Reproducing and Extending "Identifying IoT Devices and Events Based on Packet Length from Encrypted Traffic"

Mahira Ali*
Zohar Cochavi*
Murtuza Ali*
M.Ali-10@student.tudelft.nl
Z.Cochavi@student.tudelft.nl
M.A.Mohammed-1@student.tudelft.nl
Delft University of Technology
Delft, The Netherlands

Roland Kromes
Delft University of Technology
Delft, The Netherlands

ABSTRACT

This paper presents a reproduction and extension of the proposed method to identify IoT devices based on packet length analysis of encrypted traffic in Pinheiro et al.'s paper, "Identifying IoT Devices and Events Based on Packet Length from Encrypted Traffic" [15]. As the number of IoT devices continues to grow, it becomes increasingly important to monitor their behavior and detect security threats. This paper provides open-source code [1] and a simulation environment, allowing easy reproduction of the proposed solution. Furthermore, the paper investigates the influence of padding on the classifiers' performance. The findings indicate that while our implementation of the device classifier falls short in performance compared to the reference, the IoT classifier exhibits a comparable level of effectiveness. This is taking into consideration that not much time was spent on hyper-parameter tuning or other optimizations. Although the addition of padding showed predictable results for the device classifier, namely, a significant drop in performance, the IoT classifier showed surprising resilience to padding. We also tested the performance of the classifier with artificially added noise to simulate other activity on the network and noticed a significant drop in performance as well.

ACM Reference Format:
Mahira Ali, Zohar Cochavi, Murtuza Ali, and Roland Kromes. 2023. Reproducing and Extending "Identifying IoT Devices and Events Based on Packet Length from Encrypted Traffic". In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

IoT (Internet of Things) security is a critical aspect of the IoT ecosystem that concerns the protection of IoT devices, networks, and data from unauthorized access, tampering, and attacks [5]. The IoT security landscape is complex, as it involves a wide range of devices, protocols, and data types that operate in different environments, such as homes, industries, and cities [12]. Some key challenges that IoT security faces include device vulnerabilities, network security, interoperability and data privacy [9].

The paper "Identifying IoT Devices and Events Based on Packet Length from Encrypted Traffic" [15] which we are reproducing in this paper, specifically addresses the challenge of data privacy. Here, the authors proposed a classifier to identify IoT devices and their activities based on packet length analysis of encrypted traffic. The aim of the paper is to address the challenges of securing IoT devices by proposing a novel approach to identify IoT devices and their activities in encrypted traffic. The authors argue that traditional security approaches are insufficient for IoT devices due to their resource constraints and the complexity of the IoT ecosystem. The proposed method is based on the observation that IoT devices generate packets of different lengths depending on their activities, such as sending sensor data, receiving commands, or executing firmware updates. By analyzing packet length patterns, the method can identify the type of IoT device and its activity, even when the traffic is encrypted.

The primary goal of this paper was to reproduce the method proposed in the paper "Identifying IoT devices and events based on packet length from encrypted traffic" by Pinheiro, Bezerra et al [15]. To achieve this goal, we re-created their approach to identify IoT devices based on packet length analysis of encrypted traffic. Specifically, we analyzed the packet length patterns of encrypted traffic generated by various IoT devices and used machine learning algorithms to classify the type of device. After re-creating their method, we evaluated our results and compared them to the claimed results in the original paper.

In addition to reproducing the proposed solution, this paper presents several contributions that advance the understanding and practical implementation of IoT security measures. Firstly, it introduces open-source code[1], and a simulation environment for implementing the proposed solution, which is a significant contribution that enables other researchers and practitioners in the field to easily reproduce and build upon the proposed solution. Additionally, the paper provides a detailed explanation of the machine learning models used in the classification process, which was not

*All authors contributed equally to this research.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

[1] https://github.com/cochaviz/iot_classifier/tree/v0.1

covered in the original solution. Finally, the influence of padding on classification performance is investigated. The used padding scheme is based on the strategy proposed by Pinheiro et al. [16]. Taken together, these contributions demonstrate the originality of this paper.

The paper is structured as follows. Firstly, section 2 provides information about the reproduced paper, outlining the methods and results of the original study. Here, we also discuss the motivation for reproducing the study and the research questions that guided the investigation. This is followed by section 3 where the dataset used in the study is described, including the data sources, the types of IoT devices included in the dataset, and the characteristics of the traffic. Then, section 4 describes the methodology used in the research, such as the steps taken to parse and convert the Packet CAPture (PCAP) files into Comma-Separated Values (CSV) format, the machine learning algorithms used for classification, and the simulation experiment conducted to investigate the impact of packet padding on classification accuracy. Next, we present the results of the study in section 5, including the accuracy of the proposed method for identifying IoT devices based on packet length analysis of encrypted traffic, as well as the findings from the simulation experiment. Then, section 6 provides a discussion of the results. Here, we analyse the limitations of the proposed method and specify the implications of the findings for the field of IoT security. This section also discusses the potential for future research in this area. Lastly, section 7 concludes the paper with a summary of the main findings and contributions of the research.

2 BACKGROUND

The advent of the IoT has brought along with it several devices that are adopted in households, hospitals and other industries. Identifying and classifying these devices can be extremely useful for administrators to understand various characteristics of the network, so that they can address any vulnerabilities that may be present and ensure that all devices are maintained with the proper updates, Moreover, being able to keep check all IoT devices in an environment can be used to identify if a device has been hacked or is being used for malicious purposes. Being able to classify these devices could also allow for certain QoS or Quality of Service policies to be applied to the communication, which could help reduce response times for devices that are either important such as medical equipment or so that there is an enhanced user interaction experience with commercial products. This has led to extensive research in this area with methods ranging from deep packet analysis, resolving communication addresses, and using packet flows or inter-arrival times, however, some of these methods could be seen as invading the privacy of the users and can also be very resource intensive. For the purpose of the paper, the authors aim at classifying the devices using non-intrusive and privacy-preserving methods by examining packet length and its statistical characteristics. This classifier should also be lightweight enough to provide a "real-time" classification of devices and device events compared to other methods which are more resource intensive, considering that these measures are usually to be deployed on a router or gateway which has fairly restricted resources.

