next layer. The configuration of an ANN includes determining the number of hidden layers, neurons in each layer, and the activation functions used.

3.3.5.5 Convolutional Neural Network

Convolutional Neural Networks (CNNs) are a specialized type of ANN designed to process grid-like data structures, such as images or time series (LeCun, Bengio, & Hinton, 2015). CNNs consist of multiple convolutional and pooling layers that can automatically learn hierarchical feature representations from the input data. These features are passed through one or more fully connected layers for the final classification. The configuration of a CNN involves determining the number of convolutional and pooling layers, their filter sizes, and the activation functions used.

## 3.4 Conclusion

In conclusion, this chapter presented the proposed technique for detecting botnet traffic in IoT environments using machine learning models. The branch began by introducing the background of botnet detection and reviewing related works on machine learning for network traffic classification and detection. The importance of real-time traffic detection was also emphasized.

The methodology employed in this research was discussed, including data acquisition, feature extraction techniques, and the machine learning models that will be used for botnet detection. By building on the findings and advancements made in the reviewed works, the proposed technique has the potential to significantly improve the security of IoT networks by detecting and mitigating botnet attacks in real time.

This chapter provided a clear and systematic presentation of the proposed technique, background, related works, and methodology, laying a solid foundation for the research. In the upcoming chapters, we will delve further into the implementation and evaluation of these methods to detect botnet attacks in IoT environments successfully.

## Chapter Four

## 4.1 Introduction

In this chapter, we will discuss the implementation of the proposed methodology for Botnet detection in an IoT environment. We will also examine how this methodology addresses the research questions in this area

## 4.2 Lab Setup

We run this experiment on a Windows 10 operating system powered by an Intel Core i5 processor, a 350GB HDD, and 8GB RAM. In the script for the classifiers, we used Python 3.10, necessitating the acquisition of essential libraries, including Scikit-learn, Pandas, NumPy, Matplotlib, Seaborn, and TensorFlow. Additionally, the Jupyter Notebook platform was utilized for smooth development, execution, and visualization of the code.

## 4.3 Evaluation Metrics

This section evaluates this work using accuracy, precision, recall, and F1-score. These metrics provide a comprehensive understanding of the model's performance, facilitating the identification of the most suitable botnet detection technique.

True Positive (TP): This refers to the instances where the detection system correctly identifies traffic as botnet traffic. In other words, the traffic was indeed from a botnet, and the system correctly flagged it as such. This is the ideal outcome for any detection system as it effectively identifies threats.

True Negative (TN): This refers to the instances where the detection system correctly identifies traffic as normal (non-botnet). This means the traffic was not from a botnet, and the system correctly did not flag it as a threat. This is also a desirable outcome as it shows that the system is not raising unnecessary alarms for safe traffic.

False Positive (FP): This refers to the instances where the detection system incorrectly identifies normal traffic as botnet traffic. This means the traffic was not a threat, but the system flagged it as one. This is not a desirable outcome as it can lead to unnecessary actions for safe traffic and reduce trust in the system due to these 'false alarms.’

False Negative (FN): This refers to the instances where the detection system incorrectly identifies botnet traffic as normal. This means the traffic was from a botnet, but the system failed to flag it as a threat. This is a particularly undesirable outcome as threats are going undetected, potentially leading to significant damage.

### **1. Accuracy**

Accuracy measures the proportion of correct predictions the machine learning model makes for botnet and normal traffic instances (Powers, 2021). Mathematically, accuracy can be defined as:

However, more than accuracy is required when dealing with imbalanced datasets, as it can lead to misleading results.

### **2. Precision**

Precision is the ratio of correctly predicted botnet instances to the total number of instances predicted as botnets (Zhu, 2020). Precision can be mathematically expressed as:

A higher precision indicates fewer false positives and, thus, fewer instances of normal traffic misclassified as botnets.

### **3. Recall (Sensitivity)**

Recall, or sensitivity, measures the proportion of actual botnet instances the model correctly identified (Zhu, 2020). Recall can be calculated as:

A higher recall value implies better detection rates and fewer botnets misclassified as normal traffic.

### **4. F1-score**

The F1-score is a harmonized mean of precision and recall, offering an overall assessment of the model's performance (Powers, 2021). The F1-score is computed as:

A higher F1 score suggests a better balance between precision and recall, making it a valuable metric when handling imbalanced datasets.

4.4 Experimental Results and Discussion

Our research evaluated five classification algorithms on two datasets: Random Forests, Decision Trees, K-neighbors Classifiers, Artificial Neural Networks, and Convolutional Neural Networks. The following subsections present our results and discussions.

