Hindawi
Security and Communication Networks
Volume 2023, Article ID 9840813, 1 page
https://doi.org/10.1155/2023/9840813

Retraction

Retracted: Transfer Learning Auto-Encoder Neural Networks for Anomaly Detection of DDoS Generating IoT Devices

Security and Communication Networks

Received 5 December 2023; Accepted 5 December 2023; Published 6 December 2023

This article has been retracted by Hindawi, as publisher, following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of systematic manipulation of the publication and peer-review process. We cannot, therefore, vouch for the reliability or integrity of this article.

Please note that this notice is intended solely to alert readers that the peer-review process of this article has been compromised.

Wiley and Hindawi regret that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] U. Shafiq, M. K. Shahzad, M. Anwar, Q. Shaheen, M. Shiraz, and A. Gani, “Transfer Learning Auto-Encoder Neural Networks for Anomaly Detection of DDoS Generating IoT Devices,” *Security and Communication Networks*, vol. 2022, Article ID 8221351, 13 pages, 2022.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 8221351, 13 pages
https://doi.org/10.1155/2022/8221351

Research Article

Transfer Learning Auto-Encoder Neural Networks for Anomaly Detection of DDoS Generating IoT Devices

Unsub Shafiq,[1] Muhammad Khuram Shahzad,[1] Muhammad Anwar ,[2] Qaisar Shaheen ,[3] Muhammad Shiraz,[4] and Abdullah Gani [5]

[1] *Department of Computing, School of Electrical Engineering and Computer Science, National University of Sciences and Technology, Islamabad 44000, Pakistan*
[2] *Department of Information Science, Division of Science and Technology, University of Education, Lahore 54000, Pakistan*
[3] *Department of Computer Science and Information Technology, The Islamia University of Bahawalpur, Rahim Yar Khan Sub Campus, Pakistan*
[4] *Department of Computer Science, Federal Urdu University of Arts, Science & Technology, Islamabad, Pakistan*
[5] *Faculty of Computing and Informatics, University Malaysia Sabah, Jalan UMS, Kota Kinabalu 88400, Sabah, Malaysia*

Correspondence should be addressed to Abdullah Gani; abdullahgani@ums.edu.my

Received 15 January 2022; Accepted 16 April 2022; Published 9 May 2022

Academic Editor: Muhammad Arif

Machine Learning based anomaly detection ap- proaches have long training and validation cycles. With IoT devices rapidly proliferating, training anomaly models on a per device basis is impractical. This work explores the “transfer- ability” of a pre-trained autoencoder model across devices of similar and different nature. We hypothesized that devices of similar nature would have similar high level feature character- istics represented by the initial layers of the autoencoder, while the more distinct features are captured by the innermost layer of the neural network. In our experiments, the centre-most layers of autoencoder models were re-trained with limited new data belonging to a different device. Datasets of seven Mirai infected and nine Bashlite infected IoT devices were used; each dataset also included benign records representing un-infected behaviour. We observed that the model’s detection accuracy improved by an average of 9.52% for Mirai and 44.59% for Bashlite. The highest performance improvement of 26.68% and 73.00% was observed when the anomaly model of Ecobee thermostat was tested on other devices before and after transfer learning for Mirai and Bashlite respectively. Additionally, transfer learning took 47.31% and 58.27% less time for Mirai and Bashlite respectively. We further trialed the efficacy of the autoencoder based anomaly model on flow based records of network traffic using the CIC- IDS2017 dataset. It was observed that the model performed best when distinct outliers in the dataset were present, whereas the model failed to perform decently in cases where the malicious activity did not cause significant deviation in network traffic’s footprint.

1. Introduction

RESEARCH on detecting anomalous behavior by infected Internet-of-Things (IoT) devices has focused on various Machine-Learning (ML) based anomaly detection models. These encompass supervised learning methods for anomaly classification as well as unsupervised methods for detecting outliers in a dataset [1, 2]. One research in particular [3] found the use of an auto-encoder neural network as an effective means of detecting whether a IoT device was deviating from its normal network footprint.

Massive digitization and the proliferation of smart connected devices in almost all walks of life has exponentially increased the dynamic nature of our local-area networks (LANs) [4]. Additionally, with 5G realizing high bandwidth connectivity at the edge, and computing density increasing with newer chip designs; many low cost IoT devices can afford to generate significant network traffic, which can be exploited for DDoS attacks. The number of connected IoT devices is expected to grow to 43 billion devices by 2023 [5].

The case of Distributed-Denial-of-Service (DDoS) attacks launched with aid from bot infected devices is that apart from being detrimental to the victim's network and the hosted service; it causes damages to the enterprises and Internet-Service-Providers (ISP) which host such infected devices as well. Therefore, not only do enterprises and ISPs want to protect themselves from being a victim of DDOS attacks but they also want to prevent origination of DDOS traffic generation from their networks. This brings a new set of challenges to the limelight. One such challenge is being able to detect whether a device is infected with malware, specifically "Bot binaries." Compromised devices could potentially attempt to infect other resources on the network, consume valuable compute cycles and illegally use network bandwidth and reputation to advance its adversarial activities. One of the most notorious botnets has been "Mirai" which practically demonstrated the seriousness of the security threat that infected IoT devices can be. Mirai surfaced in 2016 and in the six years since has had many variations developed from the original source code [6].

Researchers have been investigating a variety of mechanisms that could help detection of compromised IoT devices. A novel approach of using autoencoders for anomaly detection was introduced by Medan et al. [3] that commanded a nearly 100% accuracy score in detecting traffic indicative of DDoS generation from IoT devices. We based our research on this model, using the same datasets of that paper [7] and investigated how well trained autoencoder models would perform on other IoT devices of similar and different in nature.

The autoencoder model learns about a device's normal network footprint, hence generating a large error when data points co-relating to an infected behavior are given as input. We hypothesized that since IoT devices may share their capa- bilities and features to varying degrees of extent, autoencoder models should be transferable across these devices. The main contribution of the paper are as follows:

(i) Transfer-ability of an autoencoder model across IoT devices and across DDoS malwares of varying degree of similarity has been demonstrated using the N-BaIoT dataset [8].

(ii) Static features of IoT devices being representative of normal network behavior have been found to be only partially effective.

(iii) The high difference in feature values for benign and attack traffic cause a distinct jump in the mean error of the autoencoder neural network. This allows for a high accuracy in the anomaly model.

2. Literature Review

2.1. DDoS. Distributed Denial of Service attacks can be termed as the loudest form of attack in the cyber world. As a brute-force approach of making an Internet resource unavailable to legit- imate users, it has high impact on the network infrastructure that lies between the origination point and the destination as well. DDoS traffic generally consists of specially crafted service requests that are often easier to generate as opposed to respond to. In general, DDoS attacks can be reflection based or exploit based as shown in Figure 1. Reflection attacks attempt to drown the victim's service with large unwanted replies to requests the victim never made. An example can be that of a DNS reflection attack. Exploits on the other hand leverage how network devices handle certain type of packets; example, for each SYN received by a device, it sends a SYN-ACK and waits for a response; this wait period consumes buffer space as the device needs to remember the half-open state of this new connection. A SYN flood hence is able to quickly exhaust the victim's resources, hence pushing it offline.

