A hybrid ML-based model for improved detection of IoT botnets

Kabilan Gnanavarothayan

Delft University of Technology

TU Delft

A hybrid ML-based model for improved detection of IoT botnets

by

Kabilan Gnanavarothayan

to obtain the degree of Master of Science in Computer Science
at the Delft University of Technology,
to be defended on Wednesday October 25, 2023 at 2:00 PM.

Student number:	4600223	
Thesis committee:	Prof. dr. M. Conti,	TU Delft, thesis supervisor, daily supervisor
	Dr. C. Lal,	University of Padova, committee member
	Dr. L. Y. Chen,	TU Delft, committee member

TU Delft

Abstract

The use of Internet of Things (IoT) devices has experienced an increase since its inception and is expected to continue to do so. However, this growth has also attracted individuals with malicious intentions. Botnet attacks on IoT devices have become more potent each year, exploiting new vulnerabilities and attacking more devices. Therefore, it is imperative to improve countermeasures. N-BaIoT is a frequently used dataset that covers botnet attacks in various stages of the botnet life cycle. Nevertheless, when examining the state-of-the-art utilizing the dataset, there are certain limitations that need to be addressed.

One limitation is the lack of detailed feature analysis in most studies. This results in less comprehension of the behavior of the malicious and benign data in the dataset, leading to a lack of feature optimization. Feature optimization is crucial as it improves the computational time of the model and makes it efficient to deploy in real-life applications. Another limitation is the uneven distribution of the malicious and benign data, resulting in unreliable evaluation scores. This issue has not been addressed in many studies.

The main contribution of this thesis is the development of a hybrid ensemble model to detect IoT botnet attacks faster and accurately. Additionally, the aim of this study is to provide a clear analysis of the behavior of the malicious and benign data and optimize the number of selected features. In the hybrid ensemble model, a minimal number of features will be utilized to evaluate performance. The comparison will be achieved by taking into account both the unbalanced and balanced datasets used for training and testing. By considering both datasets, the limitation of the distribution can be highlighted by examining the distinct performance of each dataset, making the comparison extensive and reliable.

The results indicate that the selected features and detection models proposed in this research outperform those of other studies. In both cases using the balanced and unbalanced sets, the performance score and computational time are improved. In addition, this is achieved by using fewer features compared to several studies.

Preface

Before you lies the thesis ”A hybrid ML-based model for improved detection of IoT botnets”, where a hybrid ensemble model is developed, designed, and evaluated for faster and improved detection of IoT botnet attacks. This thesis is my graduation project for the Master’s Degree in Computer Science at the EEMCS faculty of Delft University of Technology.

I did not have much personal knowledge about IoT botnets since it is a recent topic. It required extensive analysis to determine the general structure of the IoT botnets along with key components used in the detection schemes surrounding them. From the studies, the limitations were addressed surrounding the key components and this led to the final work in this thesis.

First, I would like to thank my two supervisors C.Lal, my daily supervisor, and M.Conti, my thesis supervisor, throughout this challenging process, which involved both personal aspects and the research itself. I am grateful they had the patience and that we managed to set up and finalize this thesis project. Furthermore, I would like to thank the thesis committee members for involving in the thesis defense. Finally, I would like to thank my family and friends for sticking by me through the challenges of the last few years.

Kabilan Gnanavarothayan
09-06-2023
Dordrecht, the Netherlands

Contents

1

Introduction

The use of Internet of Things(IoT) devices has experienced an increase in recent years and is expected to continue to do so in the future [45]. This trend has attracted the attention of malicious actors. Among the most notorious attacks on IoT devices are botnet attacks. Due to the relative simple firmware of these devices, it is not difficult to convert them into bots and use them to carry out attacks on services, such as DDoS attacks.

It is imperative to develop effective countermeasures against these attacks. Detection schemes have proven to be effective means of identifying potential botnet attacks and mitigating them before they cause significant harm. However, botnet attacks have become more potent over time, with new attack types infecting additional devices before they can be detected. Therefore, it is crucial to ensure that countermeasures continue to evolve and improve to combat these threats.

1.1. Background

One of the most notorious botnet attacks on IoT devices occurred on September 19, 2016, when the Mirai botnet launched two concurrent DDoS attacks against the French hosting firm OVH, generating a combined bandwidth of over 1 terabit per second [17]. The attack was orchestrated by a botnet comprising 145,607 devices, primarily digital recorders and IP cameras. This attack was twice as big as the largest attack ever recorded by Akamai and would have resulted in millions of dollars in losses for the company if it had continued.

The threat did not end there, as a new variant of the Mirai malware was discovered in December 2017, capable of exploiting ARC processors [8]. Given the widespread use of these processors in IoT devices, an estimated 1.5 billion devices worldwide could have been vulnerable. Furthermore, a similar variant known as Satori conducted an attack that caused hundreds of thousands of Huawei routers to go offline.

On August 19, 2021, Cloudflare was targeted by a botnet variant called Meris [25]. This time the botnet sent 17.2 million false requests per second to the company's servers, overwhelming their capacity to handle 25 million HTTP requests per second. As a result, the company's services were suspended for a period of time. The same botnet also attacked the Russian internet company Yandex, which received 21.8 million bogus requests per second from 250,000 IoT devices worldwide. To put this in perspective, the Mirai attack in 2016 generated a total of 450,000 requests per second.

Finally, a new variant of the Mirai malware, known as V3G4, was discovered in the second half of 2022 [29]. This variant exploits 13 additional vulnerabilities in both Linux and IoT devices, including data centers, servers, routers, and air spots among others.

1.2. Problem

It is evident that new variants of botnets will continue to emerge, affecting more devices through novel forms of attacks. It is therefore imperative to ensure that countermeasures advance in parallel. In the context of detection schemes, the selection of the dataset for analysis and assessment is the most impactful step. These datasets are generated by executing various botnet attack scenarios, rendering earlier datasets outdated as they do not account for recent botnet exploitations. Recent studies indicate that the N-BaIoT dataset [32] is frequently utilized due to its inclusion of data from numerous attack scenarios that earlier datasets fail to consider [34].

However, examining the literature employing the dataset reveals several limitations that need to be addressed. First, the dataset contains far more malicious data than benign data, as highlighted in [33], leading to inflated assessment scores, a concern that many studies fail to address. Furthermore, most research employs auto-encoders or other algorithms to identify relevant features, followed by the use of a random number of these features for model training. There is limited discussion on the relevance of the chosen features or justification for the number of features selected, which is essential for feature optimization [34][2][30][10]. Finally, the models employed for detection and evaluation are often basic ones from a library [30][6]. Modified versions, such as hybrid methods, often utilize deep learning methods. These models have a slow computational speed, which often gets compensated by using fewer data for training and testing [35][21]. This can be attributed to the lack of feature selection and the detection models chosen.

1.3. Contributions

This research aims to address the problems discussed and the proposed work is expected to resolve a significant number of them. The contributions of this research are outlined as follows:

- A hybrid ensemble model will be designed and developed to detect botnets utilizing machine learning techniques, namely Logistic Regression, Decision Tree, and Histogram-based Gradient Boosting (chapter 3). The main objective is to improve accuracy, precision, recall, and f-1 score, while also reducing the computational time in comparison to current methodologies.
- The features of the dataset will be analyzed in more detail than the current literature (chapter 4). The correlations and the behavior exhibited by the malicious and benign data will be identified and scrutinized through the utilization of kernel density estimates, heatmaps, and the SelectKBest methods. The optimization of feature selection will prioritize the selection of the minimum number of features while maintaining optimal performance. This will result in expedited training and testing computations.
- There will be a critical analysis of the literature (chapter 2). With an emphasis on addressing the limitations of the state-of-the-art and highlighting how the proposed work will improve it.
- The proposed model will be evaluated by comparing the accuracy, precision, recall, f1-score, and computational time against the existing literature (chapter 5). Furthermore, the model will be assessed on both uneven and even distributed datasets to ensure reasonable and realistic scores.

1.4. Thesis structure

To realize the aforementioned contributions several steps need to be taken. The subsequent chapters of the report are structured in the following manner. In chapter 2 the existing literature will be examined pertaining to the contemporary IoT botnet attacks, fundamental aspects of botnets, a critical evaluation of the current detection schemes, and the details of the dataset. In chapter 3 and chapter 4 the design and implementation of the feature selection and the hybrid model will be discussed. In chapter 5 the results of the method will be compared against the prior research. In chapter 6 and chapter 7 potential future improvements and the conclusion of this research will be discussed.

2

Background and Related Work

Numerous variations of IoT botnets exist, yet they do exhibit a common structure and the majority share numerous similarities. This chapter delves into the concept of botnets and presents the latest research on botnet detection from published work.

2.1. Botnets

To comprehend the functioning of a botnet, several steps must be undertaken. First, an examination of the life cycle of botnets, as it appears to be the most common trait shared among multiple botnet variants. Second, it is crucial to understand the types of attacks executed by the botnets and the harm they can inflict.

2.1.1. Lifecycle

The software architecture of an IoT botnet can vary depending on several factors, including the targeted vulnerabilities, devices, and attack types. As a result, identifying a common structure in their architecture can be challenging.

Nevertheless, some studies have attempted to generalize botnets. Although the terminology used to describe the phases of the IoT botnets may differ, the descriptions of these phases are similar.

In [11], a survey is conducted and the distinction between traditional botnets and IoT botnets is discussed, along with various software architectures and detection techniques for botnets. The life cycle is explained in four steps. First, an initial infection of the device. Followed by a connection with the command and control, attacks using the botnet, and finally a search for more vulnerable devices to convert into bots.

In [1], the life cycle is broken down into five steps. Seeking out vulnerable devices, followed by accessing the target device, infecting the device, establishing communication with the command and control, and executing the command and control's orders to launch botnet attacks.

In [9], it is simplified further into three steps. Infecting devices into bots, communicating with the command and control, and executing malicious actions sent from the control center.

In [48], the architecture and life cycle of the botnet are explained in the most straightforward and comprehensive manner. In Figure 2.1 the life cycle of an IoT botnet is broken down into three phases.

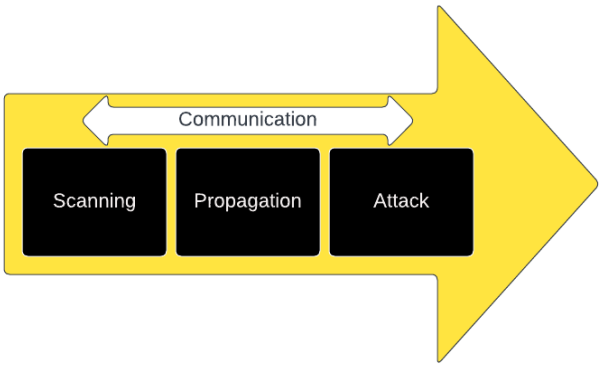

Figure 2.1: IoT Botnet life cycle phases

The phases consist of scanning, propagation, and attack:

- **Scanning phase**: In the initial stage, the adversary conducts a search for vulnerable devices. Upon discovering one, the adversary employs either brute force or an exploit to gain access. Once the device has been compromised, it transforms into a bot and begins to communicate with the botmaster through the command and control.
- **Propagation phase**: The second stage entails installing and executing a bot's source code on the device. Once this procedure is completed, the

device transforms into a bot and integrates into the botnet. Several measures could be implemented during this phase to conceal the source code or other indicators of the device's compromise. The code also instructs the bot to search the network for more vulnerable devices. However, the bot remains inactive until directed by the botmaster to initiate a botnet attack.

- **Attack phase**: During this phase, the bots are instructed by the botmaster to execute an attack. This can be a DDoS attack, cryptocurrency mining, spamming, and other similar activities. The botmaster transmits the instructions through the command and control system, which guarantees that all bots receiving the command will execute the same action.

The architecture required to execute the necessary operations is displayed in Figure 2.2, which includes the attacker/botmaster, the malicious infrastructure, the IoT devices, and the target for the attack. The components are listed and numbered in conjunction with the life cycle of the botnet.

During steps 1 and 2 the scanning phase is executed, where the bots in the botnet search for vulnerable IoT devices, also known as victims. This is informed to the malicious infrastructure consisting of the command and control and the servers utilized for deploying the botnet source code.

The propagation phase is executed during steps 3 and 4, where the known vulnerable devices are accessed, and the bot source code is installed. Once this is accomplished, the device will be integrated into the botnet under the botmaster.

Finally, the attack phase will be executed during steps 5-7, wherein the botmaster issues the command to attack the target. The command is transmitted to all the bots in the botnet by the command and control, following which the attack on the target is initiated.