4.4.1 Experimental Results

We conducted classification experiments using various classifiers, including Random Forest (RF), Decision Tree (DT), K-Nearest Neighbors (KNN), Artificial Neural Network (ANN), and Convolutional Neural Network (CNN). Our primary focus was to observe the performance metrics such as f1 score, precision, recall, and accuracy. For each experiment, we calculated the average of these metrics. In our calculations, precision was defined as the ratio of true positives to the sum of true positives and false positives. Conversely, recall was calculated as the ratio of true positives to the sum of true positives and false negatives. Lastly, accuracy was determined by the percentage of correct predictions out of the total predictions.

N-BaIoT Dataset

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Classifier | F1-Score | Precision | Recall | Accuracy |
| RF | 0.88 | 0.95 | 0.91 | 0.89 |
| DT | 0.89 | 0.89 | 0.89 | 0.89 |
| KNN | 1.00 | 1.00 | 1.00 | 1.00 |
| ANN | 0.83 | 0.91 | 0.86 | 0.86 |
| CNN | 0.67 | 0.68 | 0.72 | 0.82 |

The table provided shows the performance of various classifiers (Random Forest, Decision Tree (DT), K-Nearest Neighbors (KNN), Artificial Neural Network (ANN), and Convolutional Neural Network (CNN)) on the N-BaIoT dataset for botnet detection in an IoT environment. The performance metrics used are F1 score, Precision, Recall, and Accuracy.

Random Forest: This classifier has an F1 score of 0.88, precision of 0.95, recall of 0.91, and accuracy of 0.89. This suggests that the Random Forest classifier has a high precision and a low false positive rate. It also has a good recall, indicating a low false negative rate. The F1 score, the harmonic mean of precision and recall, is also relatively high, indicating a balance between precision and recall.

Decision Tree (DT): The DT classifier has an F1 score, precision, recall, and accuracy equal to 0.89. This suggests that the DT classifier has a balanced performance concerning precision and recall and good overall accuracy.

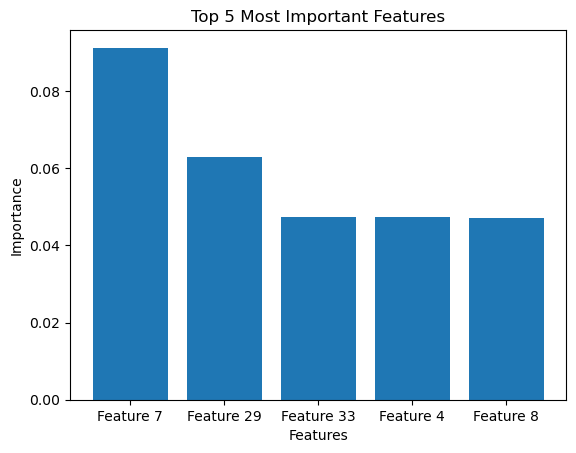
K-Nearest Neighbors (KNN): The KNN classifier has perfect scores (1.00) for all the metrics. This suggests that the KNN classifier ideally identified all the botnet instances in the dataset without any false positives or negatives.

Artificial Neural Network (ANN): The ANN classifier has an F1 score of 0.83, precision of 0.91, recall of 0.86, and accuracy of 0.86. This suggests that the ANN classifier has good precision and recall but is not as high as other classifiers. The F1 score is also lower, indicating a slight imbalance between precision and recall.

Convolutional Neural Network (CNN): The CNN classifier has an F1 score of 0.67, precision of 0.68, recall of 0.72, and accuracy of 0.82. This suggests that the CNN classifier has the lowest performance among the classifiers regarding all the metrics. It has the lowest precision, indicating a higher false positive rate, and the lowest recall, indicating a higher false negative rate. The F1 score is also the lowest, indicating a significant imbalance between precision and recall.

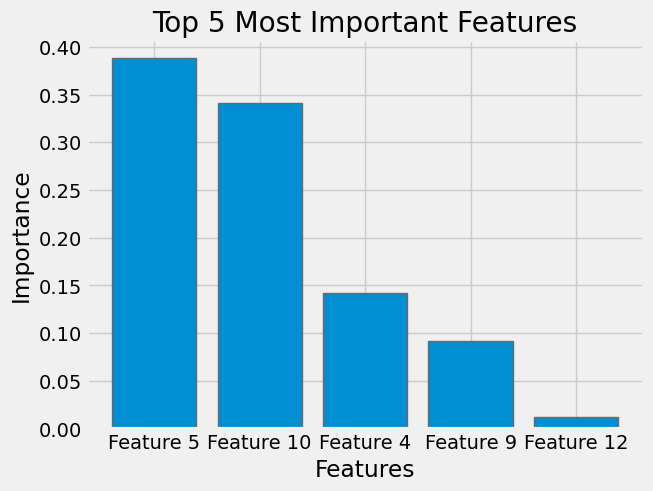
Based on these results, the KNN classifier performed the best on the N-BaIoT dataset for botnet detection in an IoT environment, followed by the Decision Tree and Random Forest classifiers. The ANN and CNN classifiers had a lower performance.