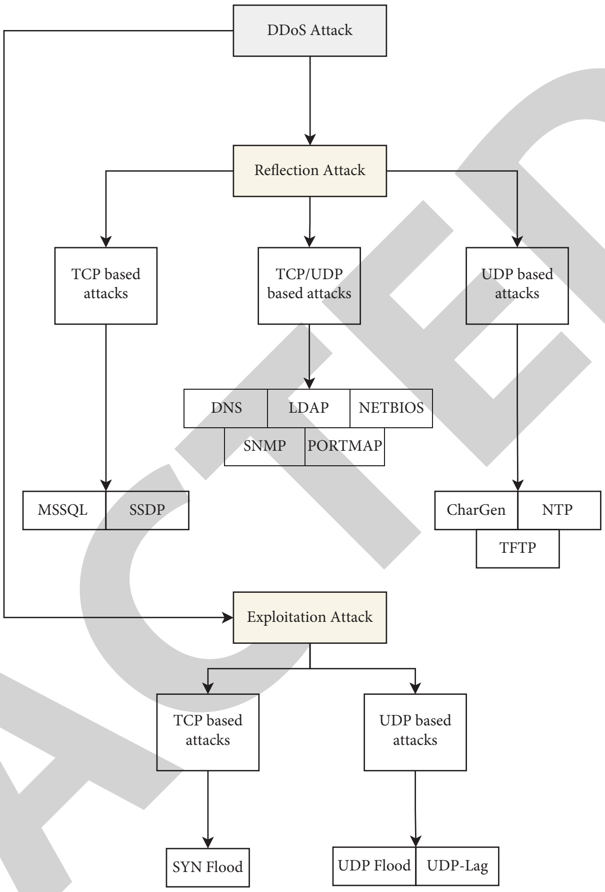


Figure 1: Categorization of DDoS traffic types.

Mirai and Bashlite use exploit based attack vectors [6, 8] in order to generate DDoS traffic. A majority of which are based on high PPS generation of UDP, TCP or HTTP request.

2.2. Bots and Botnets. The fundamental nature of botnets, i.e consisting of widely dispersed peers and command-and-controls across the Internet with masked communication methods, mean that there is no single sure-shot way that may be taken to cease all bot activity without hampering legitimate traffic. A botnet's life-cycle starts with the propagation of bot binaries [9]. The propagation phase's end-objective is to have the bot malware installed into as many systems as possible. And a variety of mechanisms can be used to this end that may or may not require human intervention. Bot malware such as Mirai actively scan for vul- nerable devices on the network, looking for devices allowing unauthenticated access or using insecure/default credentials [10]. Once breached, a small bootstrap code is run that then downloads the complete binary from the Command-and- Control (C&C). Other propagation methods include wide use of phishing emails and offerings of freeware in order to dupe users into installing the malicious bot binary into their systems. Equally important to the distribution mechanism of the bot binaries is to ensure it bypasses antivirus software which usually use signature based detection methods. Storm [10] was found to be re-encoding its malware twice every hour for this purpose. However botnets targeting IoT devices could conveniently overlook this complexity since such devices do not have the computing power necessary to run complex anti- virus software.

The next phase after infection constitutes of establishing a covert mechanism of receiving instructions from the CC, often referred to as the rallying phase [9].The prime objective in this phase is to hide the identity of the C&C and to ensure that instructions passed down to the bots are encrypted. Mechanisms include using a "fast flux" method where the C&C server's addresses are quickly rotated behind a DNS name (Storm); leveraging domain-generation-algorithms (DGA) [11, 12] where each newly infected machine attempts resolution of randomly generated domain-names in order to discover its C&C. Newer variations have exploited peer-to- peer mode of communication that further obfuscates the C&C [10, 13].

The large number of infected machines can be used for a number of malpractices that include spying, stealing of personal information and using available compute resources to attack other resources/services on the Internet. The later in particular has been used to generate large sized DDoS attacks and constitutes a persona easily identifiable in the network.

Constant evolution in the techniques of establishing botnets has kept researchers in a race to identify new mechanisms of identifying bot activity. Researches in this regard have turned to leveraging Machine Learning techniques to detect bot activity at different stages of bot infection, i.e propagation, rallying and post-infection behavior. Highnam et al. [14] targeted identification of bot malwares that used Domain-Generation-Algorithms (DGA) based domain names for finding its respective C&C. Such malware creates anomalous DNS traffic during the rallying phase. They leveraged the deterministic nature of such algorithms and trained a deep neural network composed of LSTM, CNN and ANN in order to identify whether a paritcular host was making DNS calls for domains that were DGA generated. In a similar study Tu et al. [15] leveraged the similarity of DNS queries in order to identify bot-infected machines.

Doshi et al. [16] evaluated detection DDoS traffic from consumer IoT devices by various supervised learning models. They found that K-Nearest neighbors, random forest and nueral-net models were most effective classifiers of anomalous traffic. In this case, the model is designed to detect when actual DDoS traffic gets generated.

2.3. AutoEncoder Neural Networks. Autoencoder [17] neural networks have been demonstrated as quite capable in areas such as image reconstruction and de- noising [18]. Comprising of two distinct stages, each a mirror replica of the other, the autoencoder first learns to encode input data by reducing its dimensionality and then learns to decode the compressed data such that it is as close as possible to the original input. Figure 2 represents the distinct hour-glass like shape of autoencoders. The narrowest region in the center is referred to as the Latent Representation and represents to core attributes that the neural network has learned from which it can regenerate the original input.

The loss function [19] is described as the mean difference between the reconstructed output x_R and the original input x;

$$\left(L(x, x_R) = \frac{1}{n^x}(x - x_R)^2\right). \qquad (1)$$

In such cases, the neural network is trained to perform highly well on normal data; consequently when an anomalous data point is fed to the network, the model fails to decode the data point within an acceptable level of error margin. This forms a marker of anomaly. Existing researches have explored this capability of autoencoder neural networks and used it in a variety of areas such as manufacturing [20], medical imaging [21] and network anomalies [22].

Autoencoders have enjoyed the attention of researchers in developing novel techniques for DDoS attack detection. These techniques have shown success in achieving a high accuracy with near-zero false-positive rate (FPR) [23, 24]. Yang et al. [25] have used supervised adversarial variational auto-encoder with regularization in order to detect and mitigate DDoS.

3. Procedure

3.1. Preamble. Medan et al. [3] presented use of autoencoders as an effective means for detecting DDoS traffic generation from Bashlite and Mirai infected devices. We were able to replicate their results and advanced it by evaluating performance of the trained models across different devices. We further im- plemented transfer learning by freezing all layers except the three centre-most layers of the autoencoder. Transfer-learning is often considered when only limited data is available for a similar problem. In order to simulate limited data availability, only 10% of the IoT device dataset was used for re-training the autoencoder. This was divided into a 60/20/20 split for training, optimization and testing (threshold definition).

The dataset is composed of a total of 115 features, where each feature is a statistical measure of a group of IP packets associated with the infected device. The grouping of these IP packets is dictated by their aggregation based on one or more of the following;

(i) Source-IP

(ii) Source-MAC-IP

(iii) Channel (composed of packets containing the same source and destination IP address)

(iv) Socket (composed of packets containing the same source and destination IP address and port)

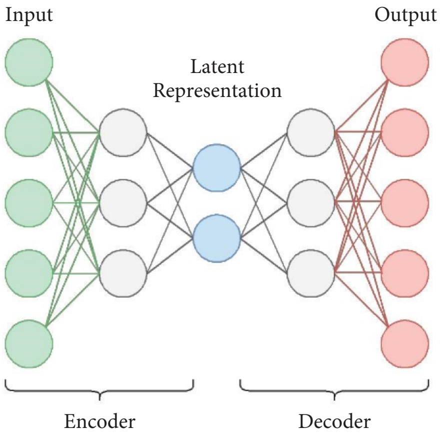


Figure 2: Logical representation of a simple autoencoder.