Other researchers feel that although this classification is useful in several settings to maintain an inventory of devices and to monitor if they are operating as expected, these methods can also be used to infer user behaviors and characteristics which can be used for malicious purposes, for example identifying devices and device events would allow someone to understand when someone is at home or is interacting with the devices. To combat this, the authors of the paper "Adaptive Packet Padding Approach for Smart Home Networks: A Trade-off between Privacy and Performance" [16] propose a padding strategy to mitigate the classification measures in the aforementioned paper to ensure that complete privacy is maintained with regard to IoT device use. However, padding also incurs an overhead, thus a balance between the obfuscation and added overhead is considered an important problem and is addressed by the authors.

2.0.1 Paper Overview. In order to determine the statistical measures used, the importance of the features, namely the mean, median and mode was calculated. It was observed that the mean is the most important measure, so the measures picked for classification were the mean packet size, the standard deviation and the number of bytes, where each of these values is computed for all packets that arrive within a one-second window. Five classifiers were implemented, namely K-Nearest Neighbours, Decision Tree, Random Forest, and also majority voting of the aforementioned classifiers. The imbalanced data was also resampled using techniques such as SMOTE, and Random Undersampling or RUS. Experiments were also performed by including or omitting certain devices such as cameras to understand their impact on classifier performance. Two 10-fold cross-validation procedures were used to determine the performance, one method used random partitioning, while the second method utilized chronological partitioning and a sliding window. The results of these experiments will be discussed and compared with our own results in section 5.

In the second paper "Adaptive Packet Padding Approach for Smart Home Networks: A Trade-off between Privacy and Performance" [16], the authors aim at creating a privacy-enhancing solution by obfuscating packet details to prevent the classification of users' IoT devices and activity. An environment utilizing SDNs is created to simulate a home network with a chain of routers. An SFlowRT service runs on the first switch to monitor the network utilization, and adjusts the padding strategy accordingly, by reducing the padding amount in case the network cannot handle the load, or increasing it if there is scope to do so. The authors also train the models mentioned in the former paper and compare and contrast the performance of the models with different padding strategies to the performance of the models without any obfuscation.

As there was no source code provided with the aforementioned papers and details on the implementation were very scarce, we tried to create an open-source reproduction[1] of the models presented and to further develop the findings of the authors by running experiments on what we felt were environments that were more representative of a real-life setting and presenting our findings.

3 IEEE TMC 2018 DATASET

The proposed solution in the reproduced paper relies on the data made available by Sivanathan et al. in their research publication titled "Experimental evaluation of cybersecurity threats to the smart-home" [18] [2]. This IoT network traffic dataset, known as the IEEE TMC 2018 dataset, consists of 20 PCAP files captured from a diverse range of smart home environments, including 21 IoT devices and 7 non-IoT devices. The dataset was collected using packet-capturing tools and contains network traffic data from various IoT devices, such as smart plugs, thermostats, cameras, and other smart home devices. Additionally, a text file mapping each Media Access Control (MAC) address to a corresponding device name was provided by the authors.

4 METHODOLOGY

This section outlines the methodology used to reproduce the execution flow of the proposed solution. Figure 1 depicts the execution flow, with green solid boxes representing the steps that were reproduced in this paper. First, we describe the conversion process from PCAP files to CSV files. Then, we discuss the five stages of classification. This includes both the pre-processing stage, where we explain the calculation of statistics, and the three-stage process for traffic classification. The latter is composed of three tasks, with the first two being to distinguish IoT devices from non-IoT devices and to identify IoT devices. We have successfully reproduced these two stages. For the first stage, a **binary** classifier was created, while a **multi-class** algorithm was implemented to identify IoT devices. In the following section, we will discuss the creation of both classifiers.

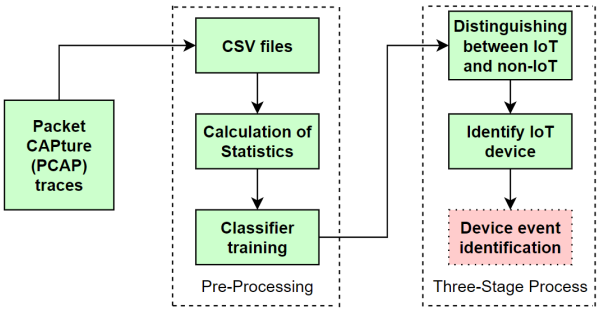

Figure 1: The execution flow of the proposed solution. The green solid boxes represent the steps that were reproduced.

4.1 PCAPs to CSVs

To parse PCAP files of the IEEE TMC 2018 dataset into CSV files, we followed a step-by-step procedure. Firstly, we obtained all the PCAPs and the list of devices using the provided script[3]. Next, we extracted each PCAP file and converted it into a corresponding CSV file. During the conversion process, we extracted certain attributes, including packet_id, timestamp, packet size, Ethernet (eth) source, device name, and IoT indicator.

[2]Accessible through https://iotanalytics.unsw.edu.au/
[3]https://github.com/arunmir/sdn-sim

4.1.1 Dropping Packets. After converting the original PCAP file to CSV, we found that the resulting file contained more packets than the published CSV files [2]. Upon further analysis, we observed that the ICMP(v6) and IGMP packets were dropped in the published file. The original paper mentions that protocols such as TCP, DNS, NTP, and ICMP form signaling traffic when there is no interaction with the devices for a period, and therefore, "these packets have standardized lengths for all devices, which makes these" unsuitable for packet length analysis. However, this statement implies that the TCP, DNS, and NTP packets would also be dropped for the same reason, which is not the case. It is important to note that excluding ICMP(v6) and IGMP packets may impact the accuracy of the results, and further analysis is needed to determine the impact of this exclusion.