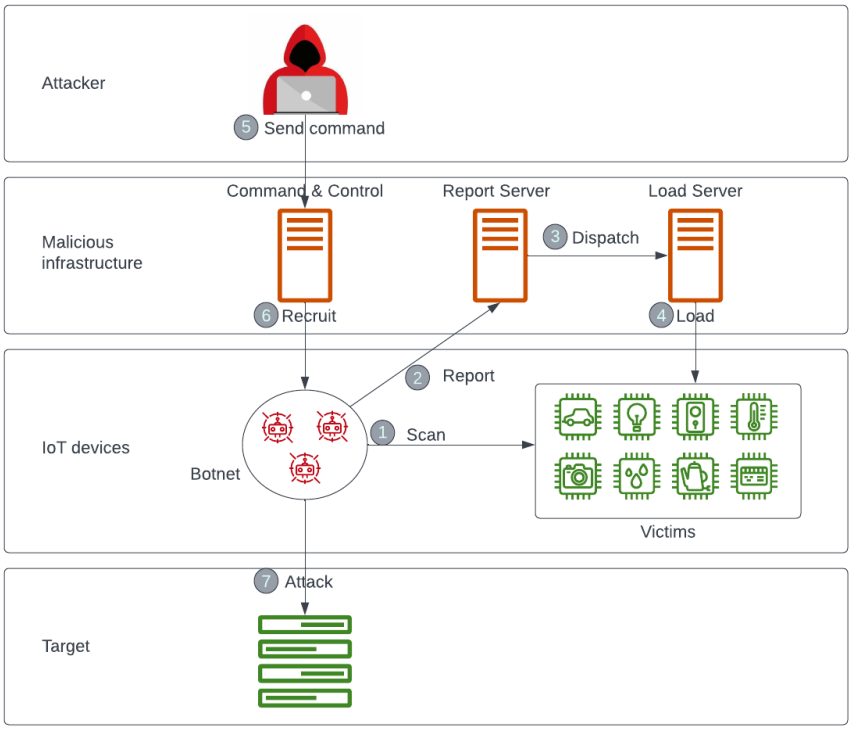

Figure 2.2: IoT Botnet architecture

2.1.2. Botnet attacks

Having conducted an analysis of the infrastructure and life cycle of the IoT botnet, it is crucial to take into account the diverse forms of attacks executed on the target. This will be accomplished by examining various botnet versions to comprehend the modus operandi of these attacks.

In [42], a survey was conducted on IoT botnet attacks, including an analysis of the various motivations behind such attacks. The study identifies two main incentives for botnet attacks: targeting specific individuals or groups and improving one's financial status. The paper also examines different types of botnet attacks, such as Distributed Denial-of-Service(DDoS) attacks, brute-forcing credentials, and phishing data from the affected devices, among others. The IoT botnet variant utilized often determines the precise attack.

Two infamous botnet variants will be discussed to provide a detailed analysis of the attacks, the Mirai and Bashlite botnet [46]. The Mirai botnet, in particular, gained notoriety following its attack in 2016, making it one of the most well-known and researched botnet variants in the literature. Numerous research focus solely on studying the botnet. In [19], 7,500 IoT honeypots were used to examine the infrastructure and behavior. The paper also explores potential strategies to stop the botnet. In [22], the Mirai infection process is modeled and simulated. The main objective is to comprehend the Mirai botnet infection process and evaluate the efficacy of rebooting the infected IoT device as a preventive measure. In general, the Mirai botnet conducts scans on vulnerable TCP/UDP ports and brute forces with a set of default credentials to gain access to the device. This process is repeated across multiple devices to establish a substantial botnet, which subsequently executes DDoS attacks on large organizations.

The Bashlite and Mirai botnets share several similarities, primarily due to the fact that the source code of Mirai is based on that of Bashlite. As a result, Mirai is often regarded as a successor to Bashlite [5]. However, there are still notable differences, and therefore Bashlite variants are identified separately from Mirai variants. In [31], the continued development of both botnet variants is outlined, along with the distinctions between them. The Bashlite has more set and hardcoded configurations compared to Mirai when selecting vulnerable devices and issuing commands to the botnet. The Bashlite also sends out more commands than a Mirai botnet. These commands are in plain text, whereas Mirai has encrypted ones. The naming of the commands differs per variant, making it challenging to differentiate. In contrast, the Mirai supports ten separate attacks and employs a binary protocol to initiate them, making it easier to distinguish. Furthermore, once an IoT device is infected, it does not always search for new vulnerable devices, unlike the Mirai variants. Apart from these differences, the process for creating bots and the type of attacks are similar.

There exist additional variants and botnets that have received less scrutiny in the current state-of-the-art, in contrast to the aforementioned two. This may be attributed to the limited availability of source code or the fact that these botnets target specific operating systems. Two such botnet variants are the Tsunami and Dofloo botnets, which are distinct from the Mirai and Bashlite variants, as stated in [15][14]. There are no studies that profile these two botnets like the Mirai and Bashlite variants, resulting in limited comprehension of the infrastructure. The Tsunami botnet is deployed on the SSH server. It infiltrates Linux operating systems by brute-forcing the login credentials. Once the device is transformed into a bot the communication is maintained through the IRC protocol. The botnet is primarily utilized for DDoS attacks.

The Dofloo botnet mainly targets devices with misconfigured Docker services. It scans to identify exposed Docker APIs on port 2375 and subsequently deploys malicious payloads containing the Dofloo source code to gain access to these devices. Once the device has been compromised, the botnet can be deployed to launch DDoS attacks and load cryptocurrency miners onto the infected machines.

2.2. Detection schemes

Studies have been conducted with the aim of detecting IoT botnets. Therefore it is imperative to obtain an overview of the recent studies, while also identifying limitations.

An analysis of recent literature reviews, surveys, and works on IoT botnet detection reveals key factors in this domain. [48][42][26][4]. The important factors are listed as follows:

- **Dataset**: Maintaining an up-to-date dataset is of paramount significance, as novel botnet variants will exploit new vulnerabilities or execute novel types of attacks. These exploits and attacks should be incorporated into the dataset, enabling the model to detect these new scenarios.
- **Feature analysis and selection**: The analysis of the extracted features from the data is imperative for the training of the model. It is essential to identify the features that can distinguish between malicious and benign data. While utilizing all features may yield satisfactory outcomes in certain scenarios, selecting a subset of features can enhance the model's computational speed and reduce resource consumption during real-time deployment.
- **Detection model**: The identification of malicious and benign data within the datasets is determined upon the utilization of this model. The selection of the classifier integrated into the model or the model's design will serve as the decisive factor, and therefore, the most critical aspect.

Table 2.1: Overview of the detection schemes

Ref.	Year	Detection model	Number of features utilized	Strengths	Limitations
[12]	2022	KNN,GNB,SVC	All	Multiple datasets and models. Uses a balanced dataset to avoid overfitting.	No computational time mentioned. No comparison with other studies.
[23]	2020	DT, ETC, RFC, SVM	All	Multiple models.	Slow computational time for best-performing models. No comparison with other studies.
[2]	2021	LR	19	Good performance.	Computational time not mentioned. Limited comparison. Uses little data for training and testing.
[33]	2018	SVM, Iso. For.	3, 5, 10	Multiple models, feature selection methods, and aims for feature optimization. Examines the performance with unbalanced and balanced sets.	No comparison with other studies. Requires different models for each device. The performance with the balanced set is okay.
[6]	2021	LR,KNN,SVM	10	Multiple models. Looks into multi-classification. Performance is good.	Slow computational time. No comparison with other studies.
[49]	2022	Deep Forest	6	Good performance.	Limited comparison. No computational time mentioned.
[28]	2022	Optimized Deep AE	22, 28, 31, 35, 36	Optimized auto-encoder to improve accuracy and reduce complexity compared to other deep learning methods. Good performance.	Requires different features per device. No computational time mentioned. No comparison with other studies.
[20]	2022	CNN	23	Good performance	No computation time mentioned. No comparison with other studies.
[10]	2022	Deep AE	All	Good performance	No computational time mentioned. No comparison with other studies.
[39]	2022	SGAN	23	Multi-classification. Good performance. Good comparison with other studies.	Uses little data for testing and training. Slow computational time.
[44]	2022	Stochastic Game Models	All	Uses game modeling to detect malicious data and predict how many devices are infected.	No computational time mentioned. No comparison with other studies.
[3]	2022	Image Processing	23	Uses image processing for feature selection. Good performance.	No computational time mentioned. No comparison with other studies.
[21]	2022	DNN-LSTM	All	Multi-classification. Good performance. Great comparison with other studies.	No computational time mentioned. Uses little data for testing and training.
[41]	2023	IHHO-NN	15	Multi-classification.	No comparison with other studies. Slow computational time.
[47]	2022	CVAE-RE-KLD	All	Good performance	No comparison with other studies. Limited statements on the computational time.

The N-BaIoT dataset [32][26] has emerged as a frequently utilized resource in recent studies. Its utility lies in the inclusion of a diverse range of attack scenarios, some of which were unexplored in previous datasets [34].

Numerous studies have utilized this dataset, as observed in Table 2.1. Several studies already covered baseline classifiers, such as Logistic regression and Support Vector Machines, among others [12][23][2][33][6][49]. Other popular classifiers are neural networks, particularly autoencoders [28][20][10]. Some instances also utilize game theory and image processing as detection techniques [39][44][3]. Finally, there are hybrid models, detection models that combine multiple classifiers, although these models appear to mostly consist of neural networks [35][21][41][47]. It can be observed that the models deploying neural networks tend to be slower and less adaptable.

Another issue is the feature selection. Most of the aforementioned papers utilize a feature selection approach that involves scoring and ranking the features, with the highest-ranking features being chosen for model training and testing. However, there is a lack of analysis demonstrating the behavior of the malicious and benign data in each feature. Furthermore, there is no argumentation for the rankings or why the selected features would effectively identify malicious data. On top of that, the studies do not aim to optimize the number of features, which involves minimizing the number of features while maintaining good performance. Instead, all studies employ an arbitrary number of features without attempting to minimize this quantity.

2.3. N-BaIoT

The N-BaIoT dataset was computed in 2018 [32]. According to IEEE, the dataset has been referenced in 570 papers during this period. This dataset has gained recognition due to its collection of data pertaining to a multitude of attacks on diverse IoT devices that were deployed in a laboratory setting.

There are nine distinct devices in total, including doorbells, thermostats, and cameras. These are commonplace devices from various companies, indicating diverse firmware. The attacks on these devices utilize two public source codes of the Mirai and Bashlite botnets. The attacks conducted in the laboratory are specific and encompass multiple phases of the botnet life cycle. This includes scanning attacks to identify vulnerable devices, as well as flooding, spamming, and DDoS attacks of the later stages. Each botnet source executed a total of five different attacks, resulting in a total of ten different attacks for each device. In addition, some unique instances such as the zero-case scenario were also taken into consideration [34].

The dataset contains numerous features. There are 23 metrics computed over 5 sliding windows, culminating in a total of 115 features. The sliding windows have a time delay of 100 ms, 500 ms, 1,5 sec, 10 sec, and 1 min. The 23 metrics are aggregates that encompass packet size, package count, and their means across various connection sources, such as IP, MAC-IP, ports, among others.

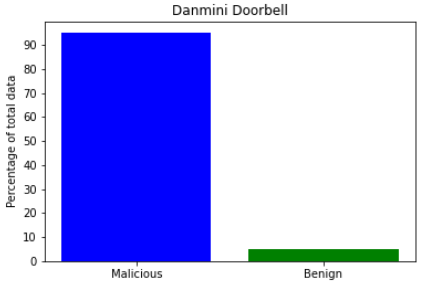

Figure 2.3: Distribution of the malicious and benign data in the Danmini Doorbell device

There are several issues related to the utilization of the dataset in research studies. First, each device comprises approximately one million data points. Owing to the extensive range of attack scenarios considered, the number of malicious data points significantly exceeds the benign data points, as evidenced in Figure 2.3. The distribution of the remaining devices is outlined in section A.1. This results in overfitting, as even if all data points are predicted as malicious, the accuracy score remains high, despite the model's inadequate performance, as remarked by some studies [33]. For this reason, the dataset that employs devices with uneven distribution will be referred to as the unbalanced dataset, while the balanced dataset will utilize an equal number of malicious and benign data points.

The second issue is the features. The inclusion of all 115 features would result in longer training and testing times for the detection techniques. Moreover, the practical application of the model in real-world scenarios would pose a challenge, as computing 115 features for each incoming package is not feasible for a network. Therefore, it is crucial to optimize the feature selection for this dataset.

3

Methodology

Having identified the state-of-the-art and the associated issues, it is time to consider the method that will be utilized to improve it. The feature selection and detection model are components that require design and discussion. In Figure 3.1, the complete process is illustrated. The steps required are sequentially numbered.

There are a number of procedures that require discussion. Initially, in steps 1 and 2 the N-BaIoT dataset is read in and preprocessed. This is followed by steps 3 and 4, which involve the feature selection resulting in the final set that comprises the data for the selected features. Finally, the hybrid ensemble model is generated, serving as the detection model encompassing the processes from steps 5 to 10. The final labels computed in step 10 will contain the predicted labels generated by the hybrid ensemble model, which will be utilized for evaluation.

