Efficient Convolutional Neural Network Classification of Internet-Scale Internet of Things Devices

Word Count: 4770

INTRODUCTION

This study investigated the performance differences between convolutional computational models for fingerprinting Internet of Things (IoT) devices. A review of prior work exemplified the use of computational models to classify a small number of IoT devices on local networks with high accuracy, yet more recent work provided a precedent for internet-scale IoT classification. While this precedent was set, no further work has extended on this proof-of-concept and thus this study is designed to investigate that gap by comparing specific performances between convolutional models to classify IoT devices across the whole internet. The models differed in their network structure (sequential versus windowed) and their kernel structure (1-Dimensional versus 2-Dimensional). The models tested were first optimized to ensure peak performance and then were put through general runs and 20-fold cross validation to collect data representative of the model's performance. One-way analysis of variance (one-way ANOVA) was used to determine significant differences within performances between all models in general and post hoc two sample T-Tests were performed to determine significant differences between the two specific groups. Run times of the models during the packet restriction tests were used to determine the most efficient model for internet wide IoT classification and determine a critical value for each model where it performs at its maximum capability at the shortest time cost. The combination of high accuracy and performance along with the low run time at its critical value suggests that the 1D Convolutional Neural Network is the most efficient model for the classification of Internet-Scale Internet of Things devices. Suggestions were made by the researcher to examine this phenomenon further by exploring both the classification of distinct cyber threats and the long-term classification performance of the model.

LITERATURE REVIEW

Numerous studies indicate the use of machine learning for fingerprinting IoT devices. Multiple works have developed models that accurately classify digital devices as IoT or Non-IoT, yet they are not wide enough for real-world use. Most prior works utilize service banner text to differentiate between device types. Kumar et al. designed an ensemble of IoT classifiers based on UPnP and DNS responses, HTTP banners, and network information, achieving a 92% coverage and 96% accuracy on 1000 manually labeled devices [1]. Despite this high accuracy, other research studies elected to solely observe network traffic to fingerprint IoT devices. Guo et al. posited that IoT devices can be classified by observing network flow because such devices exchange data with their manufacturer's servers. After discovering nearly 200 candidate servers accessed by 26 devices across 15 vendors, their methodology successfully identified IoT devices connected across the University of Southern California (USC) [2]. Meidan again tested Guo's postulate, using a supervised algorithm to classify manually labeled IoT devices in a localized lab environment to extract TCP packet features from such devices, including baby monitors, IP cameras, and printers in order to discriminate between IoT and non-IoT devices [3]. Miettinen expounded on Meidan's work, developing a random forest classifier trained on data from IoT device set up, allowing for the capture of device specific traits and mapping of such traits to device type [4]. Improving anomaly and infection detection is reliant on distinguishing between device types. Nguyen et al utilized machine learning classification capabilities to not only discriminate between IoT and non-IoT devices but also to detect anomalies in IoT devices, creating rapid intrusion detection at a high accuracy when trained on local networks [5]. Thangavelu extends this idea to an Internet Service Provider's (ISP) perspective, building a large-scale machine learning model capable of overcoming the limitations of past centralized approaches [6].

Recently, Pinheiro introduced the novel technique of distinguishing between devices based on outgoing packet specifications. He utilized the feature statistics of the packet flows studied, combining the mean and standard deviation of packet lengths with the number of bytes sent by each device in one second intervals [7]. Siby et al furthers by passively intercepting wireless signals in local networks to extract MAC addresses from flows allowing for IoT device identification [8]. Alternatively, Acar et al developed web scripting that identifies the presence of IoT devices running local HTTP services, disclosing vulnerabilities to the user of the script [9].