In order to reproduce the original CSV file, the protocol was also extracted from the packets in the PCAP files, and packets with the ICMP(v6) and IGMP protocols were dropped. The resulting data was divided into three datasets. The first dataset is the original dataset, which refers to the online available CSV file. The second dataset is the unfiltered dataset, which consists of the CSV files obtained without filtering any packets. Finally, the third dataset is the filtered dataset, which includes the CSV files obtained after filtering out the discarded protocols. In the proposed solution outlined in the original paper, the initial step was to convert the PCAP files into CSV files. This led us to believe that the authors had created their own CSV files, rather than using the original CSV files. To confirm which dataset was used in the paper, we attempted to reproduce the table displaying the mean, median, and mode statistics of packet lengths for the analyzed IoT devices. Table 3 in Appendix A shows the statistical values found by Pinheiro et. al.

After examining the statistics of each dataset, it was observed that none of these matched the results claimed in the original paper's table. Specifically, the statistical results of the original and filtered dataset fully corresponded, while the unfiltered dataset results were more aligned with the claimed results. This observation suggests that the claim of dropping certain protocols may have affected the statistical results, and hence the successive steps in the execution flow. Ideally, we would have liked to perform the execution flow for each dataset to resolve this issue. However, due to time constraints, we decided to execute the successive steps on the unfiltered dataset, which showed the highest level of alignment with the claimed results. The statistical results for each of the three datasets are presented in Table 4, Table 5, Table 6 in Appendix A.

4.1.2 Excluding Camera Devices. In the original paper, the authors claimed that the imbalance in the dataset was mainly caused by camera devices. They proceeded to exclude the camera devices and concluded that the exclusion had limited impact on the performance of the classifier. In order to validate this claim, we created two datasets: the full dataset, including all devices, and the camera-excluded dataset, which excluded all camera devices.

After a thorough investigation, we encountered some ambiguity regarding the Dropcam devices used in the experiment. In the table listing the devices used in the study, a Dropcam device from vendor NEST was included. However, the list of devices contained a Dropcam and a NEST Dropcam, both with different MAC addresses. As the paper mentions a Dropcam from vendor NEST, it is unclear

which of the two devices was used in the analysis. We assume that the inclusion or exclusion of one of the two devices will not have a significant impact on the classifier's performance. Nevertheless, we would like to point out this ambiguity for the sake of completeness. In the full dataset, both Dropcams have been included to ensure totality.

4.2 Classification

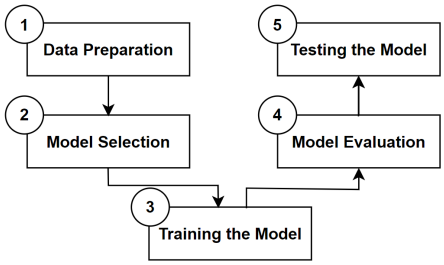

Figure 2: The five stages of classification.

Figure 2 shows the five stages of classification: Data preparation, Model selection, Training the model, Model Evaluation, and Testing the model. To implement the classification, various Python libraries have been used, including Pandas [13] for data manipulation, scikit-learn [14] for machine learning, imbalanced-learn [6] for handling imbalanced datasets, and seaborn [19] for data visualization. These libraries provide a comprehensive set of tools and functions to facilitate the different stages of the classification process, making it easier and more efficient to develop accurate and reliable classifiers.

4.2.1 Data Preparation. This stage can be divided into three steps, namely window slicing, computing statistics, and balancing. Firstly, the data was sliced into windows of one second. The authors argued that windows of one second were chosen to balance the trade-off between capturing enough traffic to make accurate classifications and reducing the amount of data that needed to be processed.

Then, the statistical mean, standard deviation, and the number of bytes transmitted over a one-second window were calculated since based on these values the classification is performed. The main reason why they choose these statistics is that they represent the basic statistical properties of the packet length distribution, which can provide useful information for identifying different IoT devices and events. For illustrative purposes, a partition of the dataset is given in Appendix B.

Lastly, solely for the IoT/non-IoT classifier, the training data was balanced before training to improve the performance of the classifier. The data imbalance was observed because some device classes were underrepresented in the dataset. To balance the data, they used a combination of oversampling, undersampling, and ensemble techniques to balance the data before training the machine learning models. Specifically, the authors used SMOTE [4] (Synthetic Minority Over-sampling Technique) for oversampling the minority class, RandomUnderSampler [5] for undersampling the majority class, and BalanceCascade [6] to ensemble the oversampled and undersampled datasets. By using a combination of oversampling, undersampling, and ensemble techniques, the authors aimed to create a more balanced dataset that could lead to better classification performance, especially for the minority class.

It is important to note that in our implementation, we opted to use no ensemble technique, unlike the original paper which used the BalanceCascade method. The primary reason for this decision was that BalanceCascade was only available in older versions of the imblearn library, and we wanted our code to be built upon the latest version of the libraries. By using the latest versions, we aimed to ensure that our code would remain up-to-date, reliable, and secure and would take advantage of new features and community support [20]. Therefore, we concluded that using no ensemble technique was the most suitable method for our purposes under the time constraints[7].

4.2.2 Model Selection. Here, the following models were chosen: K-Nearest Neighbour (KNN), Decision Tree, Random Forest, and Majority Voting. The decision to include these models was motivated by several factors. Firstly, we wanted to compare our results with the original paper and validate the proposed approach [8]. Secondly, these models are widely used and have shown promising results in classification tasks, making them a popular choice in the literature. Furthermore, each model has unique strengths and weaknesses. KNN is a simple and effective algorithm that can be used for both binary and multiclass classification [10]. Decision trees and random forests are powerful models that can handle complex datasets with high-dimensional features and are known for their interpretability [17]. Majority voting is an ensemble method that combines the predictions of multiple models to improve classification performance [8]. By using multiple models, we can explore different aspects of the data and provide a comprehensive analysis of the performance of different algorithms.

4.2.3 Training the model. This stage involved performing hyperparameter tuning and training each classifier using the selected models. In order to optimize the hyperparameters of our selected models, we employed the GridSearchCV function from the scikit-learn library. This function allows us to search over a range of hyperparameters and find the optimal values for each model. By using GridSearchCV, we were able to search through a large parameter space and find the best set of hyperparameters for each model, which we then used in our final evaluation. The 'best' performing models were selected based on the F1-score. As this score takes into consideration both the precision and the recall, it is a good candidate for evaluating a classifier on an imbalanced dataset [7][9].