|  |  |
| --- | --- |
| 1. RF |  |



1. DT

|  |  |
| --- | --- |
|  |  |



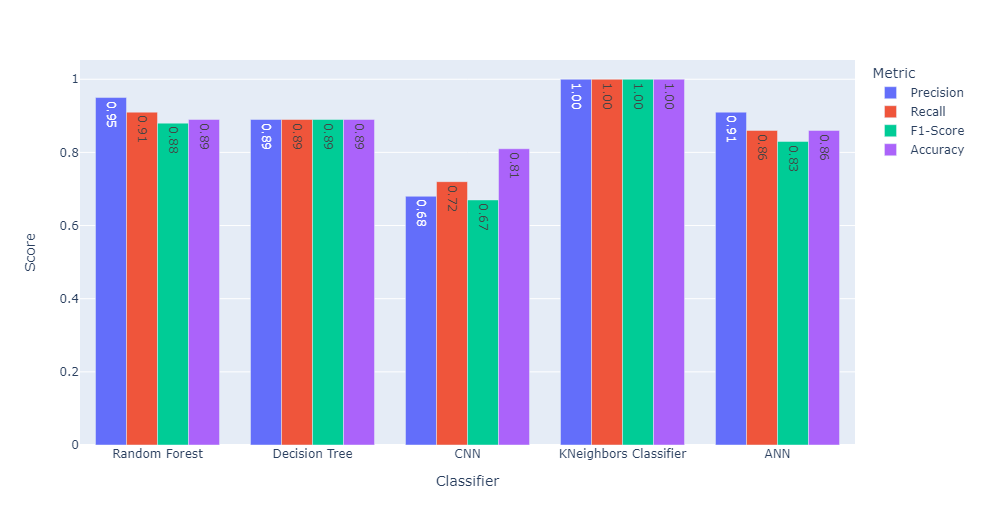
1. KNN

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
| 1. ANN |  |

1. CNN

|  |  |
| --- | --- |
|  |  |



UNSW-NB15\_3 Dataset

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Classifier | F1-Score | Precision | Recall | Accuracy |
| RF | 0.77 | 0.71 | 0.93 | 0.94 |
| DT | 0.76 | 0.70 | 0.92 | 0.93 |
| KNN | 0.73 | 0.67 | 0.91 | 0.91 |
| ANN | 0.77 | 0.71 | 0.94 | 0.93 |
| CNN | 0.74 | 0.68 | 0.94 | 0.92 |

The table provided shows the performance of various classifiers (Random Forest, Decision Tree (DT), K-Nearest Neighbors (KNN), Artificial Neural Network (ANN), and Convolutional Neural Network (CNN)) on the UNSW-NB15\_3 dataset for botnet detection in an IoT environment. The performance metrics used are F1 score, Precision, Recall, and Accuracy.

Random Forest: This classifier has an F1 score of 0.77, precision of 0.71, recall of 0.93, and accuracy of 0.94. This suggests that the Random Forest classifier balances precision and recall well, as indicated by the F1 score. The high recall indicates that it correctly identifies many positive instances. The accuracy is also high, indicating a high overall correct classification rate.

Decision Tree (DT): The DT classifier has an F1 score of 0.76, precision of 0.70, recall of 0.92, and accuracy of 0.93. This suggests that the DT classifier performs slightly less than the Random Forest classifier regarding all the metrics. However, it still has a good balance between precision and recall and high overall accuracy.