Safei Pour et al identify a shortcoming in the mentioned literature—the scope of prior work is limited to local networks and as such, does not scale to a full internet wide perspective. They instead leverage Internet wide network traffic to develop deep learning techniques to identify unreachable infected devices and predict their type from the features extracted from TCP SYN packets [10]. While their classifier is highly accurate compared to the large expanse of the Internet, it is still lower compared to models based on local networks. Yet with all these recent developments, there is no preceding work directly comparing each of these methods against each other on an Internet wide scale. In contrast to past works, a focus will be placed on a direct comparison of past models rather than solely on one new model. This work aims to fill the gap in the knowledge of internet wide IoT classifiers, justifying which model and methodology create the strongest classifier. The main reason for the importance of such a work is the need for a wide classifier in order to develop targeted security fixes for IoT devices. Creating the capability to classify each infected device on the internet allows for companies to be notified if they house an infected device and gives manufacturers the ability to make rapid target fixes to remove vulnerabilities from IoT devices. While such classifiers do exist, a full comparative study allows for improvements to be made to these classifiers. The other issue with the existing classifier Safei Pour implements is its large runtime constraint. As full monthly optimization to keep the model up to date takes over half a day to run, such a model would not be able to respond to rapidly evolving cyber threats in a reasonable time [11]. As model performance and runtime responds to the data provided, it is important to address the time constraint by focusing on key questions regarding the gaps in the literature—do more intensive models improve the classification performance of IoT devices while maintaining time efficiency, do different kernel architectures classify differently, and does reducing data given to a model adversely affect model score? Based off Safei Pour's work, it is initially suggested the more complex multi-window convolutional neural network will perform the best in IoT classification as it can "capture varying dynamics" of the data provided. It is also initially suggested that the 2D kernel convolutional model will perform worse than the 1D kernel models as the data "lacks temporal or spatial relationships," meaning the 2D model will falsely correlate based on spatial relationships causing a loss of accuracy. Finally, it is plausible that reducing data given to the model will drop the run time and performance of the model. Reducing data gives the model less to learn, reducing training time at the cost of lowered accuracy. Upon a further dive into examining these initial assumptions and answering the aforementioned questions, it will be possible to address the gaps in the current research and address a central question: which model performs most accurately and efficiently for internet wide IoT device classification?

METHODOLOGY

The underlying methodology determines similarities in network traffic flows that are exclusive to IoT devices and their respective malware in order to differentiate between malicious IoT and non-IoT devices. In order to focus on differences in the data, five baseline models will be used with comparisons only being made between the same model types. The three models will be from Safei Pour's work, as their code is made available publicly. The models are based on a convolutional neural network architecture, using multiple layers of neurons, backpropagation, and error correction to learn about the data from a given input. The convolutional models use dynamic kernels to extract and pool features from the data. The convolutional models used are 2-dimensional convolutional neural network (2-dimension kernels; 2D-CNN) 1-dimensional convolutional neural network (1-dimension kernels; 1D-CNN) and multi window 1D-CNN (Multiple 1D-CNN's in parallel; MW-1D-CNN). The convolutional models will take the matrix representation of packet flows X with t packets and features of the packets d in a form where $X \in R^{t \times d}$ and the i^{th} packet on the flow is $x_i \in R^d$. In these models, convolutional kernels $w \in R^{h \times w}$ are applied on the input matrix to discover a local correlation between the packets in the packet flow for a device and across other devices. The 2D-CNN contains consecutive two-dimensional convolutional layers (with kernels of size $w \times w$) followed by max pooling, hidden layers, and a Softmax classifier at the end. In a 1D-CNN, kernels have a width equal to the width of the input matrix $(h \times d)$. The MW-1D-CNN concatenates the outputs of varying kernel heights to better extract the correlation between the packet features. In return, the model's first layer returns an output $c_i = f(w * x_{i:i+h-1} + b)$ where $x_{i:i+h-1}$ is the sequence of packets $x_i, x_{i+1}, \dots, x_{i+h-1}$, b represents the bias, and f, the nonlinear activation function. The convolutional filter passing over the data creates a feature map $c = [c_1, \dots, c_{t-h+1}$ to which max pooling is performed overtaking the maximum value of c. The use of differing window heights $h = [2,4,6, \dots, h_{max}]$ is critical to capturing the varying dynamics of darknet packet flows [10]. These proposed convolutional models were implemented in the Keras library (version 2.3.1) with a Tensorflow-gpu backend (version 1.14) in Python 3.7. As there was a large difference between the number of samples of IoT devices and non-IoT devices, cost sensitive learning was implemented. To prevent overfitting of data, the number of epochs was set to a constant 30. In order to optimize the models, Tree Parzen Estimation was used to find the best set of hyperparameters in the search space presented in *Table 1.1* out of 100 trials with respect to loss. *Table 1.1* is reported using the *Begin:Step:End* format. The convolutional models are trained and evaluated on four NVIDIA GeForce RTX 2080TI GPU's each with 11 GB of RAM, 4352 CUDA cores, and 544 Tensor Cores to parallelize the process of training. To compare the performance of the models, the standard metrics precision, recall, F-measure, and the Area under the Receiver Operating Characteristic curve (AUC-ROC) were used. Precision is defined as the ratio of correctly classified IoT devices (true positives) and the total number of instances designated as IoT by the model (true positives + false positives) and is represented as $precision = \frac{tp}{tp+fp}$. Precision demonstrates the model's ability to designate only relevant cases as relevant. Recall is defined as the ratio of correctly classified IoT devices and the total number of IoT devices within the dataset (true positives + false negatives) and is represented as $recall = \frac{tp}{tp+fn}$. Recall is used to show the model's ability to find all relevant cases in the dataset. In order to get a better representation of the model's performance, both of these metrics are brought together using F-measure, which takes the weighted average of precision and recall, and is defined as $F - measure = 2 \times \frac{precision \times recall}{precision + recall}$. AUC-ROC measures the entire two-dimensional area underneath the ROC curve $f\big(true\ positive\ rate(tpr)\big) = false\ positive\ rate(fpr)$ where $tpr \in [0,1]$.