[4]Generates synthetic samples to increase the number of instances in the minority class [2].
[5]Randomly removes instances from the majority class to reduce its size [2].
[6]An ensemble technique that combines multiple classifiers, where each classifier is trained on a subset of the data that has been balanced by oversampling and undersampling techniques [2].
[7]It is might have been better to use an alternative ensemble technique as not using an ensemble when one is needed could result in lower accuracy and reliability of the model [11]. However, using it improperly could result in overfitting. As there was not sufficient time to investigate a proper alternative, the decision was made to use none.
[8]The original paper also included the Support Vector Machine (SVM) model. However, training this model was too time-consuming. Therefore, we decided to discard this model.
[9]To most closely resemble the final scenario in which the classifier would be trained, the tuning was done based on balanced datasets. This means that the cross-validation step used a part of the balanced dataset to evaluate the model performance. Ideally,

It is worth noting that the original paper does not mention whether hyperparameter tuning was performed or not. Nonetheless, hyperparameter tuning is a crucial step in machine learning model development as it helps to identify the optimal values for model parameters, which can significantly improve the model's performance [3]. Therefore, the absence of hyperparameter tuning in the original paper can lead to suboptimal model performance and may affect the validity and reliability of the results. It is important to conduct hyperparameter tuning to ensure that the selected models are optimized and perform at their best on the given dataset.

After the hyperparameter tuning process was completed, the selected models were trained on the training data using the optimized hyperparameters. This allowed the models to learn the patterns and relationships between the statistical measures and class labels in the data, and ultimately make accurate predictions on unseen data during the testing phase.

4.2.4 Model Evaluation. In this stage two different approaches were adopted for the device classifier: stratified evaluation with random partitioning and 10-fold cross-validation with chronological partitioning. The decision to include these approaches was based on the original paper's methodology to enable comparison of the results and validate our approach. By adopting these two evaluation approaches, we aimed to ensure that the reported performance metrics reflect the classifier's ability to generalize to new, unseen data and provide a robust estimate of its performance.

The original paper stated that the IoT/non-IoT classifier was evaluated in chronological order, but did not specify whether they used stratified or k-fold cross-validation. Cross-validation is recommended in machine learning to evaluate a model's performance on unseen data and reduce the risk of overfitting [1]. It is an effective technique for estimating model performance, particularly when the dataset is small or when the model has high variance. As such, we have explored both stratified and k-fold cross-validation techniques to ensure a thorough evaluation of our model.

4.2.5 Testing the model. Finally, the trained classifiers were evaluated on a completely new and unseen dataset to assess its generalization ability. The performance of the classifiers was measured using common evaluation metrics such as accuracy, precision, recall, and F1-score. Additionally, we reported the specificity and geometric mean metrics for the device classifier, as these metrics were also used in the original paper. However, it is important to note that these metrics are primarily designed for binary classification problems and their extension to multi-class problems is not straightforward [4]. Hence, we believe that these metrics may not provide valuable insights into the performance of the models for the device classifier.

5 EXPERIMENTS AND RESULTS

5.1 Reproducing the Classifiers

In the implementation in the paper, we see that the authors use models such as k-NN, RF, DT, and the majority voting of these models. We see that the models have very high precision and recall scores ranging from 93-96% depending on the models used, and whether over-sampled or under-sampled the data.

Comparing our results to theirs, it is clear that their classifiers are significantly more performant. This is the case for the device classifier more so than the IoT classifier which showed relatively similar results to the reference implementation. Interestingly, the specificity is more akin to that of the original paper, but this could be attributed to the high number of classes in the device classifier. As the number of devices increases, so does the probability of 'stumbling' upon a true negative (assuming a random classifier). This might explain why the specificity of the IoT classifier is not such an outlier, and it is further confirmed by the consistently lower specificity of the device classifier in the dataset without cameras (see Appendix B).

Table 1: Performance of the original and reproduced classifiers in distinguishing between IoT and non-IoT devices with over-sampling.

Metric	Method	k-NN	RF	DT	MV
Precision(%)	Reproduced	86	88	88	88
	Original	95	97	96	95
Recall (%)	Reproduced	88	89	89	89
	Original	95	97	96	95
F1-score (%)	Reproduced	87	88	88	88
	Original	95	97	96	95
Specificity (%)	Reproduced	88	89	89	89
	Original	97	98	98	97
Geometric mean (%)	Reproduced	88	89	88	89
	Original	96	97	97	96

As in the original paper, no significant difference was observed in the under- or over-sampling the dataset of the IoT classifiers (see Appendix A). Similarly, the absence of cameras in the training set for the device classifier also does not improve performance. In fact, we found a consistent performance drop of around 10% when excluding these devices.

Table 2: Performance of the original and reproduced classifiers in IoT device identification on cross-validation with chronological partitioning.

Metric	Method	k-NN	RF	DT	MV
Accuracy(%)	Reproduced	82	84	84	84
	Original	92	94	94	94
Recall (%)	Reproduced	59	63	63	63
	Original	92	94	94	94
F1-score (%)	Reproduced	63	65	65	65
	Original	92	94	94	94
Specificity (%)	Reproduced	99	99	99	99
	Original	99	99	99	99
Geometric mean (%)	Reproduced	75	77	77	76
	Original	95	96	96	96

the dataset in the validation step would not be artificially balanced to ensure more realistic scoring.