The aforementioned grouping is done on all packets streamed in a particular time frame. Five time frames have been used 100 ms, 500 ms, 1.5 s, 10 s and 1 min. The time buckets are a crucial aspect of the dataset since Mirai and Bashlite malware's main attack vector consists of generating a flood of packets for DDoS. The dataset is thus entirely composed of numeric values, whereas the network footprint is intrinsically captured by the aggregations.

Figure 3 plots normalized data of a benign record and a malicious record each from Mirai and Bashlite dataset of the IoT device Provision-PT-838-Security-Camera. The visual representation aids in building a mental picture of the outliers existing in the network footprint when DDoS traffic is em- anated from the device post infection. This is representative of how the data preparation has aided in the capturing the anomalous outliers.

The paper did not mention the exact structure of the autoencoder neural network used, hence we used a model with linearly decreasing layers for the encoder, where the latent representation consisted of 20% of the input features.

The NBaIoT dataset's structure is designed with a focus on the packet count across various time window sizes; however such data is seldom available in network industry. A more well-known format is the NetFlow or IPFIX, which is often used to get a holistic picture of the traffic trends in a net- work. In order to assess the efficacy of autoencoder based anomaly models against such data, the CIC-IDS2017 dataset was used. This dataset helped in highlighting the strengths and weaknesses of a simple autoencoder based anomaly detection module.

3.2. Transfer Learning on NBaIoT Dataset. Our experimentation [26] consisted of four parent iterations described as follows;

(i) With a scope limited to the Mirai dataset only; i.e the device and then transferred to the Mirai dataset of the remaining devices.

(ii) With a scope limited to the Bashlite dataset only; i.e the original model was trained on the Bashlite dataset of one device and then transferred to the Bashlite dataset of the remaining devices.

(iii) With the original model trained on Mirai dataset of a device and then transferred to the Bashlite dataset of the remaining devices.

(iv) With the original model trained on Bashlite dataset of a device and then transferred to the Mirai dataset of the remaining devices.

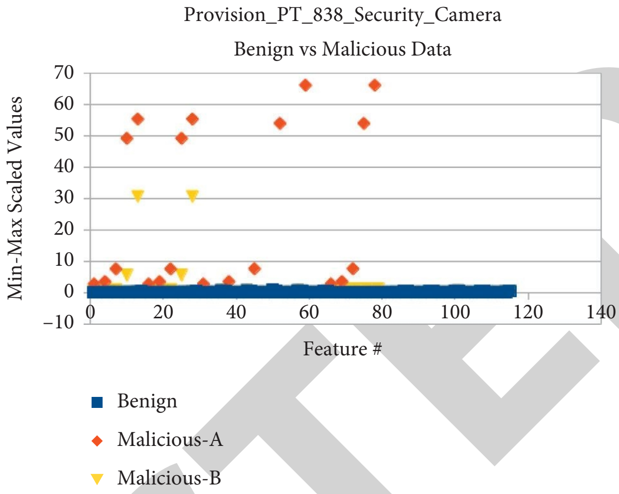


Figure 3: Example benign and malicious datapoints - NBaIoT dataset.

Each iteration of our experimentation was done in two stages. In the first stage, the autoencoder model was trained in as close resemblance as possible to the original paper, to the best of our knowledge.

The second stage was split in two parts;

(a) We ran datasets of completely unknown devices through the model and documented the model's performance.

(b) We attempted transfer learning of the model by freezing all model layers except the three centremost ones. For re- training of the model, we used only 10% of the records randomly sampled from the available dataset to simulate model training on limited data.

The autoencoder model comprises of Dense layers, whose size decreases or increase linearly as a percentage of the original input size. Figure 4 represents the shape of the autoencoder neural network used, while Figure 5 represents the frozen layers of the autoencoder when used for transfer learning.

For each device, a model was trained per malware infection using the normal (un-infected) traffic dataset. The dataset was split into 60/20/20 portions for training, optimization and validation respectively. Table 1 presents the number of datapoints used for training the original model and when simulating transfer learning.

The training dataset was also used to identify the more important features. This was done by calculating Fisher scores and ranking the 115 features in the order they had

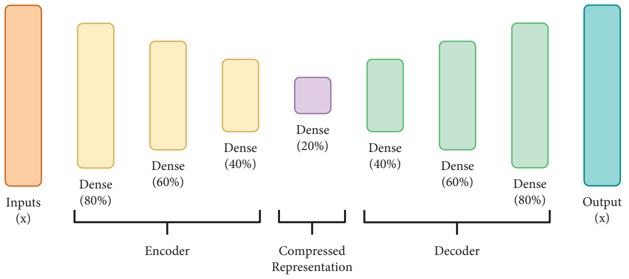


Figure 4: Autoencoder design for original model training.

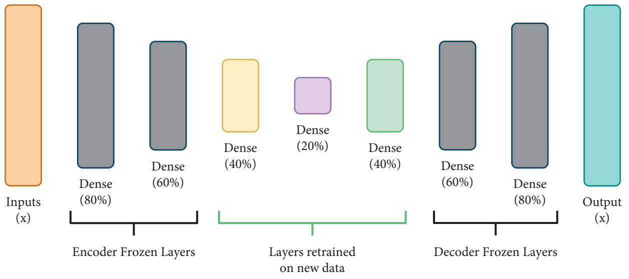


Figure 5: Autoencoder design for transfer training.

most impact on defining inter-class separation. We noticed that using only the features that had a score of 0.1 or higher gave an equally good performance as that of using all 115 features. Subsequently all of the trained models used only the features that passed the Fisher score threshold.

Each model training was allowed to run for a maximum of 100 epochs. Training was terminated if the validation loss did not decrease for five consecutive epochs. The threshold was calculated as the maximum of mean-squared error for an x percentile of normal data, the value of which was greater that 90th percentile for all iterations. This method was selected keeping in view the slight overlap in the bell curves of normal traffic's MSE and the anomaly traffic's MSE. This can be accrued to the marked difference in normal traffic patterns and DDoS traffic patterns. Since a majority of the attributes use a statistical measure sensitive to the count of packets in a time bucket, there is a distinct difference in the values of normal and the anomalous traffic.

Once a model was trained and a threshold value identified, we tested the model's performance on the dataset pertaining to other devices and noted the decrease in model accuracy. In a few cases, the model seemed to be performing better on the new device as compared to the original device; in all such cases the trained model belonged to a more feature rich IoT device as compared to the device on which it was tested.

Following this, we retrained the three centre-most layers of the existing autoencoder model and re-calculated the anomaly threshold. For this purpose, only 10% of the original dataset, randomly sampled, was used in a 60/20/20 split. The re-trained model parameters amounted to about 12.5% of the total model parameters. The new threshold value was determined on the same basis as above.

The model's performance was then tested against the dataset of the IoT device to which it was transferred to and in a vast majority of cases, we observed that the model performance had improved bringing about an accuracy at par with original model.

3.3. *Anomaly Detection on the CIC-IDS2017 Dataset.*

The structure of the feature dataset plays an important role in the performance of the neural network. A well thought-out feature creation process induces the capability of capturing anomalies in the feature values. The neural network can then learn complex non-linear relations between these features. The NBaIoT dataset has a high focus on the packet count and size, however it does not contain additional details such as protocol, packet flags, port etc.