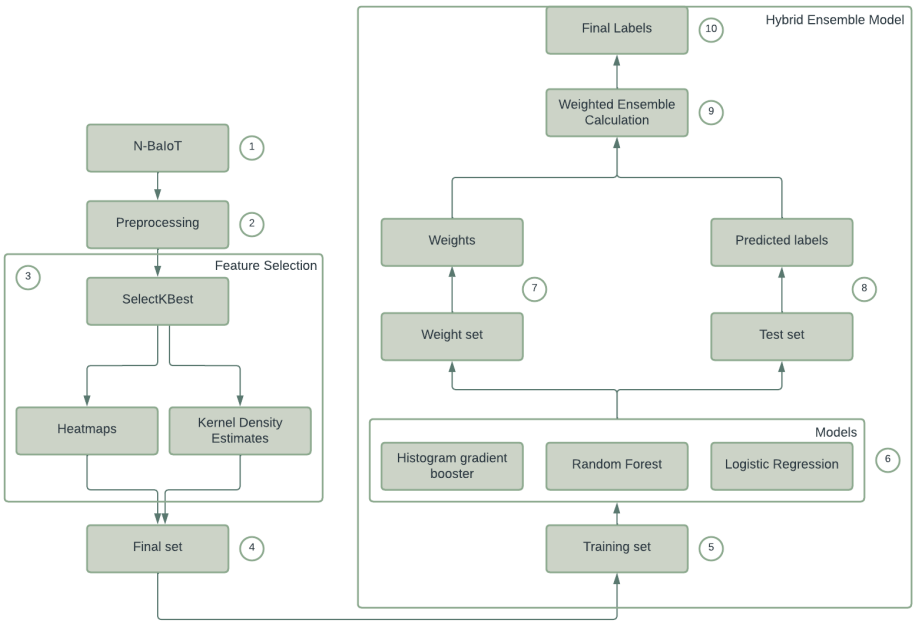

Figure 3.1: The diagram with steps of the methodology

3.1. Feature selection

Upon completion of data reading and preprocessing for the N-BaIoT, the subsequent step involves feature analysis and selection. This is executed through three techniques: the SelectKBest function, kernel density estimate plots, and heatmaps.

3.1.1. SelectKBest

The initial method will involve utilizing a univariate feature selection technique known as SelectKBest. This method has been employed in various studies [43][7]. The SelectKBest method selects the best features based on the outcomes of univariate statistical tests. It retains the K features with the highest scores and discards the rest. The scoring tests are conducted using a chosen function, such as chi-squared or F-tests, to estimate the degree of linear dependency between two random variables.

In this model, the SelectKBest function will be applied to each device individually to identify the features with the highest scores. This is due to the observed behavior of the features, which was initially determined through kernel density estimates. This will be elaborated further in chapter 4. The fundamental problem was that the densities of the benign and malicious data were comparable across all features. The highest-ranking features were determined by the scoring function utilizing F-tests. Finally, the frequency of each feature across all the devices will be computed to determine the optimal features for training and testing the hybrid ensemble model.

3.1.2. Kernel density estimates

The subsequent step involves the utilization of two methods, one of which is kernel density estimates. It is common practice to employ kernel density estimates for feature analysis [36][40]. Kernel density estimates are graphical representations that employ a kernel, such as Gaussian or linear, to determine the highest density for a particular value in the data. In essence, it is comparable to a histogram but with real values, no bins, and an emphasis on determining densities for the data points.

In this case, the kernel density estimates will be utilized on each feature to estimate the density of the malicious and benign data separately. The objective is to identify features that exhibit a distinct high density on malicious data and a dispersed density on benign data, or vice versa. This enables the features that distinguish the behavior between malicious and benign data to be highlighted. It will be applied to the features outputted by the SelectKBest method to optimize the number of features further for training and testing the hybrid model.

3.1.3. Heatmaps

Another method that will be employed during this process is heatmaps. Heatmaps have a wide range of applications [16][27]. While kernel density estimates examine features individually, heatmaps examine features collectively. Heatmaps can illustrate the relationship between two features based on the function utilized to calculate the correlation coefficient.

In this case, two heatmaps will be generated for each device to exhibit the Pearson correlation between the features. The first heatmap will display the malicious data and the second heatmap will display the benign data. The correlation coefficient will range from 0 to 1. If the coefficient between the two features is close to 0, it indicates that the features are uncorrelated, implying that the features are too dispersed and have no relationship with each other. If the coefficient is close to 1, it suggests that the two features have a high correlation, indicating the features are almost identical. Therefore, the coefficients should not be close to either 0 or 1. This approach will primarily be used on the features resulting from the kernel density estimates to encourage the selection of features by ensuring that the chosen features are not entirely uncorrelated and duplicates of one another.

3.2. Hybrid ensemble model

Upon completion of the feature selection process, the final set will be generated in step 4, encompassing all data of the selected features of the device. This set will be partitioned into the training, testing, and weight sets, which will be utilized for the purposes of model training, testing, and weight computation. Finally, the weighted ensemble calculation will be executed, to determine the final labels.

3.2.1. Classification models

The hybrid ensemble model comprises multiple steps. The initial process involves training the model using the training set, as outlined in steps 5 and 6. There are a total of three classifiers operating independently. Each model's training, test, and weight can be computed concurrently in this manner. The selection of the models was based on their individual performance on the test and weight sets, as well as their computation time for training and testing.

The classification models utilized in this study include Logistic Regressions, Random Forest classifier, and Histogram-based Gradient Boosting classification. These classification models are commonly employed in detecting botnets. Numerous studies have employed Logistic Regressions for botnet detection on the dataset [2][6]. The same applies to the Random Forest classifiers [23]. These models have demonstrated favorable computational time and detection performance. However, the third model proved to be more challenging. Most of the other classifiers recommended by the literature were found to be slow or ineffective at detecting with limited features. After investigation, Histogram-based Gradient boosting was identified as an effective method. It offered detection performances superior to Logistic Regression and comparable to the Random Forest classifier. However, it was found to be slower than the other models. But still outperformed models such as Support Vector Machines, K-nearest neighbors, and Stochastic gradient models, among others.

There are numerous advantages to opting for a hybrid model. Ensemble models operate on the philosophy that utilizing a group of experts is more effective than a single expert [18][50]. An example is the Random Forest classifier, which employs multiple Decision trees for classification. Following this theory, combining multiple different classifiers should yield superior outcomes than utilizing them individually. Furthermore, the models can be employed in parallel, allowing for separate execution of the train, test, and weight calculations for each model, as can be seen in Figure 3.1. Therefore, assuming three models M_1, M_2 and M_3, then the computational time for each model is:

$$Mt_1 = tr_1 + ts_1 \tag{3.1}$$
$$Mt_2 = tr_3 + ts_3 \tag{3.2}$$
$$Mt_3 = tr_3 + ts_3 \tag{3.3}$$

Here, Mt is the computational time for the model, tr is the training time, and ts is the test time. It is also feasible to calculate the weights in parallel with the test set predictions. This is primarily due to the fact that the weight set is smaller than the test set, and these two computations can be executed independently. Given these considerations and the weighted ensemble calculation $enst$, the total computation time for the hybrid ensemble model is:

$$T = max(Mt_1, Mt_2, Mt_3) + enst \tag{3.4}$$

In short, the maximum time for one of the three models to train and test along with the time to compute the weight ensemble calculation leads to the final label.

3.2.2. Weights

Upon completion of the training process, steps 7 and 8 follow. These two steps can be performed concurrently as aforementioned. In step 7, a weight set is generated, which constitutes a minor portion of the final set in step 5, prior to its division into the training and test sets. The weight has to be dynamic and calculated fast to minimize computational time. Given that the model is intended to analyze the devices individually, fixed values may not be optimal for all devices. The next fastest option is to utilize the weight set for each model, and accuracy scores are computed to serve as weights, namely w_1, w_2, and w_3. In the event that a model performs poorly on a specific device, its weight will be reduced, and its prediction will have a lesser impact on the final score.

3.2.3. Predictions

During step 8, the test set will be utilized to compute the prediction labels for each model. The predicted label will be computed for each data point in the test set, as the accuracy score will be determined only after the final labels are computed in step 10. Following this step, three vectors will be generated for each model, containing the predicted labels. Given that the test set is larger than the weight set, it is reasonable to expect that this step will take longer than step 7.

3.2.4. Weighted ensemble calculation

Upon computation of the weights and predicted labels for each model, the two components will be utilized in the weighted ensemble calculation, as outlined in step 9. This calculation involves combining the weights and predicted labels to compute the final label for each data point. Assuming there are four sets, W for the weights, L for the predicted labels of the Logistic Regression, R for the predicted label of the Random Forest classifier, and H for the predicted labels of the Histogram Gradient boosting classifier then:

$$W = \{w_1, w_2, w_3\} \tag{3.5}$$
$$L = \{l_1, ..., l_n\} \tag{3.6}$$
$$R = \{r_1, ..., r_n\} \tag{3.7}$$
$$H = \{h_1, ..., h_n\} \tag{3.8}$$

In this context, w_1 is the weight for the Logistic Regression, w_2 is the weight for the Random Forest classifier and w_3 is the weight for the Histogram Gradient Boosting classifier. The weighted average will be calculated for each data instance of the test set with index i. If the weighted average is above 0.5 then the label is set to 1 else to 0. As shown in:

$$f(i) = \begin{cases} 1, & \text{if } \frac{w_1 * l_i + w_2 * r_i + w_3 * h_i}{\sum_{j=1}^{3} w_j} \geq 0.5 \\ 0, & \text{otherwise} \end{cases} \tag{3.9}$$

The proposed approach is characterized by its expeditiousness, simplicity, and adaptability, which ensures the dependability of the final predicted label. This underscores the necessity of employing three models, as the third model serves as a decisive factor in the event of opposite predictions and identical weights between two models. Upon completion of this computation for all instances, the resultant set of computed labels will serve as the final predicted labels. Subsequently, evaluation scores will be computed over this set and compared with existing literature.

4

Implementation

Having discussed the methodology, the subsequent phase involves its implementation. This will be executed via a Jupyter Notebook utilizing Python 3.7 as the kernel. The main libraries employed in this process are pandas, scikit, matplotlib, numpy, seaborn, and time. These components will facilitate the implementation of the data preprocessing, feature analysis and selection, and the hybrid ensemble model, which will be discussed.

4.1. Preprocessing

The preprocessing stage commences subsequent to the loading of a device's data from the N-BaIoT dataset. This process comprises three steps. First, the data must undergo normalization to reduce the impact of outliers on the predictions. This normalization enhances the stability of the training and prediction. Failure to normalize the data may result in negative performance effects caused by outliers, which can inadvertently inflate scores. The MinMaxScalar function, an in-built method, has been employed for this purpose. This method is a standard scalar and has been utilized previously in the literature, thereby ensuring reliability in future comparisons. The method scales feature values within a specific range, with the default range $(0, 1)$ utilized in this instance.

The second phase involves creating the balanced set. Utilizing the dataset in its current state would yield the unbalanced set, characterized by the uneven ratio between malicious and benign data. Prior to anything else, the ratio between the malicious and benign data must be calculated. This ratio will then be applied in the sampling method of the panda library, which requires a fraction to determine the sample size. The balanced set will be generated by means of random sampling of the malicious data, resulting in a 50/50 distribution.

Finally, the dataset will undergo the process of label addition. The labels assigned will be 1 for the malicious data and 0 for the benign data. The resulting labels will be incorporated into a new column within the panda data frame, alongside the existing features.

4.2. Implementation of the feature selection

After the data preprocessing, the subsequent stage involves the feature analysis and selection. In section 3.1, the argumentation behind the selection methods along with their application was discussed. The implementation and the features that have been selected will be displayed in the following section. As before, it consists of three processes SelectKBest, kernel density estimates, and heatmaps.

4.2.1. Implementation of the SelectKBest

The scikit library features an integrated function for the SelectKBest method. As indicated in section 3.1, the method will be executed on each device to obtain the top three features, as illustrated in Listing 4.1

Listing 4.1: Code for the SelectKBest method

```
1 Kbest = SelectKBest(k=3)
2 X_new = Kbest.fit_transform(X, y)
3 list(compress(train_columns, Kbest.get_support()))
```

Here, X corresponds with the device's data encompassing both malicious and benign data, y with the labels for malicious and benign data, and k with the number of features to be selected based on their scores. A list of the top three features of the device is generated, which will be applied to all nine devices. The frequencies of the selected features across all nine devices are presented in Table 4.1

Table 4.1: Best scoring features across all devices

Feature	Frequency
MI_dir_l1_weight, HH_l0.01_std	3
HH_l0.01_pcc, MI_dir_l0.01_mean, H_l0.01_mean, MI_dir_l0.1_weight, H_l0.1_weight, HH_l0.1_std, HH_l0.01_radius	2
H_l1_weight, H_l0.1_mean, H_l0.01_variance	1

Based on the table, two features should be incorporated into the selection, as they have received consistently high scores across three distinct devices. The subsequent features will be selected from the next highest frequency. However, these are numerous and require further analysis through the use of kernel density estimates and heatmaps.

4.2.2. Implementation of the kernel density estimates

The kernel density estimates will be applied to analyze the distinct behavior between the malicious and benign data per feature. At first, the kernel density estimates were computed for all features and devices, and these plots can be found in section A.2. It should be noted that several features exhibit a clear distinction between malicious and benign data by demonstrating a variation in density. Therefore, all these features might serve as suitable candidates for training and testing the models. This explains why some studies utilize numerous or even all features and still achieve good results [34][28][23]. So, the SelectKBest method was employed to reduce the number of features until only the best remained.