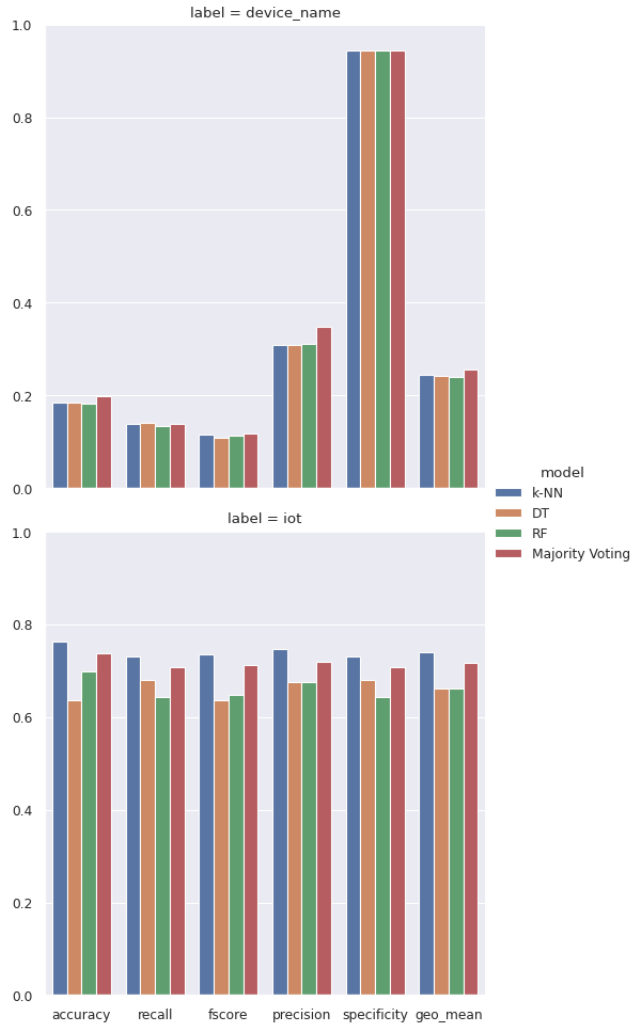

Figure 3: Performance of the IOT- and device classifiers against different scoring metrics when testing data is padded to the nearest 100 bytes.

5.2 Classifier Performance in Padded Data

We see that the classifiers in the paper "Adaptive Packet Padding Approach for Smart Home Networks: A Trade-off between Privacy and Performance" have similar performance for their models, an example is that of the Random Forest model, which has an accuracy of 96% which is in line with both the performance mentioned in the previous paper and that of our own models. They also note a performance degradation from 96% to 32% for padding method m=100 and down to 7% for padding method m=900. In our testing, we noted a performance degradation but noticed similar performance degradation for both m=100 and m=900 padding strategies with accuracy results ranging from 9-15%.

Our results show similar behavior, at least, in the case of the device classifier (Figure 3). Considering the IOT classifier, it becomes more interesting as we see a reduction in the order of 20% instead

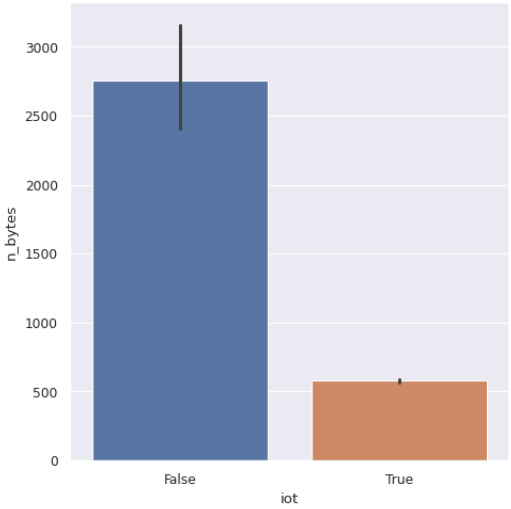

Figure 4: Mean total packet bytes found in a one-second window for IoT and non-IoT devices respectively.

of the expected 60%. When plotting the distributions of the features against the 'iot' label, it becomes clear why this performance is so high. Taking Figure 4 as an example feature, we see that the current padding strategy (m=100 bytes) does not effectively obscure IoT and non-IoT traffic. Similar, although less extreme, results can be found for the other features (see Appendix C).

5.3 Performance of Classifiers in a noisy environment.

We considered that the metrics covered in the paper were not highly indicative of a real-life scenario as explained in the next section, and so tried to filter and modify datasets to mirror that with something we felt would be more reflective of a real-life environment.

Firstly, we feel that in a normal household, all IoT device communication that is captured would be subject to some form of background noise, which could be from human-device interactions such as streaming, gaming, video-conferencing, etc, or even routine device activity such as signalling, updates, etc. In order to understand how well the classifier would perform with changes in other activity in the household, we filtered out all activity from a specific device and added in other packets from the PCAP file based on a flat probability to maintain the required noise ratio. We then calculated the accuracy of the classifier as the number of times it reproduced the correct label. We notice that accuracy drops significantly, from 91% with no noise down to 50% to a 50-50 split of device packets with random noise as seen in Figure 5.

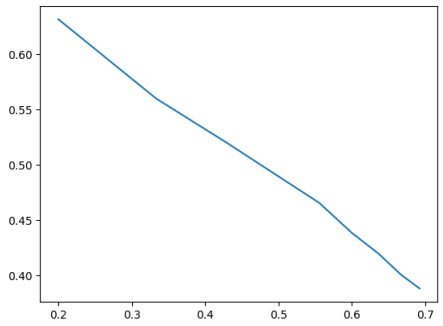

Figure 5: Classification accuracy vs. noise

Another factor that we wanted to look into is how long we would have to listen in on a device to have reasonable confidence about our prediction. In order to do this, we pick a PCAP file from our previous experiment, with a 70-30% split for device packets and noise packets. We notice that early on there are two devices with 60-40% confidence. However, over time we notice that the confidence in the correct device increases to a much more significant amount while all the other devices decrease in proportion as seen in Figure 6. However, this operates under the assumption that a device would have continuous activity for an extended period of time, which although seen in the PCAP files provided, would not translate to real life very often.