The models after optimization will be run 49 times. In order to find the most efficient model with a negligible effect on accuracy, the data supplied from each device will be cut short to n number of packets, limiting the data used by the model and thus speeding up its training. Each model has a minimum n number of packets needed to train and thus, the number of packets used for each model is listed as $n = [200, 210, 230, 250, 270, 290, 298]$. Accuracy from applying the model on the test dataset will be collected for each run for comparison. In addition to comparing fixed dataset performance, 20-fold cross validation will be run as the dataset is split randomly, simulating real world performance. The general procedure for 20-fold cross validation is listed below:

1. Shuffle the dataset randomly
2. Split the dataset into 20 groups
3. For each unique group:
    a. Take one group as test data (held-out group)
    b. Take the remaining groups as the training data
    c. Fit a model on the training data and evaluate the hold out.
    d. Retain the evaluation score and discard the model
4. Summarize the skill of the model using the evaluation score of the held-out group.

In order to compare model performance across all trials, one-way ANOVA was utilized to determine if there was a statistical difference between the model performances. Two sample T-

Tests were utilized for post-hoc testing in between two specific models within the data gathered. P values under 0.01 from ANOVA and T-Tests were marked as statistically significant meaning the chance of correlation due to chance was 1% or less. While a p value of 0.05 is typically used, the chosen p value was used as the variance between model runs was significantly low due to the computational nature of the study. The stochastic nature of these models also necessitates the need for a low p-value to remove any doubt caused by random chance. The source code for the developed methodology is made available to the research community via https://github.com/IPGeo-API/Efficient-IoT-Classification. For clarity, the methods are summarized below.

1. Develop the specified models and perform hyperparameter optimization using the train dataset.
2. With the optimized hyperparameters, train the models with the test dataset cut off at n packets per device and evaluate their performance using the test dataset.
    a. This is performed 49 times for each n value used.
    b. $n = [200, 210, 230, 250, 270, 290, 298]$
3. Perform 20-fold cross validation on each of the models to evaluate real world performance of the model.
4. Collect and evaluate differences between the metrics, performance, and time cost of the models.

LIMITATIONS

There are some important limitations to be noted in my research. The contributing factor limiting the scope of my work was the 2-week breakdown of the computer used to train and evaluate the models. This limited the models that could be used without crashing the computer limiting the scope of the research to only convolutional models. This possibility of evaluating other models could yield a better model than the 1D Convolutional Network for the classification of IoT devices. Another limitation regarding the physical specifications of the computer is the large power draw and spacing of components leading to high temperatures. The large power consumption of the 2080Ti's generates lots of heat when training at full capacity. Combined with the cramped spacing of the 4 GPU set up limiting cooling capabilities, internal temperatures of each GPU remained above 70^{o} for periods of up to 17 hours. In order to preserve its long-term functionality, the GPU automatically reduces its performance slightly to lower power draw and stabilize temperature. This reduction in performance could lead to artificially higher results for run times of each model, meaning run time results are less generalizable to running one model at a time with ample time for the GPU's to cool off. Regarding the model specifically, each model was randomly initialized, to simulate real-world training of the model. While this effectively simulates model performance in the real world, poor initializations could lead to skewed model performance, worsening its performance and increasing variation, leading to misinterpreted results. This was addressed by removing the few runs that initialized poorly and evaluating only the means of good evaluations for each model. Yet, using the same initialization for each model would yield more stable results with less variation, improving the clarity of results.