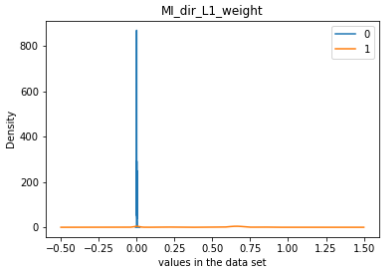

Figure 4.1: MI_dir_L1_weight feature of the Danmini doorbell device

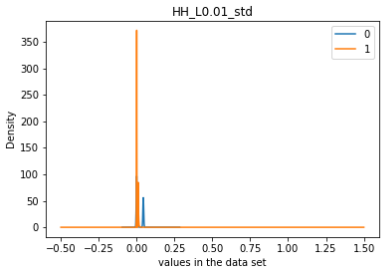

Figure 4.2: HH_L0.01_std feature of the Danmini doorbell device

In Figure 4.1 and Figure 4.2, the kernel density of the top two aforementioned features is depicted for the Danmini doorbell. This figure illustrates the reason for selecting the features. The $MI\_dir\_L1\_weight$ feature exhibits a distinct peak for the benign data, indicating a high density for a particular value, while the malicious data is more sparse. This demonstrates that the feature is useful for identifying benign data. On the other hand, the $HH\_L0.01\_std$ feature displays the opposite pattern, with a peak for the malicious data and a much lower peak for the benign data, which is also more spread out. This suggests that the feature is effective at identifying malicious data. Having one feature that is adept at identifying malicious data and another one at identifying benign data simplifies the task for the classifiers to discriminate between the two.

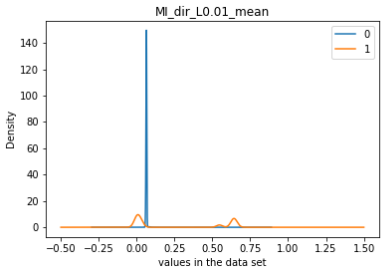

Figure 4.3: MI_dir_L0.01_mean feature of the Danmini doorbell device

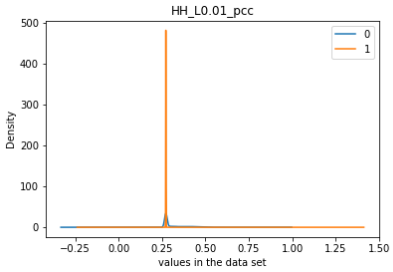

Figure 4.4: HH_L0.01_pcc feature of the Danmini doorbell device

On top of the aforementioned features, two additional features have been selected as illustrated in Figure 4.3 and Figure 4.4. The selection of these features was based on their similarity to the top two features, with each feature sharing a common characteristic with one of the top two features within each device. The main distinction is the distribution of the density between malicious and benign data. These four features will play a pivotal role in the performance of the model, which will be further elaborated in chapter 5.

4.2.3. Implementation of the heatmaps

The heatmaps will be utilized to elaborate on the interdependence between the features. The main objective is to ascertain that they are not entirely uncorrelated and that there is no excessive duplication between the features.

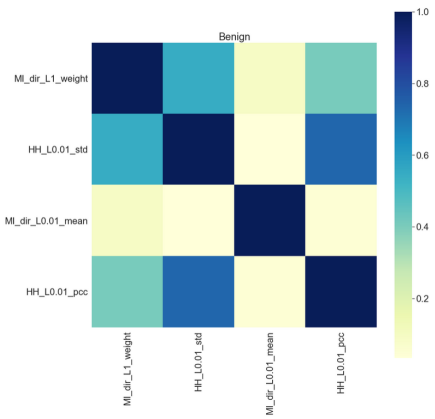

Figure 4.5: Heatmap of the features on the benign data of the Danmini doorbell device

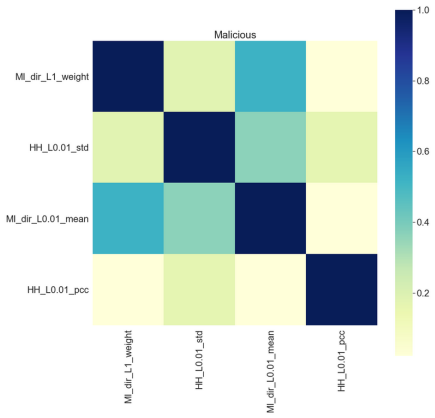

Figure 4.6: Heatmap of the features on the malicious data of the Danmini doorbell device

In Figure 4.5, the heatmap displays the Pearson correlation among the benign data. In Figure 4.6, the same can be observed for the malicious data. The absolute values were computed for the Pearson correlation, to enhance the readability of the heatmaps. Features with values greater than 0 and less than 1 are deemed suitable for selection. Although certain correlation coefficients may appear low when comparing the malicious and benign data individually, an overall examination reveals that the correlation coefficients fall within a reasonable range.

4.3. Implementation of the hybrid ensemble model

The hybrid ensemble model consists of four steps, namely model training, weight computation, test set predictions, and weighted ensemble calculation. The implementation shall be discussed through the utilization of the source code.

Listing 4.2: Code for initializing and training the classifiers

```
LR_model = LogisticRegression();
RF_model = RandomForestClassifier();
grad_model = GradientBoostingClassifier()

start_LR = time.process_time();
LR_model.fit(X_train,y_train);
end_LR = time.process_time()-start_LR

start_RF = time.process_time();
RF_model.fit(X_train,y_train);
end_RF = time.process_time()-start_RF

start_third = time.process_time();
grad_model.fit(X_train,y_train);
end_grad = time.process_time()-start_third
```

In Listing 4.2, the code for initializing the classifiers and conducting the training process can be found. The code is designed in a clear and concise manner. First, the three models are initialized with the default parameters. This is followed by training each model using the training set X_train, and the corresponding labels y_train. Simultaneously, the computational time is determined by calculating the difference between the start and end times of the model.

Listing 4.3: Code for the weight calculation

```
weight_1 = LR_model.score(X_weight,y_weight);

weight_2 = RF_model.score(X_weight,y_weight);

weight_3 = grad_model.score(X_weight,y_weight);

weights = [weight_1, weight_2, weight_3]
```

The weight calculation for each model is displayed in Listing 4.3. The code is straightforward, as the accuracy score is determined by comparing the predicted labels of the X_weight set with the ground truth labels y_weight. The resulting weights are then stored in the $weights$ array.

Listing 4.4: Code for the predictions

```
start_LR = time.process_time();
LR_label = LR_model.predict(X_test);
end_LR = time.process_time()-start_LR

start_RF = time.process_time();
RF_label = RF_model.predict(X_test);
end_RF = time.process_time()-start_RF

start_third = time.process_time();
Grad_label = grad_model.predict(X_test);
end_grad = time.process_time()-start_third
```

In Listing 4.4 the predictions over the test set are computed. Each model predicts the test set X_test, and the predicted labels are stored. Again The computational time is determined by calculating the difference between the start time of the model and the time at which the prediction is completed.

Listing 4.5: Code for the weighted ensemble calculation

```
start_hybrid = time.process_time();
final_labels = []
for i in range(len(LR_label)):
    predicted_class = (weights[0]*LR_label[i]+weights[1]*RF_label[i]+
        weights[2]*Grad_label[i])/(weights[0]+weights[1]+weights[2]);
    final_labels.append(int(round(predicted_class)))

end_hybrid = time.process_time()-start_hybrid
```

Upon completion of the computation of the weights and predicted labels, the weighted ensemble calculation commences, as depicted in Listing 4.5. The resultant labels will be computed using the weights and predicted labels from each model and are then stored in the array $final\_labels$. These labels will be utilized for evaluation. Furthermore, the computational time is calculated in a similar manner as the training and test time.

It is evident that each of the models can operate concurrently. Each section of the code where a component is computed uses the models independently. The code is designed to be simple and straightforward, thereby minimizing computational time while still delivering commendable performance.

5

Evaluation

Having presented the implementation, the next step is to proceed with the evaluation of the method. This will entail an initial assessment of the model's performance, as well as an examination of the features selected. Ultimately, the model's performance will be compared to relevant studies.

5.1. Performance

To assess the performance of the model, a number of actions must be taken. First, the evaluation metrics will be established to measure the efficacy of the model. This will be followed by an examination of the selected features and the rationale behind their selection over other options. The data of these features will then be incorporated into the hybrid model, from which the metrics will be computed. This study will also examine the performance of the hybrid model as the number of selected features increases or decreases to determine the advantages and disadvantages of such changes.

5.1.1. Evaluation metrics

The selected metrics have been utilized in related studies [35][28][33][12]. By employing these same metrics, it becomes easier to assess the effectiveness of the model and compare it to other studies. The metrics used are $Accuracy$, $Precision$, $Recall$, and $F1\text{-}score$. These metrics are calculated using the variables $True\ Positive(TP)$, $False\ Positive(FP)$, $True\ Negative(TN)$ and $False\ Negative(FN)$, according to the following formula:

$Accuracy$ is the metric that assesses the classification model by examining the number of correct predictions to the entire dataset:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{5.1}$$

$Precision$ is the metric that examines the correctly identified malicious data to the entire data identified as malicious:

$$Precision = \frac{TP}{TP + FP} \tag{5.2}$$

$Recall$ is the metric that examines the correctly identified malicious data to the entire malicious data:

$$Recall = \frac{TP}{TP + FN} \tag{5.3}$$

$F1\text{-}score$ is the measurement of accuracy using precision and recall:

$$F1\text{-}score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{5.4}$$

In addition to the aforementioned four metrics, the computational time will be computed throughout the training, testing, and weighted ensemble calculation processes. The overall computational time has been previously addressed in Equation 3.4.

It is imperative to note that the five metrics will be assessed on both the balanced and unbalanced datasets individually. This approach will generate a reliable comparison of the outcomes with previous studies that have acknowledged this issue and incorporated it into their evaluation. Additionally, it will highlight the variations in performance based on the distribution of the dataset.

5.1.2. Selected features

The model has been trained and tested using a number of three features, which is lower than the number of features utilized in other studies, including those referenced in [21][28][23][39]. These studies employ three, five, or even ten times as many features, resulting in a near-linear increase in computational time. The studies often compensate by reducing the amount of data used for training and testing. This study circumvents this by utilizing the unbalanced set, which contains all data of the device. Furthermore, the evaluation is performed separately for each device to facilitate comparison with other studies.

The features that have been selected for the model have been discussed in section 4.2. There are a total of four features, namely $MI\_dir\_l1\_weight$, $HH\_l0.01\_std$, $MI\_dir\_L0.01\_mean$, and $HH\_L0.01\_pcc$. However, since only three features will be utilized in the model, two different sets of features will be created for training and testing. Both sets will include the two highest scoring features: $MI\_dir\_l1\_weight$ and $HH\_l0.01\_std$ as shown in Table 4.1. The third feature in each set will be one of the remaining two selected features, either $MI\_dir\_L0.01\_mean$ or $HH\_L0.01\_pcc$. This approach has been adopted to examine whether alternating between the features that have a high density for either the malicious or benign data, as depicted in Figure 4.3 and Figure 4.4, will have an impact on the model's performance.

Furthermore, this setup will employ a 10-fold cross-validation, which has been implemented in several studies[21][33]. Other studies also employed a cross-validation but with fewer folds or a test/train split[12][23]. However, the utilization of these alternative approaches resulted in higher scores. Therefore, the 10-fold cross-validation technique was selected for this study.

5.1.2.1. Set with HH_L0.01_pcc feature

The first set to examine will be the one containing the $HH\_L0.01\_pcc$ feature. The main objective is to evaluate the performance of two features that exhibit a higher density of malicious data than benign, $HH\_l0.01\_std$ and $HH\_L0.01\_pcc$, alongside one feature that displays a higher density of benign data than malicious, $MI\_dir\_l1\_weight$. The rationale behind this approach is to incorporate two features capable of identifying malicious data and one capable of identifying benign effectively.

Table 5.1: The performance scores for the set with HH_L0.01_pcc feature

Device model	Unbalanced dataset				Balanced dataset			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
Danmini doorbell	99.39	99.38	99.98	99.68	96.84	94.17	99.96	96.98
Ecobee	99.80	99.80	99.99	99.90	96.16	92.97	99.92	96.32
Ennio	99.32	99.26	99.98	99.62	97.45	95.38	99.77	97.53
Philips	99.89	99.98	99.98	99.98	99.94	99.95	99.94	99.94
Provision PT 737E	99.73	99.73	99.98	99.86	98.59	97.38	99.89	98.62
Provision PT 838	99.53	99.52	99.95	99.74	98.61	97.53	99.83	98.67
Samsung	99.39	99.34	99.96	99.65	98.11	96.48	99.92	98.17
SimpleHome XCS7 1002 WHT	99.09	99.06	99.98	99.52	92.24	86.80	99.91	92.89
SimpleHome XCS7 1003 WHT	99.24	99.24	99.99	99.61	87.67	100.00	75.49	86.02

The results are presented in Table 5.1. In general, the performance evaluation yields a positive outcome. However, a clear distinction is observed between the unbalanced and balanced datasets. This outcome was anticipated, given the challenging nature of predicting the balanced dataset due to the distribution of malicious and benign data. The majority of the scores obtained from the balanced set are lower than those of the unbalanced set. The overall score of the unbalanced set is approximately 99%, while the balanced dataset has an overall score

of around 90%. The balanced dataset of the $SimpleHome$ devices exhibits the most impact, as both devices score lower than 90%.