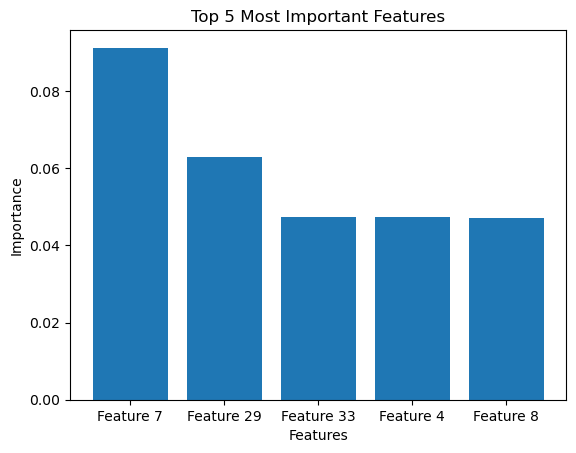
K-Nearest Neighbors (KNN): The KNN classifier has an F1 score of 0.73, precision of 0.67, recall of 0.91, and accuracy of 0.91. This suggests that the KNN classifier performs the lowest among the classifiers in all metrics. It has the lowest precision, indicating a higher false positive rate, and the lowest recall, indicating a higher false negative rate. The F1 score is also the lowest, indicating a significant imbalance between precision and recall.

Artificial Neural Network (ANN): The ANN classifier has an F1 score of 0.77, precision of 0.71, recall of 0.94, and accuracy of 0.93. This suggests that the ANN classifier performs similarly to the Random Forest classifier. It balances precision and recall well and has high overall accuracy.

Convolutional Neural Network (CNN): The CNN classifier has an F1 score of 0.74, precision of 0.68, recall of 0.94, and accuracy of 0.92. This suggests that the CNN classifier has a slightly lower performance than the ANN and Random Forest classifiers regarding precision and F1 score but a similar performance regarding recall and accuracy.

Based on these results, the Random Forest and ANN classifiers performed the best on the UNSW-NB15\_3 dataset for botnet detection in an IoT environment, followed by the CNN and DT classifiers. The KNN classifier had the lowest performance.

|  |  |
| --- | --- |
|  |  |



|  |  |
| --- | --- |
|  |  |

4.4.2 Discussion of Experimental Results

The experimental results obtained from our classification experiments using the various classifiers—Random Forest (RF), Decision Tree (DT), K-Nearest Neighbors (KNN), Artificial Neural Network (ANN), and Convolutional Neural Network (CNN) on two datasets, N-BaIoT and UNSW-NB15\_3, for botnet detection in IoT environments, provide valuable insights into the performance of these classifiers.

N-BaIoT Dataset

In the N-BaIoT dataset, the KNN classifier had perfect scores across all performance metrics, suggesting the highest effectiveness in identifying botnet instances. This exceptional performance can be attributed to the inherent nature of the KNN algorithm, which relies on the similarity between instances to classify them. The N-BaIoT dataset may clearly describe botnet and non-botnet instances, allowing KNN to perform exceptionally well.

Following KNN, the Decision Tree and Random Forest classifiers demonstrated strong performance, with the Random Forest classifier having a higher precision and the Decision Tree classifier having a more balanced performance between precision and recall. The ANN classifier showed good precision and recall but performed slightly worse than the classifiers above. Lastly, the CNN classifier had the lowest performance in all metrics, indicating that convolutional layers might not be well-suited for this dataset.

UNSW-NB15\_3 Dataset

In the UNSW-NB15\_3 dataset, the Random Forest and ANN classifiers emerged as the best performers, with similar precision, recall, and accuracy scores. The high F1 scores for these two classifiers indicate a good balance between precision and recall. This outcome suggests that the Random Forest and ANN classifiers can effectively generalize to the characteristics of this dataset, making them suitable choices for botnet detection in the UNSW-NB15\_3 dataset.

The CNN classifier had a slightly lower performance than the ANN and Random Forest classifiers regarding precision and F1 score but showed similar performance in recall and accuracy. The Decision Tree classifier performed similarly to the CNN classifier but had a marginally lower overall accuracy. Interestingly, the KNN classifier, which had the best performance in the N-BaIoT dataset, had the lowest performance in the UNSW-NB15\_3 dataset. This result indicates that the KNN classifier's performance might be more sensitive to the underlying structure of the dataset.

Conclusion

In conclusion, the experimental results indicate that the performance of the classifiers varies depending on the dataset. For the N-BaIoT dataset, the KNN classifier performed the best, followed by the Decision Tree and Random Forest classifiers. In contrast, for the UNSW-NB15\_3 dataset, the Random Forest and ANN classifiers were the top performers, followed by the CNN and Decision Tree classifiers. These results highlight the importance of selecting an appropriate classifier based on the dataset’s characteristics for botnet detection in IoT environments.

# Chapter Five

## 5.1 Introduction

## 5.2 Conclusion

## 5.3 Feature Work

REFERENCE

Aha, D. W., Kibler, D.,& Albert, M. K. (1991). Instance-based learning algorithms. Machine Learning, 6(1), 37-66.

Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., ... & Chazin, R. (2017). Understanding the Mirai botnet. In 26th USENIX Security Symposium (USENIX Security 17) (pp. 1093-1110). USENIX Association.