RESULTS

Tables and figures are listed in the appendix. Each of the three models were optimized using 100 trials on a specified search space and determined hyperparameters are shown in *Table 1.1*. Statistics for the optimized runs are shown in *Tables 1.2* and *1.3*. Tests regarding the 20-fold cross validation runs are displayed in *Tables 2.1-2.4*. Tests regarding the packet selection are displayed in *Tables 3.1-3.20. Figures 1.1-3.4* demonstrate the ROC and Precision-Recall curves of each of the models. *Figures 4.1-4.3* visually display the relationships between accuracy, loss, and training time against the number of packets the model was given to train.

Run-Time

The most prominent finding is the difference in the time to optimize, with the MW-1D-CNN taking 15 hours longer than the 2D-CNN and 16 hours longer than the 1D-CNN. This difference was tested when performing 49 training trials on the models at 298 packets and was determined as significant by both one-way ANOVA ($p = 1.12 * 10^{-307}$, see *Table 3.13*), the post hoc T-Test run between the 1D-CNN and the MW-1D-CNN ($p = 1.4 * 10^{-215}$, $M_{1D-CNN} = 75.35$, $M_{MW-1D-CNN} = 622.24$, see *Table 3.15*), and the post hoc T-Test run between the 2D-CNN and the MW-1D-CNN ($p = 8.8 * 10^{-204}$, $M_{2D-CNN} = 85.28$, $M_{MW-1D-CNN} = 622.24$, see *Table 3.14*).

Model Performance

Surprisingly, while the Multi Window model takes significantly longer to run than the sequential 1D Convolutional model, the sequential 1D model seemingly performs just as well in not only the optimized runs, but in the 20-fold cross validation tests ($p = 0.0253$, $M_{1D-CNN} = 87.66$, $M_{MW-1D-CNN} = 88.12$, see *Table 2.3*) as well. In the data restriction tests at all packet levels, the 1D-CNN outperforms the MW-1D-CNN slightly, but none the less significantly ($p = 3.53 * 10^{-5}$, $M_{1D-CNN} = 89.12$, $M_{MW-1D-CNN} = 88.71$, see *Table 3.4*). Compared to the 2D model, both of the 1D based models outperform in classification of IoT devices and discrimination against non-IoT devices across all packet lengths as determined by one-way ANOVA ($p = 1.93 * 10^{-13}$, see *Table 3.1*), the post hoc T-Test between the 2D-CNN and 1D-CNN ($p = 3.66 * 10^{-10}$, $M_{2D-CNN} = 87.14$, $M_{1D-CNN} = 89.12$, *Table 3.2*), and the post hoc T-Test between the 2D-CNN and MW-1D-CNN ($p = 2.53 * 10^{-9}$, $M_{2D-CNN} = 87.14$, $M_{MW-1D-CNN} = 88.71$, see *Table 3.3*).

This result correlates with the differences in $FMeasure$, $Precision$, and $Recall$ (*Table 1.3*) and AUC-ROC scores ($AUC_{1D-CNN} = 0.922$, $AUC_{MW-1D-CNN} = 0.920$, $AUC_{2D-CNN} = 0.910$, see *Table 1.2*; *Figures 1.1-3.4*) between the optimized models. The higher loss measure across all packet lengths tested implies that the 2D-CNN yields the most deviation between the predicted and actual result as determined by one-way ANOVA ($p = 3.85 * 10^{-4}$, see *Table 3.17*) and the post hoc T-Test between the 2D-CNN and 1D-CNN ($p = 6.97 * 10^{-4}$, $M_{2D-CNN} = 0.286$, $M_{1D-CNN} = 0.255$, see *Table 3.19*). This development is further supported by the lowest accuracy in the 20-fold cross validation test as determined by one-way ANOVA ($p = 2.15 * 10^{-11}$, see *Table 2.1*), the post hoc T-Test between the 2D-CNN and 1D-CNN ($p = 4.54 * 10^{-8}$, $M_{2D-CNN} = 86.10$, $M_{1D-CNN} = 87.66$, see *Table 2.4*), and the post hoc T-Test between the 2D-CNN and MW-1D-CNN ($p = 5.65 * 10^{-10}$, $M_{2D-CNN} = 86.10$, $M_{MW-1D-CNN} = 88.12$, see *Table 2.3*).