Table 5.2: The computational time for the set with HH_L0.01_pcc feature

Device model	Unbalanced dataset			Balanced dataset		
	Training time	Test time	Calculation time	Training time	Test time	Calculation time
Danmini doorbell	86.12	0.09	0.376	6.602	0.016	0.032
Ecobee	82.25	0.075	0.344	1.755	0.002	0.009
Ennio	19.86	0.04	0.13	5.18	0.009	0.026
Philips	119.85	0.125	0.391	37.813	0.05	0.13
Provision PT 737E	84.44	0.094	0.31	9.978	0.022	0.05
Provision PT 838	83.605	0.105	0.306	17.016	0.033	0.0734
Samsung	19.395	0.044	0.13	5.595	0.014	0.036
SimpleHome XCS7 1002 WHT	90.57	0.092	0.31	7.07	0.015	0.032
SimpleHome XCS7 1003 WHT	85.28	0.078	0.31	2.204	0.006	0.014

In Table 5.2 the computational times are displayed, compromising three components, the training and testing time for the longest-running classifier, and the weighted ensemble calculation time as discussed in Equation 3.4. It can be concluded that the computational time is proportional to the size of the dataset employed. For instance, the Philips device has the largest dataset and therefore the highest computational time. Likewise, for the balanced dataset, the computational time is linked to the dataset with the highest number of benign data, as this dataset is computed by selecting an equal amount of malicious and benign data. The ensemble calculation is proportional to the size of the test data because the computational time is computed over the predicted labels.

5.1.2.2. Set with MI_dir_L0.01_mean feature

The subsequent set comprises the $MI\_dir\_L0.01\_mean$ feature. This set contains two features exhibiting a higher density of benign data than malicious data, namely $MI\_dir\_l1\_weight$, and $MI\_dir\_L0.01\_mean$, and one feature exhibiting a higher density of malicious data than benign data, namely $HH\_l0.01\_std$. An analogical reasoning to the previous set, albeit reversed, with two features capable of identifying benign data and one feature capable of identifying malicious data effectively.

Table 5.3: The performance scores for the set with MI_dir_L0.01_mean feature

Device model	Unbalanced dataset				Balanced dataset			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
Danmini doorbell	99.99	99.99	99.99	99.99	99.98	99.98	99.99	99.98
Ecobee	99.99	99.99	99.99	99.99	99.96	99.95	99.96	99.96
Ennio	99.95	99.97	99.97	99.97	99.95	99.97	99.93	99.95
Philips	99.97	99.99	99.97	99.98	99.97	99.98	99.96	99.97
Provision PT 737E	99.97	99.98	99.98	99.98	99.97	99.98	99.96	99.97
Provision PT 838	99.97	99.98	99.98	99.98	99.97	99.96	99.96	99.96
Samsung	99.96	99.99	99.96	99.98	99.96	99.96	99.97	99.96
SimpleHome XCS7 1002 WHT	99.99	99.99	99.99	99.99	99.98	99.98	99.98	99.98
SimpleHome XCS7 1003 WHT	99.99	99.99	99.99	99.99	99.92	99.88	99.96	99.92

The results are displayed in Table 5.3. The findings are excellent, with both unbalanced and balanced data achieving approximately $99,9\%$ on all metrics for each device. Although there is a slight variance in performance, with the unbalanced data yielding better scores than the balanced data, the distinction is almost negligible.

Table 5.4: The computational time for the set with MI_dir_L0.01_mean feature

Device model	Unbalanced dataset			Balanced dataset		
	Training time	Test time	Calculation time	Training time	Test time	Calculation time
Danmini doorbell	98.24	0.11	0.366	8.173	0.014	0.036
Ecobee	96.83	0.095	0.356	1.981	0.00625	0.0109
Ennio	22.92	0.047	0.134	5.605	0.014	0.03
Philips	124.87	0.125	0.389	37.813	0.05	0.13
Provision PT 737E	83.64	0.11	0.302	12.64	0.022	0.055
Provision PT 838	85.93	0.111	0.295	21.52	0.03125	0.0797
Samsung	24.567	0.038	0.129	7.57	0.016	0.036
SimpleHome XCS7 1002 WHT	92.73	0.109	0.303	8.153	0.014	0.033
SimpleHome XCS7 1003 WHT	84.228	0.078	0.303	2.95	0.009	0.016

In Table 5.4 the computational time is displayed for the set. The computational times are comparable to the preceding set. However, a slight decrease in speed is noted. This could be attributed to the presence of a program running in the background on the computer or the inherent complexity of the feature, which may pose challenges in training or testing the classifiers. The rationale behind the distribution and size of the dataset also appears applicable in this context.

5.1.3. Other features selections

In addition to the application of three specific features, other sizes were also examined to determine their relative performance.

First, an evaluation was conducted on the top two features, namely $MI\_dir\_l1\_weight$ and $HH\_l0.01\_std$, in conjunction with the aforementioned metrics to determine their performance. The results indicated a notable improvement in computational time by approximately $10 - 20\%$. However, this improvement was accompanied by a significant decrease in overall performance scores. The scores were comparable to the three feature sets with the $HH\_L0.01\_pcc$ in some instances. Nevertheless, the majority of the scores for the unbalanced set were approximately 90% instead of $99,9\%$ and the balanced set experienced an even greater decline.

In the second case, additional features were incorporated for evaluation. A total of five features were selected, including the original four, $MI\_dir\_l1\ \_weight$, $HH\_l0.01\_std$, $MI\_dir\_L0.01\_mean$, and $HH\_L0.01\_pcc$. Then, a fifth feature was added that exhibited a high density for benign data, $MI\_dir\_L0.1\_weight$. This feature was included to replicate the same rationale for the three features. This feature was also ranked in the second row of Table 4.1. Overall the performance did improve but by a small margin. The scores were approximately $99,99\%$, which is not a significant improvement compared to Table 5.3. Furthermore, the computational time increased almost proportionally to the number of features used, with an increase of approximately $30 - 40\%$ for all instances in the unbalanced set.

Considering these trade-offs, it is apparent that utilizing three features represents the optimal choice. This particular set exhibits great performance scores and fast computational time. Whereas increasing or decreasing the number of features will result in a detrimental effect on one or both of these metrics.

5.2. Comparison

The comparison between the performance of the proposed approach and other studies will be examined through three factors, namely the unbalanced dataset, the balanced dataset, and computational time. Each of these factors will be discussed in separate sections. First, a list of studies that employ the N-BaIoT dataset, along with an explanation for the selection of each study for comparison. Subsequently, a table and several visuals will be employed to display the scores of the studies, thereby facilitating a clear understanding of how the proposed method fares in comparison to the existing studies.

5.2.1. Unbalanced set

In this section, a comparative analysis of the proposed approach utilizing the unbalanced set will be conducted. To achieve this, multiple studies that also evaluate performance with an unbalanced set must be selected, which is readily available in the literature. As stated in chapter 2, it is a limitation of various studies. Furthermore, the average of each metric across all devices has been calculated, since comparing each device will be impractical.

5.2.1.1. Studies for the unbalanced set

The average has been computed across all devices for the proposed approach. This yields the following scores for the Accuracy, Precision, Recall, and F-1 score: 99.98%, 99.99%, 99.98%, and 99.98%. These scores will be compared to the other studies that have utilized an unbalanced set.

In [21], a hybrid model is proposed that combines DNN and LSTM to classify multiple samples obtained from various devices. The dataset used in this study has an unbalanced ratio of 1:3 for benign to malicious samples, thus chosen for comparison purposes. The evaluation metrics yield scores of 99.94%, 99.91%, 99.86%, and 99.86%. On top of this, the study includes comparisons with several other papers that also utilize an unbalanced dataset and the same performance criteria. Therefore three more studies referenced from this paper were included for comparison [38], [24], and [13]. In [38], a MPL-AE model is proposed with the scores of 99.25%, 98.84%, 98.00%, and 98.00%. A DBN model is proposed in [24] with scores of 95.60%, 98.27%, 92.82%, and 92.82%. For the CNN and LSTM models proposed in [13], the scores are 94.30%, 93.48%, 93.67%, and 93.58%.

In [39], an SGAN model is utilized. The paper puts more emphasis on multi-classification, mainly the distinct attack types in the N-BaIoT dataset. But also examines the binary classification by utilizing the complete Danmini Doorbell device. However, this approach solely evaluates the accuracy in comparison to other studies, with the SGAN model achieving a score of 99.88%.

In [28], an optimized auto-encoder is proposed. There are many features that have been utilized for each device in the dataset, ranging from 25 to 40. The dimensionality is reduced by an autoencoder, which reduces the number of features to two. These features are utilized for training and testing the model. Furthermore, the study computes the same metrics with scores of 99.68%, 99.73%, 99.43%, and 99.48%.

In [34], a federated deep learning method is proposed. This study employs and compares several deep-learning techniques. Regarding the distribution of the test and training sets, there are five batches containing different attacks from the dataset. Each batch is assessed individually. However, the distribution between the malicious and benign data is still uneven. For this reason, the average was computed over all the batches for each metric which resulted in the following scores: 98.98%, 98.26%, 97.88%, and 97.87%.

In [33] several techniques along with various feature sets have been analyzed. This study examines both unbalanced and balanced datasets and therefore will be applied to both cases. The study evaluates each device individually and only computes the accuracy and precision. The average of these two metrics was computed across all devices: 96.65% and 98.71%.

5.2.1.2. Summary of the comparison for the unbalanced set

Table 5.5: Comparison with existing studies for the unbalanced set

Reference	Year	Model	Accuracy(%)	Precision(%)	Recall(%)	F1-score(%)
Proposed	2023	LR-RFC-Grad	99.98	99.99	99.98	99.98
[21]	2022	DNN-LSTM	99.94	99.91	99.86	99.86
[39]	2022	SGAN	99.88	-	-	-
[28]	2022	AE	99.68	99.73	99.43	99.48
[34]	2022	FDL	98.98	98.26	97.88	97.87
[38]	2021	MLP-AE	99.25	98.84	98.00	98.00
[24]	2020	DBN	95.60	98.27	92.82	92.82
[13]	2020	CNN,LSTM	94.30	93.48	93.67	93.58
[33]	2018	SVM,Iso,LR	96.65	98.71	-	-

In Table 5.5 the results are presented. First, the study is mentioned along with the published year. This is followed by the proposed technique. The reference number of each study serves as the identifier in the histogram. Finally, the performance metrics are incorporated.

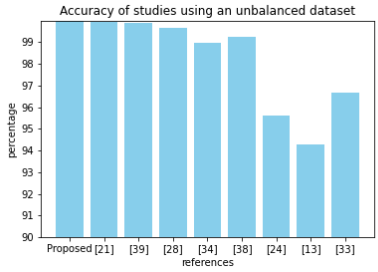

Figure 5.1: Accuracy scores of studies with the unbalanced dataset

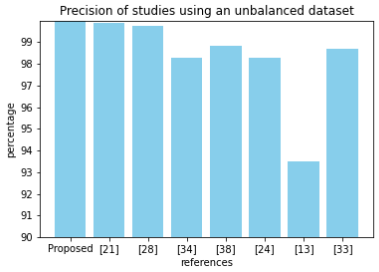

Figure 5.2: Precision scores of studies with the unbalanced dataset

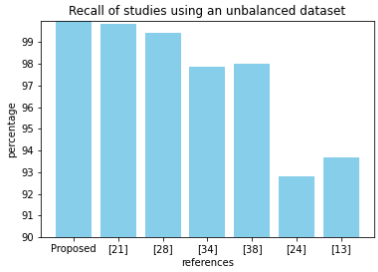

Figure 5.3: Recall scores of studies with the unbalanced dataset

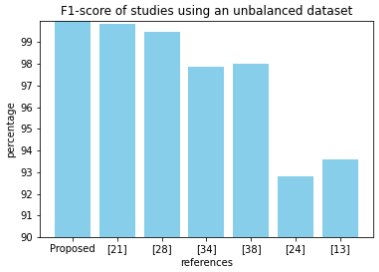

Figure 5.4: F-1 scores of studies with the unbalanced dataset

In Figure 5.1, Figure 5.2, Figure 5.3, and Figure 5.4 the histograms are displayed for each performance metric. Based on the results, it can be concluded that the proposed approach surpasses all the chosen studies for each metric. Despite the slight difference, the improvement is noteworthy considering the number of features utilized and the scores achieved.

5.2.2. Balanced set

In this section, we will conduct a comparative analysis of the performance of the balanced dataset against various studies. This task has proven to be somewhat challenging due to the limited number of studies that have addressed and implemented the data distribution issue in the N-BaIoT. However, some studies have been identified and will be evaluated in a similar manner as the unbalanced dataset.