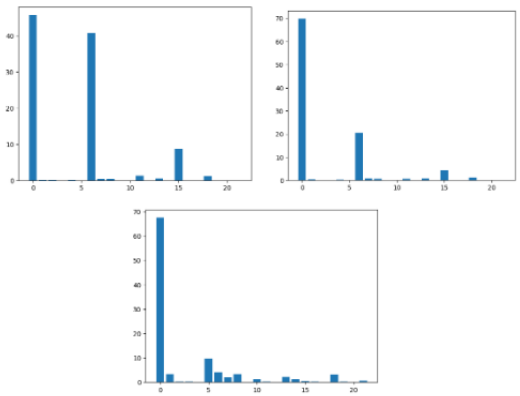

Figure 6: Confidence of classifier over time

5.4 Classifier Performance without MAC Filtering

Finally, we wanted to check how the classifier would behave without access to the MAC addresses of the devices, i.e if the attacker is listening to the packets outside of the local network. In this case, we group the packets from all devices in 1-second window intervals and attempt to classify the devices. We picked three devices with an equal amount of activity. As measures such as accuracy would not make much sense in this context, we use the confidence of the model in its classifications and sum up the confidence of all the classifications made over the entire dataset of a day. Owing to the large size of the dataset, we would expect the distribution of confidence in each class to reflect that of the original distribution of the devices. Howe,ver we notice that the confidence distribution that we get after all the predictions do not match what we expect. However, this is not conclusive as we can attribute this to the artificial dataset where an unreal amount of interactions take place in a relatively short period of time. A more detailed explanation of this is provided in subsection 6.3.

5.5 Padding Simulation

We also attempted to create a quick simulation of how we would expect the padding solution detailed in the paper "Adaptive Packet Padding Approach for Smart Home Networks: A Trade-off between Privacy and Performance". The solution in the paper consists of a model created using Mininet and SFlowRT to create a virtual network and to monitor the traffic in the network to determine the optimal padding level and set it accordingly. We consider this to be beyond the scope of this project, in the interest of reducing complexity and addressing the limited time. However, we wanted to create a basic environment to show how such a system would behave as a Proof Of Concept. In this solution, we utilize docker-compose to create a network with three containers, with one container simulating the home network, one container behaving as the router or padding mechanism and the final container behaving as the attacker which runs the classifier, as seen in Figure 7. We tried to simulate real-world performance by reading a PCAP file and sending packets over the home net, with inter-packet delay. When we run this with a sample PCAP, we noticed much lower accuracy when compared to running the same classifier on a CSV file with the same dataset, we can attribute this to delays and overhead generated in the running of the programs, which leads to inconsistencies in the way the packets are sent, received and processed. Even though we do not achieve the same results as that on the static test files, using a different or more efficient implementation would result in better performance and can be considered for future work.

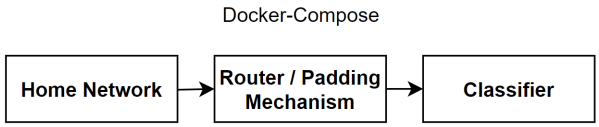

Figure 7: Simulation environment

6 DISCUSSION

6.1 Performance Disparity

While a 10% improvement disparity in the IoT classifier could be attributed to a lack of tuning and other optimizations, a performance difference of 40% is harder to justify with the same argument. Naturally, there could be a problem in our implementation which failed to come to light. Another consideration would be the way testing was done in the original paper.

The paper seems to use the terms evaluation and testing as synonyms which made it rather hard to determine where data was balanced and where it wasn't. As the data was balanced to improve the training, this data should not be used when testing the 'true'

model performance. While it might be considered a stretch, the paper never explicitly mentions the use of a separate unbalanced test set.

6.2 Analysing performance in real-world setting

Some test cases that we generated for the classifier involve a high number of devices at once, ranging from 3 up to 22. However, if we were to consider the fact that a device on average is active for 2.4 hours out of 24 hours, which is an optimistic approximation, we can calculate the probability of the number of devices being active simultaneously at a particular point in time. We can model this probability by using a probabilistic distribution such as the binomial distribution. Taking the number of IoT devices in a household as 8 and the time active as 0.1 or 2.4 hours, we only see a 20% chance that there are 2 or more devices active at the same time. Considering this, we can assume that the performance of the classifier is much greater than noticed in the experiments, as there is very little chance that a high number of devices is active at the same instant.

6.3 Limitations

It is worth noting that the use of the IEEE TMC 2018 dataset is accompanied by a number of limitations. Firstly, the dataset was collected in 2018, and as such, may not be entirely representative of the current state of IoT devices and their network traffic. Furthermore, the dataset does not provide information on the behavior or actions performed by the devices, making it difficult to infer the purpose of the network traffic based solely on packet length. Additionally, there may be potential biases or gaps in the data due to the specific smart home environments from which the data was collected.

One of the most glaring limitations compared to the original paper was resources. This project was as part of a normal course load and computing power was limited to standard workstation laptops. This means that significantly less effort went into various optimizations such as hyper-parameter tuning, and finding less optimistic results should therefore not be a surprise.

As the authors of the original paper only shared a part of the original dataset and did not publish the code used to perform their experiments, we have no way to evaluate our models with respect to those in the paper apart from comparing results, this lead to us trying to figure out a grey box of sorts with very limited information about the system itself.

Moreover, when performing these kinds of experiments, we would always like to have some kind of statistical validation to ensure that our results are consistent. Although our results seem to be consistent over the limited trials we have performed, we were not able to perform all the tests required to fully support our initial observations due to the lack of time and certain resources. Performing all these tests and fortifying them with statistical validation would be very helpful in confirming our results.

6.3.1 The Issue of MAC addresses. When we label and train the data present in the PCAP files, we do so under the assumption that we can filter the packets per device. However, it is important to note that this is not always possible and is dependent on the nature of the attacker. In case the attacker is present in the network or is listening in on WiFi, It is reasonable to expect that he can resolve the MAC addresses accordingly. However, if the attacker is listening from an external point, procedures such as NAT, or encapsulation might make it impossible for the attacker to distinguish devices based on MAC address. We test the performance of the classifiers in both these cases.

7 CONCLUSION

In this report, we aimed to reproduce the findings in "Identifying IoT Devices and Events Based on Packet Length from Encrypted Traffic" by Pinheiro et al. for identifying IoT devices based on packet length. While we were able to find similar results, they were hard to replicate exactly due to discrepancies in the dataset used in this review and the original paper. That being said, the performance of the device classifier presented in this report could not match that of the reference implementation. This could in part be explained by the poor tuning, which would also justify the difference in the performance of the IoT classifier.