The CIC-IDS2017 dataset consists of records that share similarity with the IETF ratified IPFIX [27] standard. A majority of the complaint network devices have the capability of exporting IPFIX records in real time, hence are an ideal candidate to replace the effort required for feature extraction. The autoencoder based anomaly model was run on this dataset and the model performance was recorded [26]. However due to limited DDoS data, we did not perform transfer-learning of the autoencoder model onto other devices. The benign dataset was split 60/20/20 for training, optimization and testing respectively. Additionally, iterations were run with a variety of optimizer functions in order to maximize performance.

4. Result Evaluation

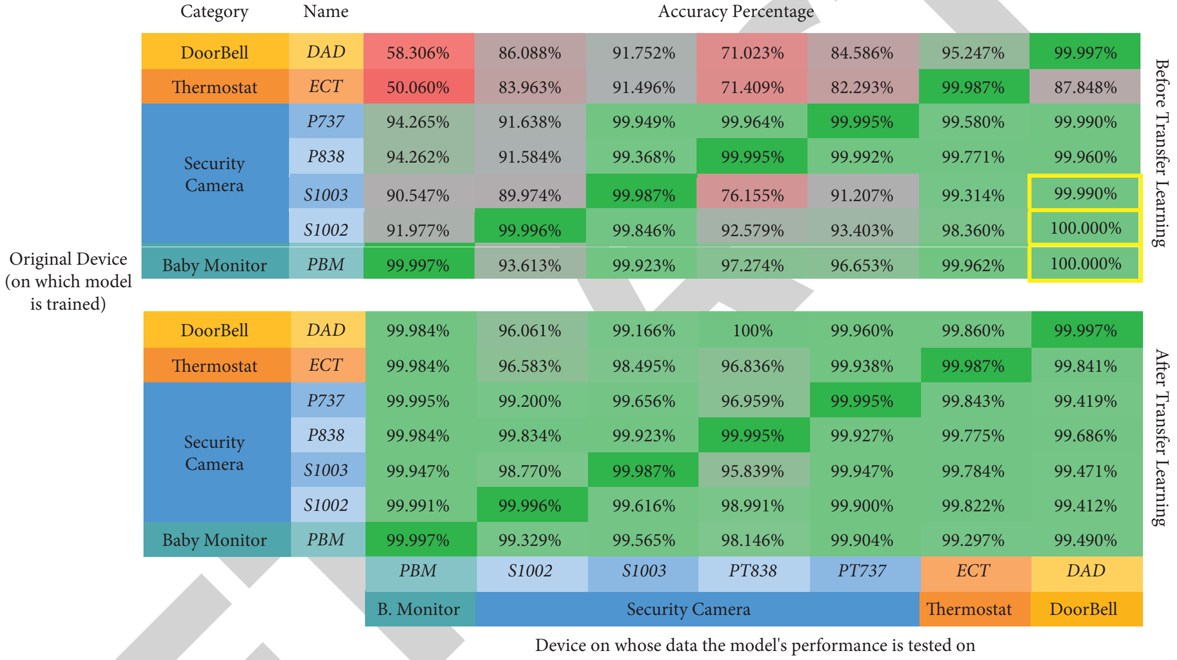
4.1. *Transfer Learning on NBaIoT Dataset.*

Figures 6 and 7 contain matrix representations of the model's accuracy when tested with unseen data of a different IoT device. The autoencoder is trained on the dataset of the devices listed in the first column on the left. It is then tested on the datasets of the devices listed horizontally in the last row. For example, the first cell containing accuracy of 58.306% corresponds to a model trained on Danmini Door- bell and tested against the dataset of Philips Baby Monitor represents the model performance before transfer-learning. Post transfer-learning, this value increases to 99.984%. The counter-diagonal of the first matrix contains the benchmark accuracy of the model's performance on the same device it was trained on.

The first iteration of the experiment consisted of transferring the anomaly model of an IoT device trained on the Mirai dataset to the Mirai dataset of remaining IoT devices. Table 2 summarizes the percentage decrease in accuracy for each device whose model was tested against the remaining devices pre and post transfer-learning. Before transfer-learning, the average decrease in model accuracy was 8.68%; with Danmini- Doorbell and Ecobee-Thermostat as the highest contributors. Models trained on these two device were the least reliable when tested against the remaining devices; with each model posting an average decrease in accuracy of 18.83% and 22.14% respectively. In all cases, transfer-learning using 10% of new device's data was found to be sufficient in restoring the models accuracy. Post transfer-learning, the average decrease in model accuracy improved to 0.752%.

Table 1: Representation of the Dataset size and its consumption during training stages.

Device	Abbreviation	Total data	Original model training			Transfer-learning		
			Training	Optimization	Validation	Training	Optimization	Validation
Danmini doorbell	DAD	49,548	29,729	9,910	9,910	2,973	991	991
Ecobee thermostat	ECT	13,113	7,868	2,623	2,623	787	262	262
Ennio doorbell	END	39,100	23,460	7,820	7,820	2,346	782	782
Philips B120N10 baby monitor	PBM	175,240	105,144	35,048	35,048	10,514	3,505	3,505
Provision PT 737E security camera	P737	62,150	37,290	12,430	12,430	3,729	1,243	1,243
Provision PT 838 security camera	P838	98,514	59,108	19,703	19,703	5,911	1,970	1,970
Samsung SNH 1011N Webcam	SNH	52,150	31,290	10,430	10,430	3,129	1,043	1,043
SimpleHome XCS7 1002 WHT security camera	S1002	46,581	27,949	9,316	9,316	2,795	932	932
SimpleHome XCS7 1003 WHT security camera	S1003	19,528	11,717	3,906	3,906	1,172	391	391