5.2.2.1. Studies for the balanced set

Similar to the unbalanced set, the average has been computed across all devices for the proposed approach utilizing the balanced set. The resultant scores are 99.97%, 99.96%, 99.96%, and 99.96%.

In [37], a hybrid deep learning method was proposed. The study has a distribution of 1:2 between benign and malicious data, which is not entirely balanced but still considerable. The results are 99.96%, 99.77%, 99.66%, and 99.66% for the Accuracy, Precision, Recall, and F-1 score. It is likely that the scores would decrease if the distribution ratio had been 1:1.

In [33], various methods are examined. As previously noted, this study addresses the distribution concern and distinguishes between balanced and unbalanced datasets. However, similar to the unbalanced set, the average must be computed across all devices. Consequently, the resulting scores are 92.55% and 89.06%.

In [12], several methods have been employed. Only the Danmini doorbell device is utilized for training and testing. Subsequently, the benign data and 6000 instances of each attack type were extracted, resulting in a nearly balanced 1:1 distribution. To classify the data, a range of baseline and ensemble techniques were employed. The highest-performing method was selected for comparison, yielding scores of 90.87%, 85.67%, and 87.83%.

5.2.2.2. Summary of the comparison for the balanced set

Table 5.6: Comparison with existing studies for the balanced set

Reference	Year	Model	Accuracy(%)	Precision(%)	Recall(%)	F1-score(%)
Proposed	2023	LR-RFC-Grad	99.97	99.96	99.96	99.96
[12]	2022	KNN-GNB-SVC	90.87	85.67	-	87.83
[37]	2021	Hybrid DL	99.96	99.77	99.66	99.66
[33]	2018	SVM,Iso,LR	92.55	89.06	-	-

In Table 5.6 an identical table is computed as was done for the unbalanced set, but this time for the balanced set. While the selected studies differ, the columns serve similar functions as in the preceding section.

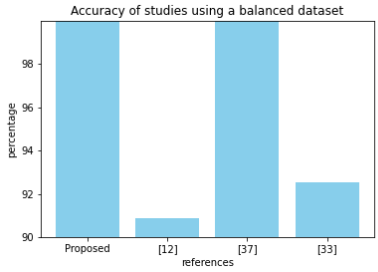

Figure 5.5: Accuracy scores of studies with the balanced dataset

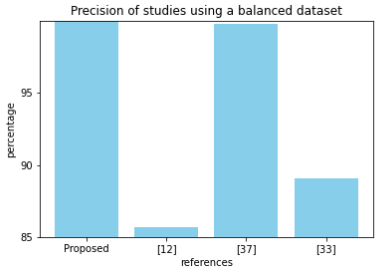

Figure 5.6: Precision scores of studies with the balanced dataset

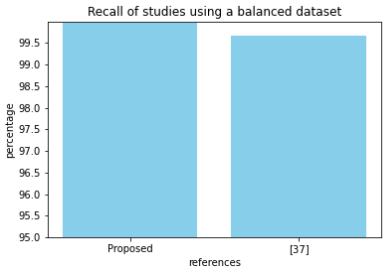

Figure 5.7: Recall scores of studies with the unbalanced dataset

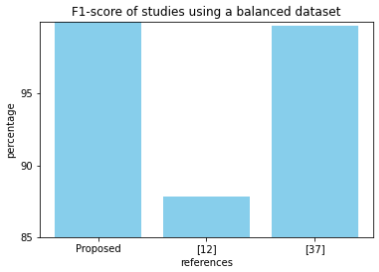

Figure 5.8: F-1 scores of studies with the unbalanced dataset

In Figure 5.5, Figure 5.6, Figure 5.7, and Figure 5.8 the histograms are displayed. As observed, there is a significant improvement in the performance utilizing the balanced dataset. It is worth noting that the margin of improvement is slightly reduced in one study, which may be attributed to a ratio that is not precisely 1:1. Nevertheless, with the limited number of features and scores considered, the improvement is satisfactory.

5.2.3. Computational time

This section will present a comparative analysis of the computational time of the proposed approach. It has been observed that a significant number of studies fail to provide adequate information on computational time, often utilizing limited data for both training and testing purposes, leading to shorter computational time. Consequently, the number of studies available for comparison is limited.

5.2.3.1. Studies for the computational time

For the proposed approach, the computational time of the Danmini doorbell device utilizing the unbalanced set was used for comparison. The resulting time was 99 seconds. The unbalanced set of the Danmini doorbell consists of about 1 million samples, which should be adequate for comparison with other studies.

In [34] the computational time of several deep learning methods is calculated. As aforementioned, the study utilizes five batches of data from all the devices. These batches contain about 200-300 thousand samples for training. It is important to note that this represents only a third of the Danmini doorbell device and should be considered when making comparisons. In this paper, the best-performing method was considered, which has a computation time of 180 seconds.

In [23] several baseline methods were employed. In this study, the training and test time for the Danmini doorbell device was computed. The best-performing model, which took 1550 seconds, was taken into account for comparison.

Table 5.7: Comparison with existing studies for the computational time

Reference	Year	Model	Computational time(s)
Proposed	2023	LR-RFC-Grad	99
[34]	2022	CDL	180
[23]	2020	RFC	1550

5.2.3.2. Summary of the comparison for the computational time

In Table 5.7 the comparison of computational time is displayed. Specifically, the computational time of the Danmini doorbell was taken into account, given its relevance to one of the studies and its proximity to the other.

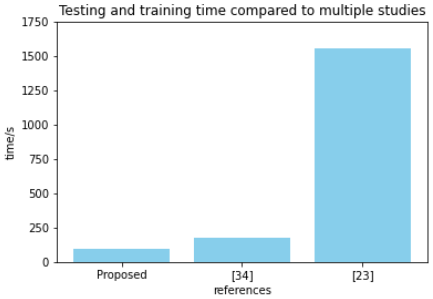

Figure 5.9: Computational time between multiple studies

In Figure 5.9 the histogram is displayed. It is evident that the model outperforms the methods by a large margin even when approximately 30% of the data is employed for the training and testing. This can be attributed to both the classifiers and the features selected for this approach. The main factor is likely the number of features selected as previously discussed. Most of the studies employ about three or even ten times as many features, resulting in a significant increase in computational time.

6

Discussion and Future work

This section presents a discussion of the evaluation results. The initial focus is on the interpretation of the findings, followed by an examination of the limitations of the study. Additionally, potential future research is explored.

6.1. Interpretation

It is evident from chapter 5 that the proposed model outperforms several studies on both the unbalanced dataset and the balanced dataset, while also exhibiting faster computational time. In fact, the proposed model's performance on the balanced dataset is impressive when compared to studies that utilize an unbalanced set. This can be attributed to the extensive feature analysis, which has enabled more effective optimization. It is worth noting that no prior studies have achieved comparable performance using only three features. As noted in chapter 2, utilizing fewer features reduces computation time and resource requirement for the detection model, which is reflected in the results obtained. The model is flexible and the individual classifiers can be adjusted accordingly. No hyperparameter search was conducted to avoid specifying the model to the dataset instances, which facilitates easier deployment of the proposed model in real-time scenarios. However, the performance of the model in real-time instances remains uncertain.

6.2. Limitations

In addition to the contributions of this study, it is important to acknowledge and address some limitations. As with any research, there is always room for improvement. The following limitations have been identified:

- **Multi-classification**. Given that the dataset encompasses ten distinct attacks, it might be advantageous to consider multi-classification as opposed to binary classification. Multi-classification offers superior identification of specific attacks, which can prove valuable in deploying the model within network environments where these attacks are prevalent. However, it is important to note that this approach might result in increased computation time, particularly when dealing with botnets that execute a broad range of attacks. While some studies have explored this approach [39]. The number of studies remains limited compared to the binary research, thus the binary classification was chosen for additional comparison.

- **Real-time**. Another issue addresses all studies that only rely on the dataset to determine the performance of the proposed works. The deployment of models in real-world scenarios introduces several factors that can affect their performance. It is challenging to draw conclusions regarding the performance of the models without specific case studies conducted.

6.3. Future work

As previously noted, there are two significant limitations that can be addressed in future research. Multi-classification is rather trivial to achieve. The labels used in the dataset must be modified to include a label for each attack type, rather than solely malicious and benign labels. The challenge here is to develop a model that is capable of detecting a specific attack. It may be advisable to employ a distinct model for each attack type. This enables the models to detect the attacks with greater specificity. This, in turn, would improve the real-time detection of botnet attacks, as it would allow for the deployment of a specific model in the network location where the attack type is most likely to occur.

A significant challenge is applying the model in real-time scenarios. This is difficult to accomplish as it requires simulating the attack in a real-time institution. Ideally, this can be achieved through case studies involving an institution such as a university or company that is willing to participate. It will require an individual who has expertise in doing these types of botnet attacks, such as ethical hackers, and a sufficient number of IoT devices that can serve as vulnerable devices and botnets. Subsequently, the model must be deployed in a location on the network where it can monitor the packages. For each package, the required features must be calculated and then detected by the model. This again underscores the importance of feature optimization. These are some limitations and future improvements to mention and there may be others.

7

Conclusion

This thesis proposed a hybrid ensemble model for detecting botnet attacks along with improved feature analysis and optimization to overcome the limitations identified in the literature. Given the increasing potency of botnet attacks and their capability to exploit newer vulnerabilities and impact a greater number of devices with each passing year, it is imperative to enhance the current state of research in this area.

The literature review delved into the development and utilization of bots through an analysis of the lifecycle and overall architecture of botnet attacks. Specific botnet attacks were also examined to shed light on the need for a dataset that is current and encompasses the various stages of a botnet attack. The N-BaIoT dataset met these requirements. However, the studies conducted using the dataset were subject to limitations, such as uneven distribution, insufficient feature analysis, and feature optimization.

All these limitations have been addressed and improved in this study. Through feature analysis and optimization, it was determined that only three features were necessary, which is a significant improvement in efficiency compared to existing studies that utilize three to ten times more features. This was deployed in the hybrid ensemble model, which yielded improved results for both the performance scores and computational time compared to the literature.

The model was applied to both the unbalanced and balanced sets, yielding excellent results. The distinct ratio in these datasets facilitates convenient comparison to the studies, as it is the only factor that influences the results that should not be exclusive to a study. This is in contrast to the feature selection and detection model employed in the studies, which is subjective. The findings revealed that even the proposed model utilizing the balanced set outperformed studies that employed an unbalanced set.

With this, the study has successfully achieved its contributions. Future works may involve expanding the study to encompass multi-classification for each attack type, rather than binary classification. On top of that, the model should be employed in several case studies utilizing a real-world scenario.

References

[1] "A collaborative approach to early detection of IoT Botnet". In: *Computers Electrical Engineering* 97 (2022), p. 107525. ISSN: 0045-7906. DOI: https://doi.org/10.1016/j.compeleceng.2021.107525. URL: https://www.sciencedirect.com/science/article/pii/S0045790621004717.

[2] Fereshteh Abbasi, Marjan Naderan, and Seyed Enayatallah Alavi. "Anomaly detection in Internet of Things using feature selection and classification based on Logistic Regression and Artificial Neural Network on N-BaIoT dataset". In: *2021 5th International Conference on Internet of Things and Applications (IoT)*. 2021, pp. 1–7. DOI: 10.1109/IoT52625.2021.9469605.

[3] Aurélien Agniel, David Arnold, and Jafar Saniie. "Image Processing for Detecting Botnet Attacks: A Novel Approach for Flexibility and Scalability". In: *2022 IEEE International Conference and Expo on Real Time Communications at IIT (RTC)*. 2022, pp. 8–12. DOI: 10.1109/RTC56148.2022.9945055.

[4] Ihsan Ali et al. "Systematic Literature Review on IoT-Based Botnet Attack". In: *IEEE Access* 8 (2020), pp. 212220–212232. DOI: 10.1109/ACCESS.2020.3039985.

[5] Hatem A. Almazarqi et al. "Profiling IoT Botnet Activity in the Wild". In: *2021 IEEE Global Communications Conference (GLOBECOM)*. 2021, pp. 1–6. DOI: 10.1109/GLOBECOM46510.2021.9686012.

[6] Antonia Raiane S. Araujo Cruz, Rafael L. Gomes, and Marcial P. Fernandez. "An Intelligent Mechanism to Detect Cyberattacks of Mirai Botnet in IoT Networks". In: *2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. 2021, pp. 236–243. DOI: 10.1109/DCOSS52077.2021.00047.

[7] Manikandan Ayyanar et al. "Predicting the Cardiac Diseases using SelectKBest Method Equipped Light Gradient Boosting Machine". In: *2022 6th International Conference on Trends in Electronics and Informatics (ICOEI)*. 2022, pp. 117–122. DOI: 10.1109/ICOEI53556.2022.9777224.

[8] Madelyn Bacon. *Okiru malware puts billions of connected devices at risk: TechTarget*. Jan. 2018. URL: https://www.techtarget.com/searchsecurity/news/252433491/Okiru-malware-puts-billions-of-connected-devices-at-risk.

[9] Pamela Beltrán-García et al. "IoT Botnets". In: *Telematics and Computing*. Ed. by Miguel Felix Mata-Rivera, Roberto Zagal-Flores, and Cristian Barría-Huidobro. Cham: Springer International Publishing, 2019, pp. 247–257.