Another goal was to explore the proposal and analysis of the paper "Adaptive Packet Padding Approach for Smart Home Networks: A Trade-off between Privacy and Performance". As expected, the performance dropped significantly when testing the device classifier on padded data. More interestingly, the IoT classifier did not see as much of a drop in performance as the device classifier for a padding value $m = 100$, indicating that more padding is needed to obscure IoT vs non-IoT traffic. We also noticed that the classifier performance drops with an increase in noise which could be created by other activity in the network, however, we find that listening over time would help to improve the confidence in the classification of a device, provided that the device interaction lasts for a longer period of time.

Thus, we have been able to successfully replicate part of the results of the original paper, extend upon them with recent research, and provide the methodology as open-source[1] software for other researchers.

REFERENCES

[1] Davide Anguita, Luca Ghelardoni, Alessandro Ghio, Luca Oneto, and Sandro Ridella. 2012. The'K'in K-fold Cross Validation.. In *ESANN*. 441–446.

[2] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. 2018. *imbalanced-learn User Guide.* https://imbalanced-learn.org/stable/user_guide.html

[3] Enas Elgeldawi, Awny Sayed, Ahmed R Galal, and Alaa M Zaki. 2021. Hyperparameter tuning for machine learning algorithms used for arabic sentiment analysis. In *Informatics*, Vol. 8. Multidisciplinary Digital Publishing Institute, 79.

[4] Zafiirah Hosenie, Robert J Lyon, Benjamin W Stappers, and Arrykrishna Mootoovaloo. 2019. Comparing multiclass, binary, and hierarchical machine learning classification schemes for variable stars. *Monthly Notices of the Royal Astronomical Society* 488, 4 (2019), 4858–4872.

[5] Md Mahmud Hossain, Maziar Fotouhi, and Ragib Hasan. 2015. Towards an analysis of security issues, challenges, and open problems in the internet of things. In *2015 ieee world congress on services*. IEEE, 21–28.

[6] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. 2017. Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research* 18, 17 (2017), 1–5. http://jmlr.org/papers/v18/16-365.html

[7] Zachary Chase Lipton, Charles Elkan, and Balakrishnan Narayanaswamy. 2014. Thresholding classifiers to maximize F1 score. *arXiv preprint arXiv:1402.1892* (2014).

[8] Ofer Matan. 1996. On voting ensembles of classifiers. In *Proceedings of AAAI-96 workshop on integrating multiple learned models*. Citeseer, 84–88.

[9] Christoph P Mayer. 2009. Security and privacy challenges in the internet of things. *Electronic Communications of the EASST* 17 (2009).

[10] Tingting Mu and Asoke K Nandi. 2009. Multiclass classification based on extended support vector data description. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39, 5 (2009), 1206–1216.

[11] David Opitz and Richard Maclin. 1999. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research* 11 (1999), 169–198.
[12] Maria Rita Palattella, Mischa Dohler, Alfredo Grieco, Gianluca Rizzo, Johan Torsner, Thomas Engel, and Latif Ladid. 2016. Internet of things in the 5G era: Enablers, architecture, and business models. *IEEE journal on selected areas in communications* 34, 3 (2016), 510–527.
[13] The pandas development team. 2020. *pandas-dev/pandas: Pandas.* https://doi.org/10.5281/zenodo.3509134
[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
[15] Antônio J Pinheiro, Jeandro de M Bezerra, Caio AP Burgardt, and Divanilson R Campelo. 2019. Identifying IoT devices and events based on packet length from encrypted traffic. *Computer Communications* 144 (2019), 8–17.
[16] Antônio J Pinheiro, Paulo Freitas de Araujo-Filho, Jeandro de M Bezerra, and Divanilson R Campelo. 2020. Adaptive packet padding approach for smart home networks: A tradeoff between privacy and performance. *IEEE Internet of Things Journal* 8, 5 (2020), 3930–3938.
[17] Yanjun Qi. 2012. Random forest for bioinformatics. In *Ensemble machine learning: Methods and applications*. Springer, 307–323.
[18] Arunan Sivanathan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. 2018. Classifying IoT devices in smart environments using network traffic characteristics. *IEEE Transactions on Mobile Computing* 18, 8 (2018), 1745–1759.
[19] Michael L. Waskom. 2021. seaborn: statistical data visualization. *Journal of Open Source Software* 6, 60 (2021), 3021. https://doi.org/10.21105/joss.03021
[20] Jing Zhou and Robert J Walker. 2016. API deprecation: a retrospective analysis and detection method for code examples on the web. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. 266–277.

A DATASET STATISTICAL MEASURES

Table 3: Statistical measures in bytes of Pinheiro et. al.

Device Name	Mode	Mean	Median
Amazon Echo	66	115	90
Belkin Wemo switch	66	366	118
Belkin wemo motion sensor	66	112	66
Blipcare Blood Pressure meter	54	105	59
Dropcam	156	206	156
HP Printer	86	134	86
Insteon Camera	102	104	90
Light Bulbs LiFX Smart Bulb	123	96	92
NEST Protect smoke alarm	509	294	350
Netatmo Welcome	1510	471	86
Netatmo weather station	350	171	113
PIX-STAR Photo-frame	66	111	78
Samsung SmartCam	66	421	346
Smart Things hub	60	71	60
TP-Link Day Night Cloud camera	1514	532	160
TP-Link Smart plug	172	107	66
Triby Speaker	66	104	66
Withings Aura smart sleep sensor	66	138	66
Withings Smart Baby Monitor	66	66	79
Withings Smart scale	333	333	300
iHome plug	54	98	108

Table 4: Statistical measures in bytes for the unfiltered dataset.

Device Name	Mode	Mean	Median
Amazon Echo	66	110	75
Belkin Wemo switch	66	362	118
Belkin wemo motion sensor	66	120	66
Blipcare Blood Pressure meter	54	108	59
Dropcam	156	185	156
HP Printer	86	108	86
Insteon Camera	60	115	73
Light Bulbs LiFX Smart Bulb	123	97	92
NEST Protect smoke alarm	509	300	350
Netatmo Welcome	86	510	86
Netatmo weather station	350	189	113
PIX-STAR Photo-frame	66	110	78
Samsung SmartCam	66	360	458
Smart Things	60	71	60
TP-Link Day Night Cloud camera	1514	628	343
TP-Link Smart plug	172	107	66
Triby Speaker	66	106	66
Withings Aura smart sleep sensor	66	142	66
Withings Smart Baby Monitor	66	79	66
Withings Smart scale	333	305	333
iHome	54	97	108

Table 5: Statistical measures in bytes for the filtered dataset.