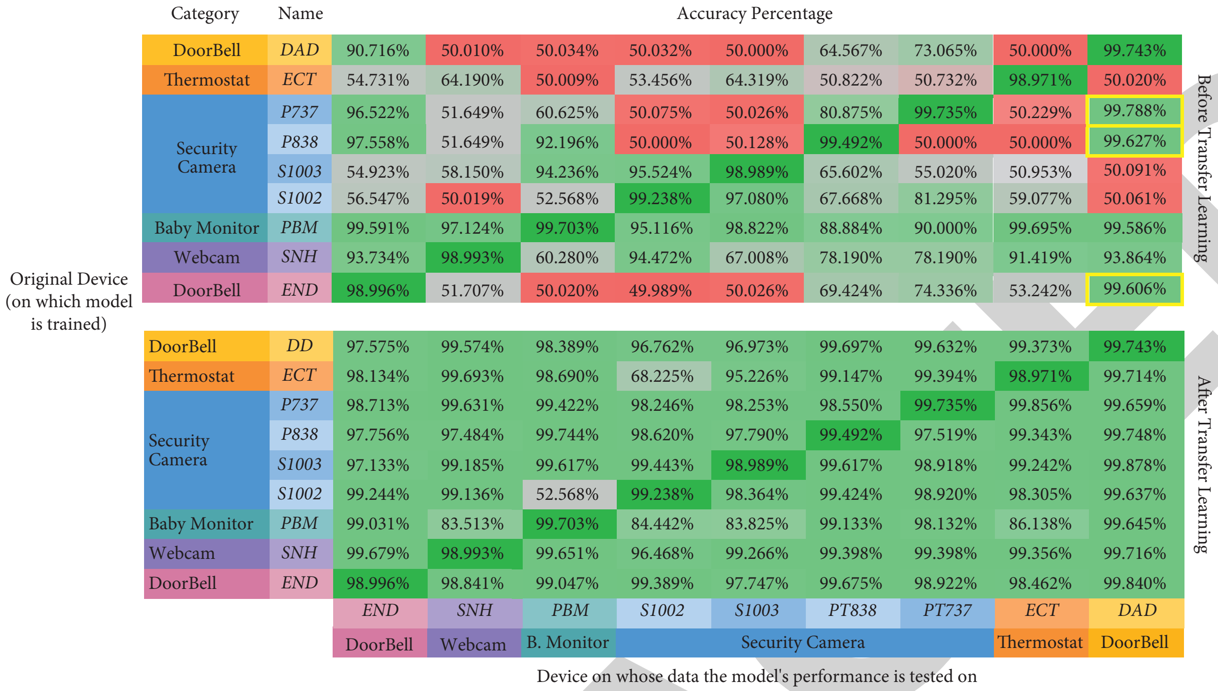


Category	Name	PBM	S1002	S1003	PT838	PT737	ECT	DAD
		B. Monitor	Security Camera				Thermostat	DoorBell
DoorBell	DAD	58.306%	86.088%	91.752%	71.023%	84.586%	95.247%	99.997%
Thermostat	ECT	50.060%	83.963%	91.496%	71.409%	82.293%	99.987%	87.848%
Security Camera	P737	94.265%	91.638%	99.949%	99.964%	99.995%	99.580%	99.990%
	P838	94.262%	91.584%	99.368%	99.995%	99.992%	99.771%	99.960%
	S1003	90.547%	89.974%	99.987%	76.155%	91.207%	99.314%	99.990%
	S1002	91.977%	99.996%	99.846%	92.579%	93.403%	98.360%	100.000%
Baby Monitor	PBM	99.997%	93.613%	99.923%	97.274%	96.653%	99.962%	100.000%
DoorBell	DAD	99.984%	96.061%	99.166%	100%	99.960%	99.860%	99.997%
Thermostat	ECT	99.984%	96.583%	98.495%	96.836%	99.938%	99.987%	99.841%
Security Camera	P737	99.995%	99.200%	99.656%	96.959%	99.995%	99.843%	99.419%
	P838	99.984%	99.834%	99.923%	99.995%	99.927%	99.775%	99.686%
	S1003	99.947%	98.770%	99.987%	95.839%	99.947%	99.784%	99.471%
	S1002	99.991%	99.996%	99.616%	98.991%	99.900%	99.822%	99.412%
Baby Monitor	PBM	99.997%	99.329%	99.565%	98.146%	99.904%	99.297%	99.490%

Figure 6: Iteration-I: Model accuracy for mirai dataset before and after transfer learning.

The second iteration was exactly similar to the first, except that the scope was limited to the Bashlite dataset. Table 2 summarizes the percentage decrease in accuracy for each device whose model was tested against the remaining de- vices pre and post transfer-learning. The average decrease in model accuracy at 30.63% was notably higher as com- pared to the Mirai dataset. Danmini-Doorbell and Ecobee- Thermostat were the highest contributors in this case as well. Models trained on these two devices lost their ac- curacy by 40.04% and 44.65% respectively when tested against other device data. In two cases, it was observed that the model was not sufficiently re-trained with only 10% of the new device's data. This is clearly evident in Figure 7 when a model trained on Ecobee-Thremostat was transferred to SimpleHome-XCS7-1002-WHT-Security- Camera with an accuracy of only 68.225% and when a model trained on SimpleHome-XCS7-1002-WHT-Security-Camera is transferred to Philips-B120N10-Baby-Monitor with an accu- racy of only 52.568%. However in both cases, increasing the size of dataset used for transfer learning to 15% significantly improved performance of the re-trained model. Post transfer- learning, the average decrease in model accuracy improved to 2.33%.

We further expanded the scope by evaluating the efficacy of transfer-learning across the two different malware datasets. The two dataset consist of a partial overlap in the types of attacks generated, namely the syn and scan types. Other attack types while similar in nature, use different protocols and approaches for generating DoS traffic. And contribute towards the rationale of transfer-learning. Table 3 summarizes the percentage decrease in accuracy for each device whose model was tested against the remaining devices pre and post transfer-learning. For an anomaly model trained on the Mirai dataset of an IoT device, its performance on the Bashlite dataset on the remaining IoT devices saw an average accuracy decrease of 32.47%. Post transfer-learning, this value reduced to 3.88%. Similarly, for an anomaly model trained on the



Original Device (on which model is trained)

Category	Name	Accuracy Percentage									
DoorBell	*DAD*	90.716%	50.010%	50.034%	50.032%	50.000%	64.567%	73.065%	50.000%	99.743%	Before Transfer Learning
Thermostat	*ECT*	54.731%	64.190%	50.009%	53.456%	64.319%	50.822%	50.732%	98.971%	50.020%	
Security Camera	*P737*	96.522%	51.649%	60.625%	50.075%	50.026%	80.875%	99.735%	50.229%	99.788%	
	P838	97.558%	51.649%	92.196%	50.000%	50.128%	99.492%	50.000%	50.000%	99.627%	
	S1003	54.923%	58.150%	94.236%	95.524%	98.989%	65.602%	55.020%	50.953%	50.091%	
	S1002	56.547%	50.019%	52.568%	99.238%	97.080%	67.668%	81.295%	59.077%	50.061%	
Baby Monitor	*PBM*	99.591%	97.124%	99.703%	95.116%	98.822%	88.884%	90.000%	99.695%	99.586%	
Webcam	*SNH*	93.734%	98.993%	60.280%	94.472%	67.008%	78.190%	78.190%	91.419%	93.864%	
DoorBell	*END*	98.996%	51.707%	50.020%	49.989%	50.026%	69.424%	74.336%	53.242%	99.606%	
DoorBell	*DD*	97.575%	99.574%	98.389%	96.762%	96.973%	99.697%	99.632%	99.373%	99.743%	After Transfer Learning
Thermostat	*ECT*	98.134%	99.693%	98.690%	68.225%	95.226%	99.147%	99.394%	98.971%	99.714%	
Security Camera	*P737*	98.713%	99.631%	99.422%	98.246%	98.253%	98.550%	99.735%	99.856%	99.659%	
	P838	97.756%	97.484%	99.744%	98.620%	97.790%	99.492%	97.519%	99.343%	99.748%	
	S1003	97.133%	99.185%	99.617%	99.443%	98.989%	99.617%	98.918%	99.242%	99.878%	
	S1002	99.244%	99.136%	52.568%	99.238%	98.364%	99.424%	98.920%	98.305%	99.637%	
Baby Monitor	*PBM*	99.031%	83.513%	99.703%	84.442%	83.825%	99.133%	98.132%	86.138%	99.645%	
Webcam	*SNH*	99.679%	98.993%	99.651%	96.468%	99.266%	99.398%	99.398%	99.356%	99.716%	
DoorBell	*END*	98.996%	98.841%	99.047%	99.389%	97.747%	99.675%	98.922%	98.462%	99.840%	
		END	*SNH*	*PBM*	*S1002*	*S1003*	*PT838*	*PT737*	*ECT*	*DAD*	
		DoorBell	Webcam	B. Monitor	Security Camera				Thermostat	DoorBell	

Device on whose data the model's performance is tested on

Figure 7: Iteration-II: Model accuracy for bashlite dataset before and after transfer learning.