[10] Marta Catillo, Antonio Pecchia, and Umberto Villano. "Botnet Detection in the Internet of Things through All-in-One Deep Autoencoding". In: New York, NY, USA: Association for Computing Machinery, 2022. ISBN: 9781450396707. DOI: 10.1145/3538969.3544460. URL: https://doi.org/10.1145/3538969.3544460.

[11] Smita Dange. "IoT Botnet: The Largest Threat to the IoT Network". In: Sept. 2019. DOI: 10.1007/978-981-15-0132-6.

[12] Priscilla Kyei Danso et al. "Ensemble-based Intrusion Detection for Internet of Things Devices". In: *2022 IEEE 19th International Conference on Smart Communities: Improving Quality of Life Using ICT, IoT and AI (HONET)*. 2022, pp. 034–039. DOI: 10.1109/HONET56683.2022.10019140.

[13] Gonzalo De La Torre Parra et al. “Detecting Internet of Things attacks using distributed deep learning”. In: *Journal of Network and Computer Applications* 163 (2020), p. 102662. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2020.102662. URL: https://www.sciencedirect.com/science/article/pii/S1084804520301363.

[14] Mirabelle Dib et al. “A Multi-Dimensional Deep Learning Framework for IoT Malware Classification and Family Attribution”. In: *IEEE Transactions on Network and Service Management* 18.2 (2021), pp. 1165–1177. DOI: 10.1109/TNSM.2021.3075315.

[15] Bardia Esmaeili et al. “A GNN-Based Adversarial Internet of Things Malware Detection Framework for Critical Infrastructure: Studying Gafgyt, Mirai and Tsunami Campaigns”. In: *IEEE Internet of Things Journal* (2023), pp. 1–1. DOI: 10.1109/JIOT.2023.3298663.

[16] Ahmad Firdaus et al. “The Summer Heat of Cryptojacking Season: Detecting Cryptojacking using Heatmap and Fuzzy”. In: *2022 International Conference on Cyber Resilience (ICCR)*. 2022, pp. 1–5. DOI: 10.1109/ICCR56254.2022.9995891.

[17] Josh Fruhlinger. *The Mirai botnet explained: How IOT devices almost brought down the internet*. Mar. 2018. URL: https://www.csoonline.com/article/3258748/the-mirai-botnet-explained-how-teen-scammers-and-cctv-cameras-almost-brought-down-the-internet.html.

[18] Isha Gandhi and Mrinal Pandey. “Hybrid Ensemble of classifiers using voting”. In: *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*. 2015, pp. 399–404. DOI: 10.1109/ICGCIoT.2015.7380496.

[19] Harm J. Griffioen and Christian Doerr. “Examining Mirai’s Battle over the Internet of Things”. In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security* (2020).

[20] Mikhail Gromov, David Arnold, and Jafar Saniie. “Edge Computing for Real Time Botnet Propagation Detection”. In: *2022 IEEE International Conference and Expo on Real Time Communications at IIT (RTC)*. 2022, pp. 13–16. DOI: 10.1109/RTC56148.2022.9945060.

[21] Tooba Hasan et al. “Securing Industrial Internet of Things Against Botnet Attacks Using Hybrid Deep Learning Approach”. In: *IEEE Transactions on Network Science and Engineering* (2022), pp. 1–1. DOI: 10.1109/TNSE.2022.3168533.

[22] Alvi Jawad et al. “A Formal Analysis of the Efficacy of Rebooting as a Countermeasure Against IoT Botnets”. In: *ICC 2022 - IEEE International Conference on Communications*. 2022, pp. 2206–2211. DOI: 10.1109/ICC45855.2022.9838865.

[23] Shreehar Joshi and Eman Abdelfattah. “Efficiency of Different Machine Learning Algorithms on the Multivariate Classification of IoT Botnet Attacks”. In: *2020 11th IEEE Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*. 2020, pp. 0517–0521. DOI: 10.1109/UEMCON51285.2020.9298095.

[24] Tran Viet Khoa et al. “Collaborative Learning Model for Cyberattack Detection Systems in IoT Industry 4.0”. In: *2020 IEEE Wireless Communications and Networking Conference (WCNC)*. 2020, pp. 1–6. DOI: 10.1109/WCNC45663.2020.9120761.

[25] Brian Krebs. *Krebsonsecurity hit by huge new IOT botnet “Meris”*. Sept. 2021. URL: https://krebsonsecurity.com/2021/09/krebsonsecurity-hit-by-huge-new-iot-botnet-meris/.

[26] Rajesh Kumar Yadav and Karamveer Karamveer. “A Survey on IOT Botnets and their Detection Approaches”. In: *2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*. 2022, pp. 1901–1906. DOI: 10.1109/ICAC3N56670.2022.10074482.

[27] Satsuki Kumatani et al. “Time-Varying Data Visualization Using Clustered Heatmap and Dual Scatterplots”. In: *2016 20th International Conference Information Visualisation (IV)*. 2016, pp. 63–68. DOI: 10.1109/IV.2016.50.

[28] Badr Lahasan and Hussein Samma. “Optimized Deep Autoencoder Model for Internet of Things Intruder Detection”. In: *IEEE Access* 10 (2022), pp. 8434–8448. DOI: 10.1109/ACCESS.2022.3144208.

[29] Ravie Lakshmanan. *New mirai botnet variant “v3g4” exploiting 13 flaws to target linux and IOT devices*. Feb. 2023. URL: https://thehackernews.com/2023/02/new-mirai-botnet-variant-v3g4.html.

[30] Moemedi Lefoane et al. “Machine Learning for Botnet Detection: An Optimized Feature Selection Approach”. In: New York, NY, USA: Association for Computing Machinery, 2022. ISBN: 9781450387347. DOI: 10.1145/3508072.3508102. URL: https://doi.org/10.1145/3508072.3508102.

[31] Artur Marzano et al. “The Evolution of Bashlite and Mirai IoT Botnets”. In: *2018 IEEE Symposium on Computers and Communications (ISCC)*. 2018, pp. 00813–00818. DOI: 10.1109/ISCC.2018.8538636.

[32] Yair Meidan et al. “N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders”. In: *IEEE Pervasive Computing* 17.3 (2018), pp. 12–22. DOI: 10.1109/MPRV.2018.03367731.

[33] Sven Nõmm and Hayretdin Bahşi. “Unsupervised Anomaly Based Botnet Detection in IoT Networks”. In: *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. 2018, pp. 1048–1053. DOI: 10.1109/ICMLA.2018.00171.

[34] Segun I. Popoola et al. “Federated Deep Learning for Zero-Day Botnet Attack Detection in IoT-Edge Devices”. In: *IEEE Internet of Things Journal* 9.5 (2022), pp. 3930–3944. DOI: 10.1109/JIOT.2021.3100755.

[35] Segun I. Popoola et al. “Hybrid Deep Learning for Botnet Attack Detection in the Internet-of-Things Networks”. In: *IEEE Internet of Things Journal* 8.6 (2021), pp. 4944–4956. DOI: 10.1109/JIOT.2020.3034156.

[36] Asma Qureshi et al. “Relationship between kernel density function estimates of gait time series and clinical data”. In: *2017 IEEE EMBS International Conference on Biomedical Health Informatics (BHI)*. 2017, pp. 329–332. DOI: 10.1109/BHI.2017.7897272.

[37] Sirajuddin Qureshi et al. “A Hybrid DL-Based Detection Mechanism for Cyber Threats in Secure Networks”. In: *IEEE Access* 9 (2021), pp. 73938–73947. DOI: 10.1109/ACCESS.2021.3081069.

[38] Valerian Rey et al. “Federated learning for malware detection in IoT devices”. In: *Computer Networks* 204 (Feb. 2022), p. 108693. DOI: 10.1016/j.comnet.2021.108693. URL: https://doi.org/10.1016%2Fj.comnet.2021.108693.

[39] Kumar Saurabh et al. “GANIBOT: A Network Flow Based Semi Supervised Generative Adversarial Networks Model for IoT Botnets Detection”. In: *2022 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*. 2022, pp. 1–5. DOI: 10.1109/COINS54846.2022.9854947.

[40] Norisato Suga et al. “Estimation of Probability Density Function Using Multi-bandwidth Kernel Density Estimation for Throughput”. In: *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. 2020, pp. 171–176. DOI: 10.1109/ICAIIC48513.2020.9065033.

[41] Fatma Taher et al. “Reliable Machine Learning Model for IIoT Botnet Detection”. In: *IEEE Access* (2023), pp. 1–1. DOI: 10.1109/ACCESS.2023.3253432.

[42] Simon Nam Thanh Vu et al. “A Survey on Botnets: Incentives, Evolution, Detection and Current Trends”. In: *Future Internet* 13.8 (2021). ISSN: 1999-5903. URL: https://www.mdpi.com/1999-5903/13/8/198.

[43] M. D. Tislenko, A. V. Gaidel, and A. V. Kupriyanov. “Comparison of feature selection algorithms for Data classification problems”. In: *2022 VIII International Conference on Information Technology and Nanotechnology (ITNT)*. 2022, pp. 1–5. DOI: 10.1109/ITNT55410.2022.9848765.

[44] Olivier Tsemogne et al. “Game-Theoretic Modeling of Cyber Deception Against Epidemic Botnets in Internet of Things”. In: *IEEE Internet of Things Journal* 9.4 (2022), pp. 2678–2687. DOI: 10.1109/JIOT.2021.3081751.

[45] Lionel Sujay Vailshery. *IOT connected devices worldwide 2019-2030*. Nov. 2022. URL: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/.

[46] M. Vijayakumar and Shiny Angel T.S. “A Survey on IoT Security: Security Threads and Analysis of Botnet Attacks Over IoT and Avoidance”. In: Jan. 2022, pp. 141–154. ISBN: 978-981-16-8663-4. DOI: 10.1007/978-981-16-8664-1_13.

[47] Jianyu Wang and Jianli Pan. “MBM-IoT: Intelligent Multi-Baseline Modeling of Heterogeneous Device Behaviors against IoT Botnet”. In: *2022 IEEE 19th Annual Consumer Communications Networking Conference (CCNC)*. 2022, pp. 711–712. DOI: 10.1109/CCNC49033.2022.9700572.

[48] Majda Wazzan et al. “Internet of Things Botnet Detection Approaches: Analysis and Recommendations for Future Research”. In: *Applied Sciences* (2021).

[49] Yalian Wu, Xieen He, and Xingnian Chen. “IoT-Botnet Traffic Detection Based on Deep Forest”. In: *2022 IEEE 22nd International Conference on Communication Technology (ICCT)*. 2022, pp. 1388–1393. DOI: 10.1109/ICCT56141.2022.10072774.

[50] Junyang Zhao et al. “Experiments with Feature-Prior Hybrid Ensemble Method for Classification”. In: *2014 Tenth International Conference on Computational Intelligence and Security*. 2014, pp. 223–227. DOI: 10.1109/CIS.2014.108.

A

Figures and plots

This appendix contains the remaining figures and plots that have been mentioned but not shown in the thesis report to avoid overload.

A.1. Distribution of the data in the devices

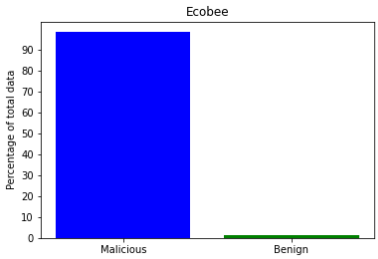

Figure A.1: Distribution of the data in the Ecobee device

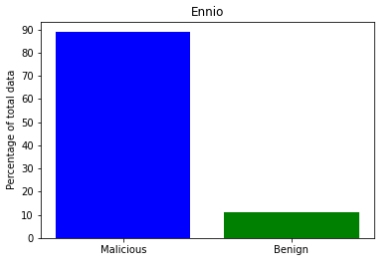

Figure A.2: Distribution of the data in the Ennio device

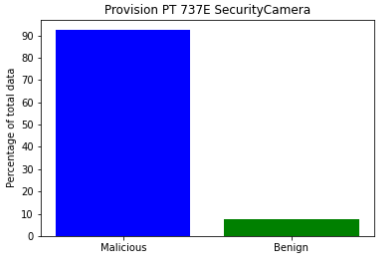

Figure A.3: Distribution of the data in the Provision PT 737E Security Camera device

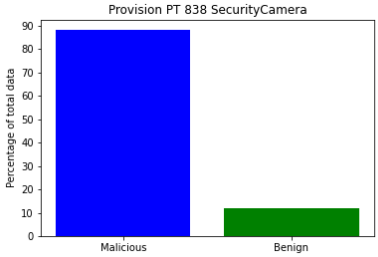

Figure A.4: Distribution of the data in the Provision PT 838 Security Camera device