Packet Restriction

While model performance is important to the scope of this work, there is also a need for a focus on efficiency in the real world. By limiting the data used to train the models, run time is decreased (see *Figure 4.3*), yet there is no significant difference between accuracies on all differing packet lengths on the 2D-CNN ($p = 0.244$, see *Table 3.7*) and MW-1D-CNN ($p = 0.196$, see *Table 3.8*) as determined by one-way ANOVA. While one-way ANOVA determines there is a significant difference across all packet levels for the 1D-CNN ($p = 9.25 * 10^{-7}$, see *Table 3.5*), there is no significant difference across packet levels 230 and above ($p = 0.0489$, see *Table 3.6*). *Figure 4.1* shows the relationship between accuracy and the number of packets used for training each model.

These findings are validated by using the same comparative techniques with loss, which measures error, in lieu of accuracy. No significant difference was found between losses on all differing packet lengths on the 2D-CNN ($p = 0.448$, see *Table 3.11*) and MW-1D-CNN ($p = 0.323$, see *Table 3.12*) as determined by one-way ANOVA. While again one-way ANOVA determines there was a significant difference across all packet levels for the 1D-CNN ($p = 1.42 * 10^{-3}$, see *Table 3.9*), there is no significant difference across packet levels above 230 ($p = 0.781$, see *Table 3.10*). *Figure 4.2* shows the relationship between loss and the number of packets used for training each model.

DISCUSSION

Run Time

This aforementioned difference in run time was expected due to the complexity of the models. As all models trained have identical linear layers applied after the convolutional layers, therefore the difference herein lies solely in the complexity and structure of the convolutional layers. The maximum convolutional layers of the 1D and 2D sequential models are limited to 4 layers with one convolutional operation per layer. The 1D Multi Window Model, while only having 1 convolutional layer, has at least 40 convolutional operations running parallel to each other, leading to a drastic increase in run time. Optimizing the model resulted in it having 80 convolutional operations. Compared to the 2 convolutional operations of the 2D-CNN and the 3 operations of the 1D-CNN determined by the hyperparameter optimization, it is suggested by the significant results that run time is strongly influenced by the complexity and number of convolutional operations performed while training the model.

Model Performance

While the MW-1D-CNN is the most complex model and takes the longest by far to train, it is incorrect to automatically deem it the highest performing model due to that factor. More important to model performance is the structure of the convolutional kernel used, whether its 1D or 2D. Different kernel structures look at the data differently when determining correlations to learn. The 1D kernel solely cares about the data within each feature of the packet separately, foregoing position of the data. The 2D kernel factors in position of the data as well, leading to potential missed correlations and learning opportunities due to the position of the data. The presented results showing both models based on the 1D convolutional kernel outperforming the 2D model suggest that the 1D kernels used effectively capture the dynamics of the IoT devices and can confidently discriminate them from non-IoT devices. These results also suggest the 2D architecture falters at both discriminating against non-IoT devices (more false negatives) and classifying IoT devices (less true positives). Between models on the 1D architecture, the less complex sequential model performs similarly compared to the multi-window model in the 20-fold cross validation test. In the packet restriction test, the sequential model outperforms the multi-window model across all packets tested. This strongly suggests that kernel structure factors into model performance more than model complexity, which the more complex counterintuitively led to a small decrease in performance in one case.

Packet Restriction

Real world implementation of an IoT classifier requires not only high accuracy, but also necessitates the most efficient training time in order to adapt to changes in the IoT landscape occurring each day. By restricting the data provided to the model, run time is reduced. While reducing data given to the model logically seems to carry a tradeoff in reducing accuracy, that tradeoff does not occur within the tested packet range for the 2D-CNN and MW-1D-CNN, and only occurs in the 1D-CNN once packets are reduced to below 230. By retaining performance while reducing packets, it is suggested that there is no tradeoff for accuracy when reducing packets until a certain critical value is reached. While testing did not reveal the critical value for the 2D-CNN and the MW-1D-CNN, the critical value for the 1D-CNN was determined to be 230, meaning that the 1D-CNN retains its max performance until it is restricted to less than 230 packets. As these results were validated by analyzing the loss (error) of the model, it is strongly suggested that the model’s efficiency (shortest training time yielding maximum model performance) is maximized at the critical value.