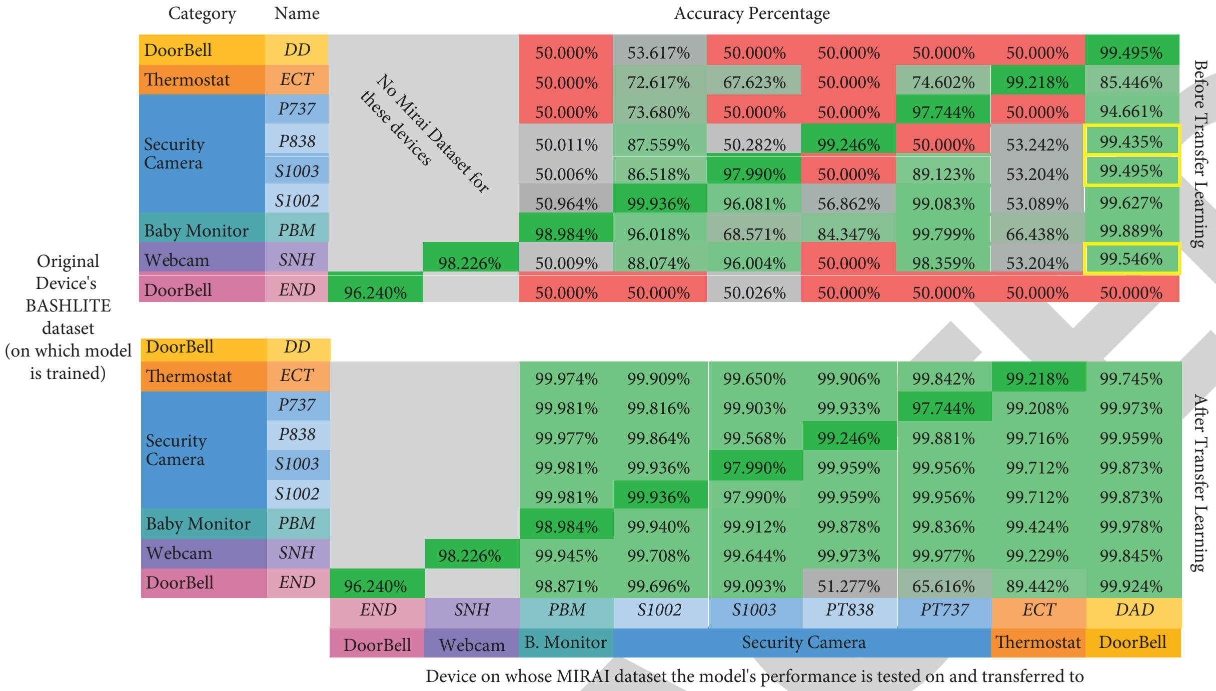
Table 2: Percentage decrease in model accuracy before and after transfer-learning (TL) across device and malware type.

IoT device	Mirai (% decrease)		Bashlite (% decrease)	
	Pre-TL	Post-TL	Pre-TL	Post-TL
DAD	18.83	0.82	40.04	1.25
ECT	22.15	1.37	44.65	4.24
P737	2.43	0.82	32.35	0.70
P838	2.51	0.14	32.01	1.00
S1003	8.79	1.03	33.77	−0.14
S1002	3.97	0.37	35.22	6.08
PBM	2.09	0.71	3.61	7.99
SNH	—	—	17.02	−0.12
END	—	—	37.07	0.01
Average (%)	8.68	0.75	30.64	2.33

Table 3: Percentage decrease in model accuracy before and after transfer-learning (TL) across device and malware type.

IoT device	Mirai to Bashlite (% decrease)		Bashlite to Mirai (% decrease	
	Pre-TL	Post-TL	Pre-TL	Post-TL
DAD	34.66	10.09	49.14	−0.33
ECT	28.68	1.45	32.76	−0.62
P737	33.30	1.78	37.19	−2.11
P838	30.26	4.87	34.42	−0.59
S1003	33.36	5.12	27.14	−1.95
S1002	37.97	1.37	24.00	0.36
PBM	29.03	2.46	13.28	−0.85
SNH	—	—	22.16	−1.56
END	—	—	48.04	10.36
Average (%)	32.47	3.88	32.02	0.30

Bashlite dataset of an IoT device, its performance on the Mirai dataset on the remaining IoT devices saw an average accuracy decrease of 32.02%. Post transfer-learning, this value reduced to 0.30%. In this iteration, it was observed that in some cases, the model's performance superseded its original accuracy on its own dataset when tested on different IoT device's data. Such instances have been highlighted in Figure 8.



Accuracy Percentage

Original Device's BASHLITE dataset (on which model is trained)

	Category	Name	END	SNH	PBM	S1002	S1003	PT838	PT737	ECT	DAD
			DoorBell	Webcam	B. Monitor	Security Camera				Thermostat	DoorBell
Before Transfer Learning	DoorBell	DD			50.000%	53.617%	50.000%	50.000%	50.000%	50.000%	99.495%
	Thermostat	ECT			50.000%	72.617%	67.623%	50.000%	74.602%	99.218%	85.446%
	Security Camera	P737			50.000%	73.680%	50.000%	50.000%	97.744%	50.000%	94.661%
		P838			50.011%	87.559%	50.282%	99.246%	50.000%	53.242%	99.435%
		S1003			50.006%	86.518%	97.990%	50.000%	89.123%	53.204%	99.495%
		S1002			50.964%	99.936%	96.081%	56.862%	99.083%	53.089%	99.627%
	Baby Monitor	PBM			98.984%	96.018%	68.571%	84.347%	99.799%	66.438%	99.889%
	Webcam	SNH		98.226%	50.009%	88.074%	96.004%	50.000%	98.359%	53.204%	99.546%
	DoorBell	END	96.240%		50.000%	50.000%	50.026%	50.000%	50.000%	50.000%	50.000%
After Transfer Learning	DoorBell	DD									
	Thermostat	ECT			99.974%	99.909%	99.650%	99.906%	99.842%	99.218%	99.745%
	Security Camera	P737			99.981%	99.816%	99.903%	99.933%	97.744%	99.208%	99.973%
		P838			99.977%	99.864%	99.568%	99.246%	99.881%	99.716%	99.959%
		S1003			99.981%	99.936%	97.990%	99.959%	99.956%	99.712%	99.873%
		S1002			99.981%	99.936%	97.990%	99.959%	99.956%	99.712%	99.873%
	Baby Monitor	PBM			98.984%	99.940%	99.912%	99.878%	99.836%	99.424%	99.978%
	Webcam	SNH		98.226%	99.945%	99.708%	99.644%	99.973%	99.977%	99.229%	99.845%
	DoorBell	END	96.240%		98.871%	99.696%	99.093%	51.277%	65.616%	89.442%	99.924%

No Mirai Dataset for these devices

Device on whose MIRAI dataset the model's performance is tested on and transferred to

Figure 8: Iteration-IV: A Bashlite model's accuracy for before and after transfer learning to a mirai dataset.

It was hypothesized that the models trained on a IoT device with overlapping hardware features of other similar device may generally fare well when tested on their dataset. In order to build context, Table 4 lists the static feature set of each IoT device. Philips Baby Monitor has the highest number of features among the pool, while the Samsung Webcam has the lowest number of features. While there were instances where a high co-relation was observed in favor of this hypothesis. For example, in the case where a model trained on Philips- Baby-Monitor was tested on the dataset of all remaining IoT devices in iteration I and II; this did not manifest in a majority of the iterations. This is evident in the iteration accuracy matri- ces represented in Figures 6–9. Danmini-Doorbell and Ecobee- Thermostat are among the devices with high number of static features as well; yet models trained on their dataset have low accuracy against other devices. Devices such as the doorbell require an external stimulus to come online, while remaining in idle or sleep mode a majority of times. As opposed to the doorbell, the baby monitor remains active at all times, provisioning live audio and video feeds. These functional properties have a major impact on the benign behavior of an IoT device on the basis of which the autoencoder model is trained.The rationale around this behavior can be due to the fact that the mere presence of certain hardware features can not represent the device's network footprint reasonably. And this should be punctuated with some quantitative representation of the IoT device's software features.