Device Name	Mode	Mean	Median
Amazon Echo	66	118	74
Belkin Wemo switch	66	451	259
Belkin wemo motion sensor	66	126	66
Blipcare Blood Pressure meter	54	114	59
Dropcam	156	189	156
HP Printer	140	146	140
Insteon Camera	102	119	90
Light Bulbs LiFX Smart Bulb	123	113	123
NEST Protect smoke alarm	509	329	509
Netatmo Welcome	1510	745	192
Netatmo weather station	350	256	350
PIX-STAR Photo-frame	66	125	76
Samsung SmartCam	66	359	421
Smart Things	60	74	60
TP-Link Day Night Cloud camera	1514	680	347
TP-Link Smart plug	172	132	172
Triby Speaker	66	134	66
Withings Aura smart sleep sensor	66	157	66
Withings Smart Baby Monitor	66	83	66
Withings Smart scale	333	329	333
iHome	54	90	76

Table 6: Statistical measures in bytes for the original dataset.

Device Name	Mode	Mean	Median
Amazon Echo	66	118	74
Belkin Wemo switch	66	455	259
Belkin wemo motion sensor	66	127	66
Blipcare Blood Pressure meter	54	114	59
Dropcam	156	189	156
HP Printer	140	206	140
Insteon Camera	102	120	90
Light Bulbs LiFX Smart Bulb	123	113	123
NEST Protect smoke alarm	509	329	509
Netatmo Welcome	1510	745	192
Netatmo weather station	350	256	350
PIX-STAR Photo-frame	66	125	76
Samsung SmartCam	66	366	421
Smart Things	60	74	60
TP-Link Day Night Cloud camera	1514	702	349
TP-Link Smart plug	172	132	172
Triby Speaker	66	134	66
Withings Aura smart sleep sensor	66	157	66
Withings Smart Baby Monitor	66	83	66
Withings Smart scale	333	329	333
iHome	54	90	76

B PERFORMANCE OF THE ORIGINAL AND REPRODUCED CLASSIFIERS

Table 7: Performance of the original and reproduced classifiers in distinguishing between IoT and non-IoT devices with over-sampling.

Metric	Method	k-NN	RF	DT	MV
Precision(%)	Reproduced	86	88	88	88
	Original	95	97	96	95
Recall (%)	Reproduced	88	89	89	89
	Original	95	97	96	95
F1-score (%)	Reproduced	87	88	88	88
	Original	95	97	96	95
Specificity (%)	Reproduced	88	89	89	89
	Original	97	98	98	97
Geometric mean (%)	Reproduced	88	89	88	89
	Original	96	97	97	96

Table 8: Performance of the original and reproduced classifiers in distinguishing between IoT and non-IoT devices with under-sampling.

Metric	Method	k-NN	RF	DT	MV
Precision(%)	Reproduced	87	88	88	88
	Original	93	95	95	95
Recall (%)	Reproduced	89	89	89	89
	Original	93	95	95	95
F1-score (%)	Reproduced	88	88	88	88
	Original	93	95	95	95
Specificity (%)	Reproduced	89	89	89	89
	Original	93	95	95	95
Geometric mean (%)	Reproduced	88	89	88	89
	Original	93	95	95	95

Table 9: Performance of the original and reproduced classifiers in IoT device identification on cross-validation with chronological partitioning.

Metric	Method	k-NN	RF	DT	MV
Accuracy(%)	Reproduced	82	84	84	84
	Original	92	94	94	94
Recall (%)	Reproduced	59	63	63	63
	Original	92	94	94	94
F1-score (%)	Reproduced	63	65	65	65
	Original	92	94	94	94
Specificity (%)	Reproduced	99	99	99	99
	Original	99	99	99	99
Geometric mean (%)	Reproduced	75	77	77	76
	Original	95	96	96	96

Table 10: Performance of the original and reproduced classifiers in IoT device identification except cameras on cross-validation with chronological partitioning.

Metric	Method	k-NN	RF	DT	MV
Accuracy(%)	Reproduced	76	76	76	76
	Original	90	93	92	93
Recall (%)	Reproduced	56	55	55	55
	Original	90	93	92	93
F1-score (%)	Reproduced	58	57	56	57
	Original	90	93	92	93
Specificity (%)	Reproduced	97	97	97	97
	Original	99	99	99	99
Geometric mean (%)	Reproduced	69	68	68	69
	Original	94	96	96	96

C OTHER RESULTS FROM PADDING

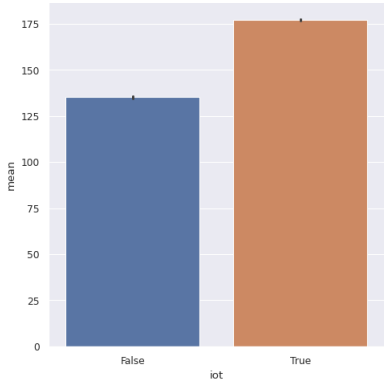

Figure 8: Mean of all the mean values found in a one-second window for IoT and non-IoT devices respectively.

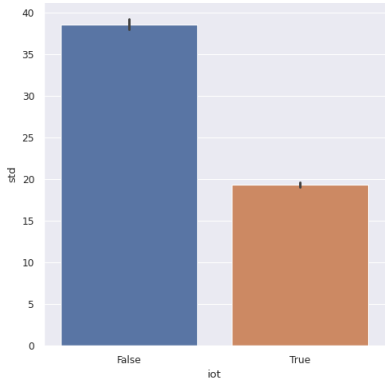

Figure 9: Mean of all the standard deviation values found in a one-second window for IoT and non-IoT devices respectively.