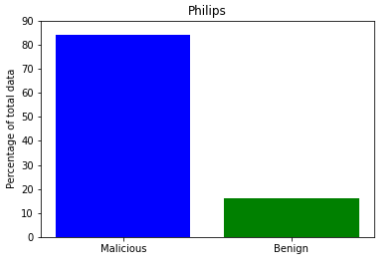

Figure A.5: Distribution of the data in the Philips device

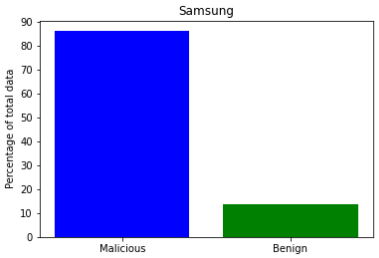

Figure A.6: Distribution of the data in the Samsung device

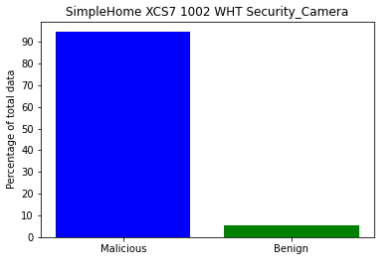

Figure A.7: Distribution of the data in the SimpleHome XCS7 1002 WHT Security Camera device

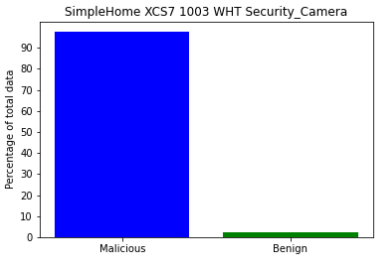

Figure A.8: Distribution of the data in the SimpleHome XCS7 1003 WHT Security Camera device

A.2. All kernel density plots of the devices

Figure A.9: 1/4 of kernel density estimates of the Danmini doorbell

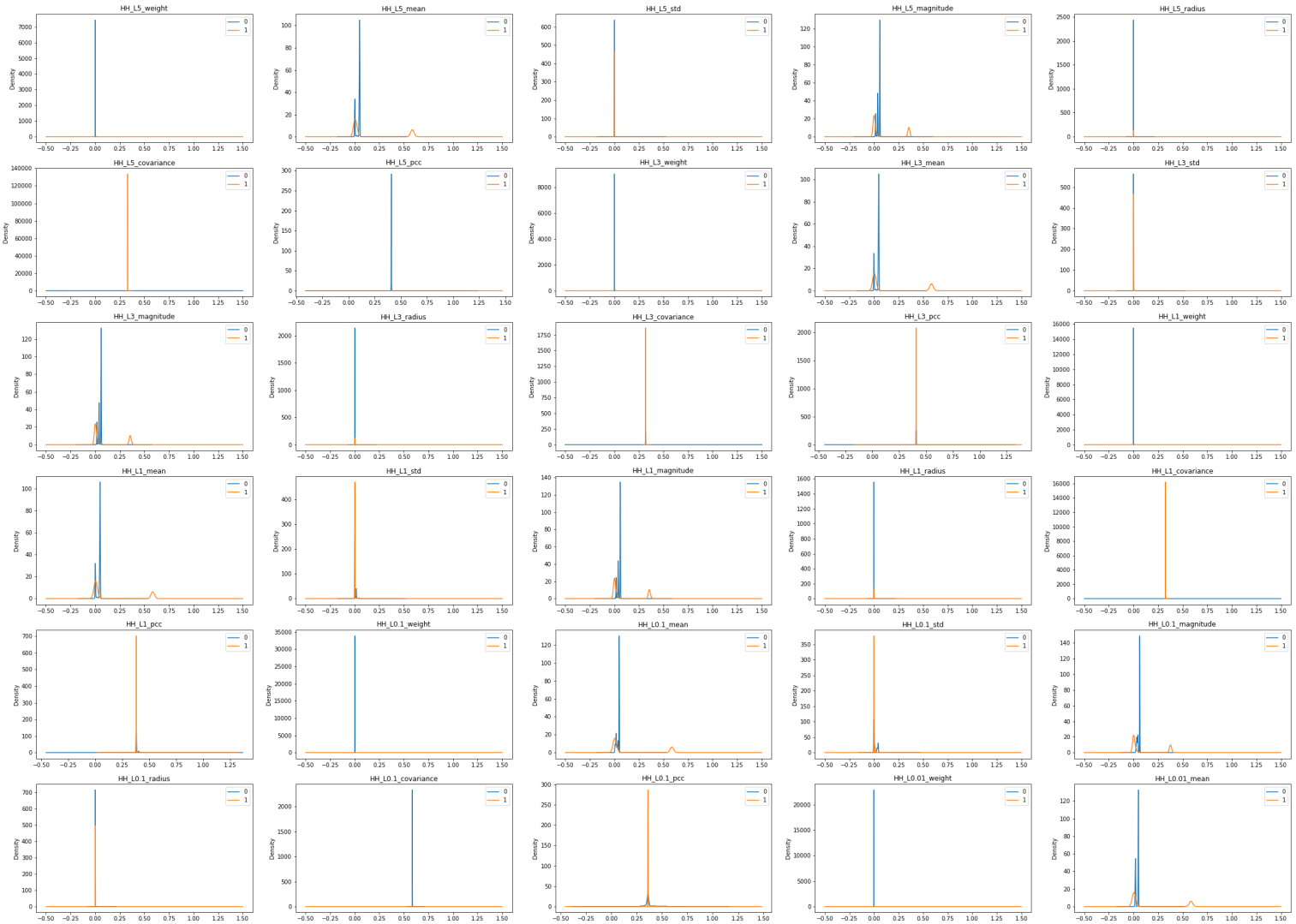

Figure A.10: 2/4 of kernel density estimates of the Danmini doorbell

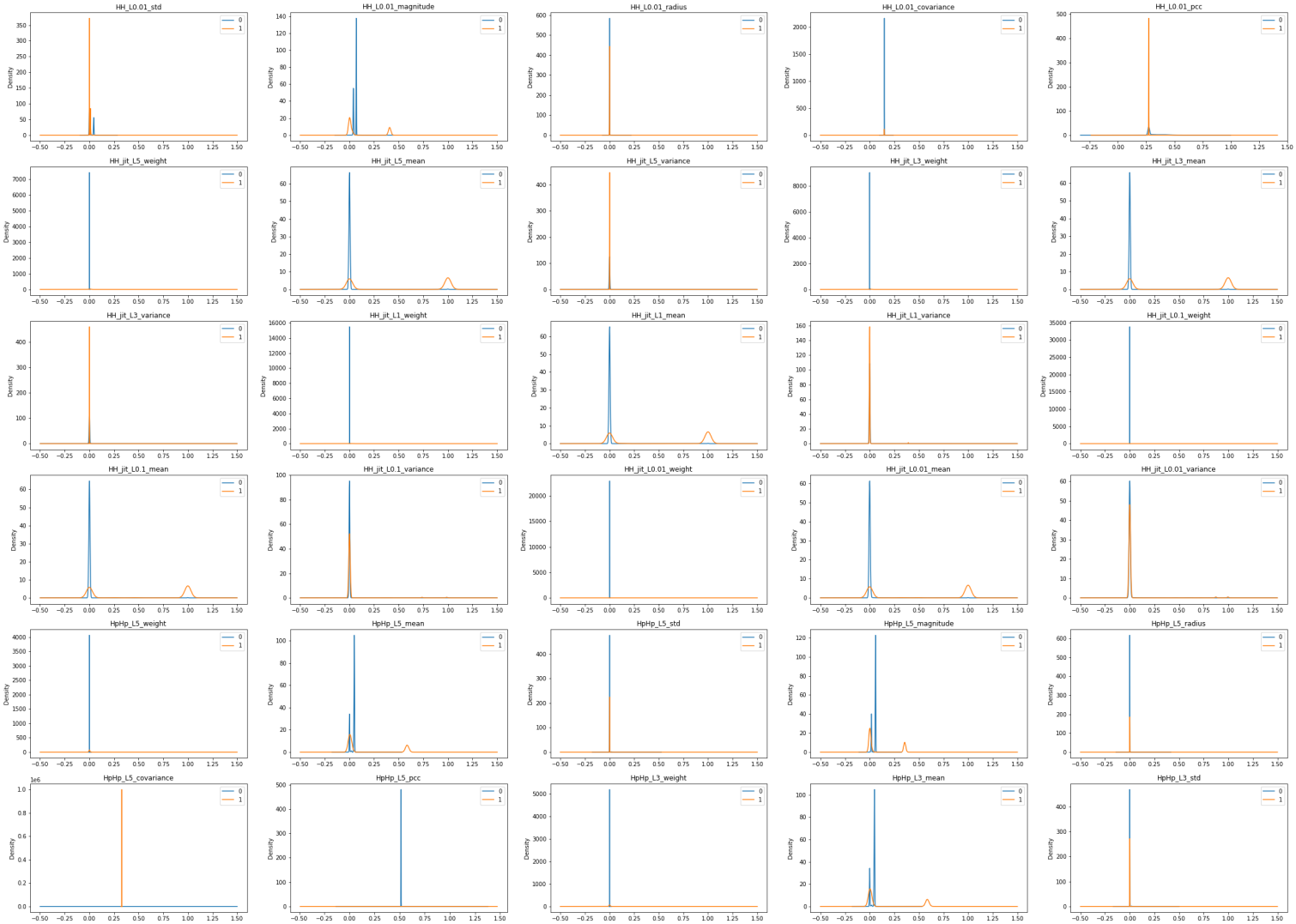

Figure A.11: 3/4 of kernel density estimates of the Danmini doorbell

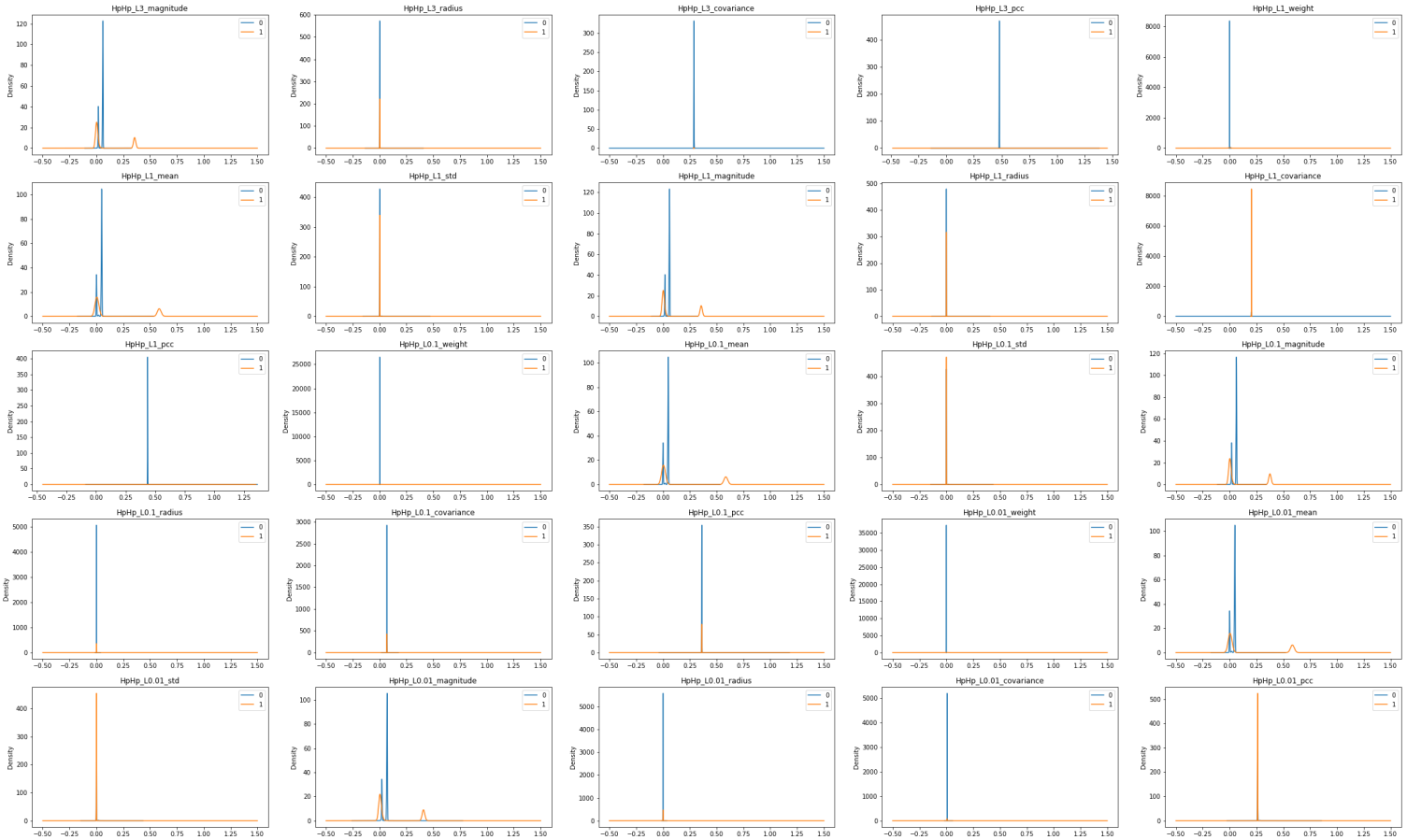

Figure A.12: 4/4 of kernel density estimates of the Danmini doorbell

The plots for the remaining devices can be found in the corresponding Github repository