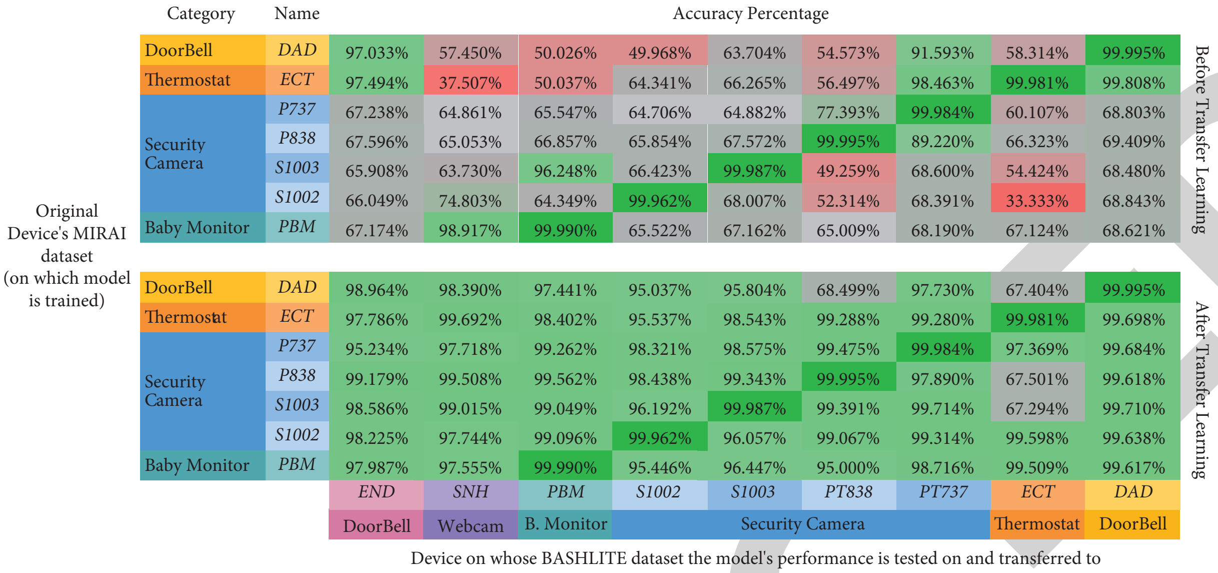
Finally, Tables 5 and 6 summarizes the average time taken in training a new autoencoder as well as when it is transfer learned to other IoT devices. On average 47.31% and 58.27% of time was saved when a model was transfer-learned as opposed to when learnt from scratch for Mirai and Bashlite datasets respectively.

4.2. Anomaly Detection on the CIC-IDS2017 Dataset. The CIC-IDS2017 dataset consists of network flow data captured in a lab environment simulating various types attacks on a multitude of devices. While the dataset itself is quite extensive, DoS attack variants were only performed against Windows Server 16. Figure 10 represents the anomaly model's accuracy in its default configuration. We observed that the autoencoder inherently performed better in detecting DoS based attacks, and clearly lacked in capability in cases that were more subtle in nature in terms of network footprint. Plotting randomly sampled data records showed that such attacks seldom reflected anomalous values in the extracted flow record, thus making it indistinguishable from the benign records. Figures 11 and 12 presents these plots.

We ran multiple iterations on the Windows Server 16 device by tweaking the hyper-parameters of the autoencoder in order to improve performance. The Adamax optimization showed mild improvement in accuracy. Figure 13 represents the summary of anomaly model's accuracy against various optimizer functions. Epochs were capped at 100, however it was observed that all training cycles remained well-below this limit. The model was stopped preemptively if the loss did not decrease for five consecutive iterations.

Table 4: IoT Device feature distribution in NBaIoT dataset.

Device	Total features	Network			Camera		Audio				Display	Sensor					Sd card
		WiFi	Lan	ZigBee	Picture	Video	In	TwoWay	Record	Speaker		Thermal	Proximity	Motion	Humidity	Noise	
DAD	8	1	1	0	1	1	1	1	0	1	0	0	0	1	0	0	0
END	8	1	0	0	1	1	1	1	0	1	1	0	0	1	0	0	0
ECT	7	1	0	1	0	0	0	0	0	0	1	1	1	1	1	0	0
P737	7	1	1	0	0	1	1	1	0	0	0	0	0	1	0	0	1
P838	7	1	1	0	0	1	1	1	0	0	0	0	0	1	0	0	1
S1003	4	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1
S1002	7	1	0	0	0	1	1	1	0	1	0	0	0	1	0	0	1
PBM	9	1	0	0	0	1	1	1	0	1	0	1	0	1	1	1	0
SNH	4	1	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0



Category	Name	Accuracy Percentage									
DoorBell	*DAD*	97.033%	57.450%	50.026%	49.968%	63.704%	54.573%	91.593%	58.314%	99.995%	Before Transfer Learning
Thermostat	*ECT*	97.494%	37.507%	50.037%	64.341%	66.265%	56.497%	98.463%	99.981%	99.808%	
Security Camera	*P737*	67.238%	64.861%	65.547%	64.706%	64.882%	77.393%	99.984%	60.107%	68.803%	
	P838	67.596%	65.053%	66.857%	65.854%	67.572%	99.995%	89.220%	66.323%	69.409%	
	S1003	65.908%	63.730%	96.248%	66.423%	99.987%	49.259%	68.600%	54.424%	68.480%	
	S1002	66.049%	74.803%	64.349%	99.962%	68.007%	52.314%	68.391%	33.333%	68.843%	
Baby Monitor	*PBM*	67.174%	98.917%	99.990%	65.522%	67.162%	65.009%	68.190%	67.124%	68.621%	
DoorBell	*DAD*	98.964%	98.390%	97.441%	95.037%	95.804%	68.499%	97.730%	67.404%	99.995%	After Transfer Learning
Thermostat	*ECT*	97.786%	99.692%	98.402%	95.537%	98.543%	99.288%	99.280%	99.981%	99.698%	
Security Camera	*P737*	95.234%	97.718%	99.262%	98.321%	98.575%	99.475%	99.984%	97.369%	99.684%	
	P838	99.179%	99.508%	99.562%	98.438%	99.343%	99.995%	97.890%	67.501%	99.618%	
	S1003	98.586%	99.015%	99.049%	96.192%	99.987%	99.391%	99.714%	67.294%	99.710%	
	S1002	98.225%	97.744%	99.096%	99.962%	96.057%	99.067%	99.314%	99.598%	99.638%	
Baby Monitor	*PBM*	97.987%	97.555%	99.990%	95.446%	96.447%	95.000%	98.716%	99.509%	99.617%	
		END	*SNH*	*PBM*	*S1002*	*S1003*	*PT838*	*PT737*	*ECT*	*DAD*	
		DoorBell	Webcam	B. Monitor	Security Camera				Thermostat	DoorBell	

Original Device's MIRAI dataset (on which model is trained)