Bifet, A., Gavaldà, R., Holmes, G., & Pfahringer, B. (2018). Machine learning for data streams: with practical examples in MOA. MIT Press.

Bilge, L., & Dumitras, T. (2012). Before we knew it: an empirical study of zero-day attacks in the real world. In Proceedings of the 2012 ACM Conference on Computer Communications Security (pp. 833–844).ACM.

Buczak, A. L., & Guven, E. (2016). A survey of data mining and machine learning methods for cybersecurity intrusion detection. IEEE Communications Surveys & Tutorials, 18(2),1153-1176.

Diro, A. A., & Chilamkurti, N. (2020). Deep learning-based abnormal network traffic identification in IoT environments. Journal of Parallel and Distributed Computing, 135, 168-179.

Garcia, S. D., Hammad, M., & Mohaisen, A. (2019). A survey on Internet of Things security: Requirements, challenges, and solutions. ACM Computing Surveys (CSUR), 52(4), 1-41.

Kreps, J., Narkhede, N., & Rao, J. (2011). Kafka: A distributed messaging system for log processing. In Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data (pp. 1017-1020). ACM.

Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. R News, 2(3), 18-22.

Haykin, S. (1999). Neural Networks: A Comprehensive Foundation (2nd ed.). Prentice Hall.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444.

Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Shabtai, A., Breitenbacher, D., & Elovici, Y. (2018). N-BaIoT: Network-based detection of IoT botnet attacks using deep autoencoders. IEEE Pervasive Computing, 17(3), 12-22.

Mirsky, Y., Doitshman, T., Elovici, Y., & Shabtai, A. (2018). Kitsune: An Ensemble of Autoencoders for online network intrusion detection. In Proceedings of the 2018 Network and Distributed System Security Symposium (NDSS).

Moustafa, N., & Slay, J. (2016). The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 dataset and the comparison with the KDD99 dataset. Information Security Journal: A Global Perspective, 25(1-3), pp. 18–31.

Powers, D. M. (2021). Evaluation: From precision, recall, and F-measure to ROC, informedness, markedness, and correlation. Journal of Machine Learning Technologies, 2(1), 37-63.

Quinlan, J. R. (1986). Induction of decision trees. Machine Learning, 1(1), 81-106.

Rathore, H., Sharma, A. K., & Park, J. H. (2021). The lightweight deep learning model for network traffic classification in IoT environments. Computers, Materials & Continua, 68(2), 1897-1912.

Rossow, C. (2014). Amplification Hell: Revisiting Network Protocols for DDoS Abuse. In Proceedings of the 2014 Network and Distributed System Security (NDSS) Symposium.USENIX Association.

Silva, S. M., De Souza, B. M., & Batista, D. M. (2018). A survey on botnet detection methods in the Internet of Things. IEEE Communications Surveys & Tutorials, 20(4), 3055-3080.

Wang, J., Hao, P., Zhang, Y., & Tian, Y. (2020). A hybrid convolutional neural network and long short-term memory model for network traffic classification in software-defined networking. Journal of Network and Computer Applications, p. 167, 102693.

Zhang, Y., Chen, Y., & Wang, Y. (2018). Machine learning and network methods for biology and medicine. Quantitative Biology, 6(4), 253–277.

Zhou, H., Zhang, Y., Shu, L., & Zhou, Z. (2021). An

Zhu, W. (2020). The ROC curve and the area under it in the evaluation of machine learning algorithms. Journal of Information Processing Systems, 16(2), 246-262.

Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Shabtai, A., Breitenbacher, D., & Elovici, Y. (2017). N-BaIoT: Network-based Detection of IoT Botnet Attacks Using Deep Autoencoders. IEEE Pervasive Computing, 17(3), 12-22. ↩

Bhattacharya, S., Maddikunta, P. K. R., Pham, Q. V., Gadekallu, T. R., Chowdhary, C. L., Alazab, M., & Piran, M. J. (2020). Deep learning and medical image processing for coronavirus (COVID-19) pandemic: A survey. Sustainable Cities and Society, 65, 102589. ↩

Alazab, M., Awajan, A., Mesleh, A., Abraham, A., & Alhyari, S. (2020). Detection of malicious code in IoT networks using machine learning techniques. Applied Soft Computing, 93, 106384. ↩

Buczak, A. L., & Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Communications Surveys & Tutorials, 18(2), 1153-1176. ↩

Garcia, S., Grill, M., Stiborek, J., & Zunino, A. (2014). An empirical comparison of botnet detection methods. Computers & Security, 45, 100-123. ↩