CONCLUSIONS AND IMPLICATIONS

With the continuous adoption of the IoT paradigm ongoing in this world, security and privacy concerns can lead to devastating consequences. This work complements ongoing research on IoT devices by optimizing efficiency of a generic, passive method to classify internet wide IoT devices. By optimizing multiple binary classifiers and cross evaluating them, the work is able to provide a pathway to find the most efficient model for classification. This pathway is not only created by this work’s suggestion of the most efficient model, but also key discoveries regarding the behavior of models regarding IoT-centric data.

Cross evaluation of the accuracies of the models following analysis of their run time revealed the structure of the model's convolutional kernel impacts model performance significantly while model complexity has no effect on model. This reasoning was suggested due to the 2D-CNN being outperformed by both 1D models, both 1D models performing similarly, and the fact that 2D structure factors in position of the data. This seemingly suggests the 2D structure miss correlations leading to a lower accuracy and higher error when predicting IoT devices, in line with my initial assumptions. This development contributes to the aforementioned pathway by suggesting the best kernel structure to use for internet wide IoT classification. The 1D kernel is strongly suggested as it outperforms models based on the 2D architecture and throws out the position of the data, only factoring the data for each packet fed to the model.

Building on model performances, efficiency was also found to be a critical factor in determining the best overall model to use for IoT classification. It was first determined that reducing packets would decrease run time, yet the initial assumption that there would be a tradeoff of accuracy was only partially true. While a tradeoff does exist when decreasing packets given to the model, it does not take effect until a certain critical value is reached and packets are reduced past that value. Due to this critical value, it is suggested that a model performs at its maximum performance until data is reduced past the critical value and efficiency is maximized at the critical value. This allows for models' data input to be reduced to a point where it still performs at its maximum while taking the shortest time to train. This contributes to the discussed pathway by suggesting a baseline critical value for the 1D-CNN at 230 packets and showing the critical value for the 2D-CNN and MW-1D-CNN lies below the 200 packet minimum tested.

By analyzing accuracy and efficiency of all models, it can be determined the 1D-CNN is the most efficient model to implement for IoT classification. As accuracy is significantly important in IoT classification, the 2D-CNN was removed from consideration as it performs significantly worse than the 1D-CNN and MW-1D-CNN. While the 1D-CNN outperforms the MW-1D-CNN across all packet levels tested, there was no significant difference in performance in the 20-fold cross evaluation test. Due to the likelihood there was no difference in performance between both 1D models, efficiency and run time of the model was also tested. While the 1D-CNN has a higher critical value than the MW-1D-CNN, the 1D-CNN takes significantly less time to train, meaning it achieves the same performance as the MW-1D-CNN while taking less time, suggesting the 1D-CNN has the highest efficiency. While it was initially assumed the MW-

1D-CNN would be the best model due to its complexity intuitively leading to a higher accuracy, the suggestion that kernel structure affects model performance more than complexity explains why the 1D-CNN performed as well as its more complex counterpart. As there was no difference in accuracy and performance, the less complex model is the most efficient as a result of its lesser training time, further supporting the final suggestion of the 1D-CNN being the best and most efficient internet wide IoT classifier.

SUGGESTIONS FOR FUTURE RESEARCH

With the analysis of the model results and discussion on the implications of the results, this work has sufficiently filled the gaps in the research by using multitudes of comparative techniques to evaluate different IoT classifier model performances. Further, the new understanding generated by this work opens a route for future investigations on IoT classifiers and development of platforms utilizing such classifiers. Due to changing protocols in IoT devices, behaviors are changing rapidly, yet it is unknown whether long term changes can hinder model performance. With a longer scale, research can focus on building classifiers resistant to such changes, allowing for new cyber threats to be discovered instantly, instead of discovered when the model is updated. Additionally, further research can aim to not only classify by device type, but also classify distinct viruses and cyber threats associated with the device, allowing for the linking of devices to a central threat. For example, a device could be a part of a collection of similarly infected devices controlled by one group, making it urgent to detect in real-time. On top of that, these developments along with the current work lead to the development of threat detection platforms, which aim to alert organizations of distinct cyber threats. Finally, work must be done on improving the accuracy of classifiers to enable agile IoT characterization. Misclassification can lead to a waste of resources on non-IoT devices while allowing larger threats to go without detection, making it imperative to further the sophistication of such classification models. Overall, these findings serve to catalyze further work in this field with the aim of providing a more secure cyberspace in the future.