Device on whose BASHLITE dataset the model's performance is tested on and transferred to

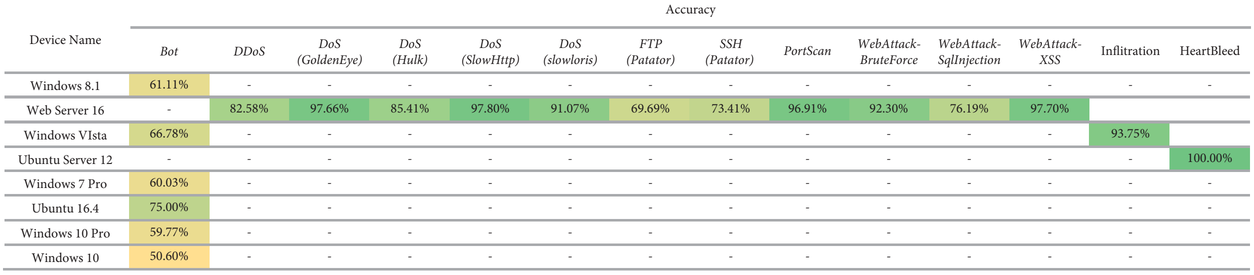
Figure 9: Iteration-III: A Mirai model's accuracy for before and after transfer learning to a bashlite dataset.

Table 5: Time saved when transfer-learning on Mirai dataset.

IoT device	Average training time (seconds)		% decrease
	Original model	Transfer learning	
DAD	118.7	67.0	43.50
ECT	48.5	25.9	46.65
P737	77.9	50.0	35.79
P838	159.3	102.5	35.67
S1003	34.0	25.0	26.47
S1002	100.9	30.3	70.01
PBM	348.9	93.8	73.12
Average	126.9	56.4	47.31

Table 6: Time saved when transfer-learning on Bashlite dataset.

IoT device	Average training time (seconds)		% decrease
	Original model	Transfer learning	
DAD	120.5	59.1	50.94
ECT	104.6	30.6	70.78
P737	149.5	93.6	37.38
P838	491.4	133.1	72.92
S1003	60.4	36.0	40.32
S1002	257.0	49.0	80.92
PBM	776.7	126.5	83.72
SNH	205.2	67.2	67.26
END	83.7	66.8	20.23
Average	249.9	73.6	58.27



Device Name	Accuracy													
	Bot	*DDoS*	*DoS (GoldenEye)*	*DoS (Hulk)*	*DoS (SlowHttp)*	*DoS (slowloris)*	*FTP (Patator)*	*SSH (Patator)*	*PortScan*	*WebAttack-BruteForce*	*WebAttack-SqlInjection*	*WebAttack-XSS*	Inflitration	HeartBleed
Windows 8.1	61.11%	-	-	-	-	-	-	-	-	-	-	-	-	-
Web Server 16	-	82.58%	97.66%	85.41%	97.80%	91.07%	69.69%	73.41%	96.91%	92.30%	76.19%	97.70%		
Windows VIsta	66.78%	-	-	-	-	-	-	-	-	-	-	-	93.75%	
Ubuntu Server 12	-	-	-	-	-	-	-	-	-	-	-	-	-	100.00%
Windows 7 Pro	60.03%	-	-	-	-	-	-	-	-	-	-	-	-	-
Ubuntu 16.4	75.00%	-	-	-	-	-	-	-	-	-	-	-	-	-
Windows 10 Pro	59.77%	-	-	-	-	-	-	-	-	-	-	-	-	-
Windows 10	50.60%	-	-	-	-	-	-	-	-	-	-	-	-	-

Figure 10: Model accuracy of a simple autoencoder based anomaly detector on CIC-IDS2017 dataset.

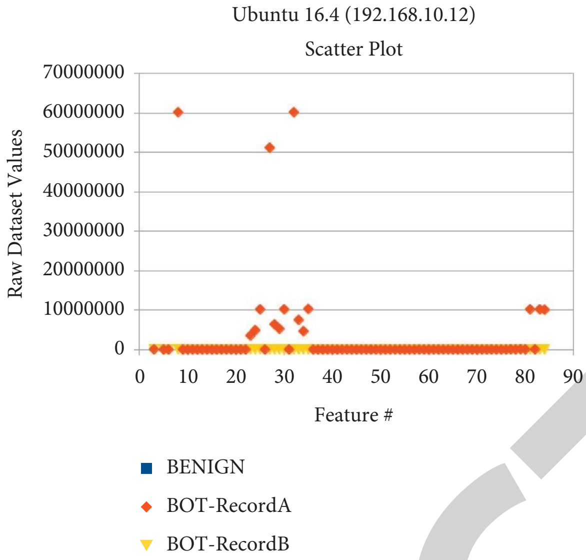


Figure 11: Plot of two malicious records from the CIC-IDS2017 dataset.

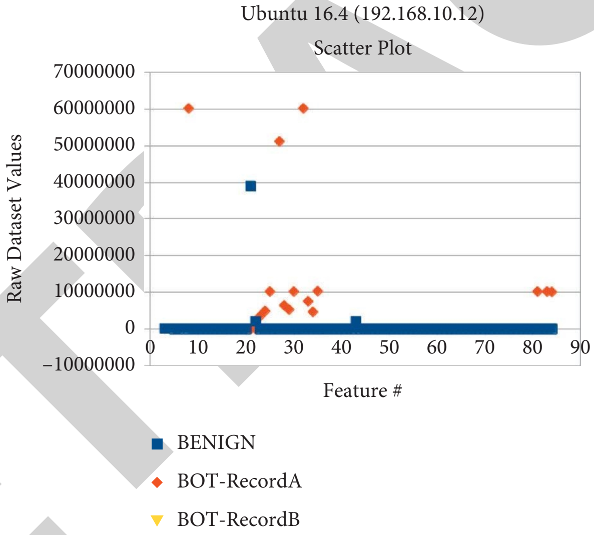
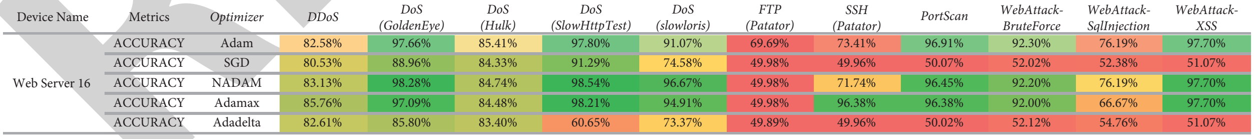


Figure 12: Overlaying a benign record on the malicious records of CIC-IDS2017 dataset.



Device Name	Metrics	*Optimizer*	*DDoS*	*DoS (GoldenEye)*	*DoS (Hulk)*	*DoS (SlowHttpTest)*	*DoS (slowloris)*	*FTP (Patator)*	*SSH (Patator)*	*PortScan*	*WebAttack-BruteForce*	*WebAttack-SqlInjection*	*WebAttack-XSS*
Web Server 16	ACCURACY	Adam	82.58%	97.66%	85.41%	97.80%	91.07%	69.69%	73.41%	96.91%	92.30%	76.19%	97.70%
	ACCURACY	SGD	80.53%	88.96%	84.33%	91.29%	74.58%	49.98%	49.96%	50.07%	52.02%	52.38%	51.07%
	ACCURACY	NADAM	83.13%	98.28%	84.74%	98.54%	96.67%	49.98%	71.74%	96.45%	92.20%	76.19%	97.70%
	ACCURACY	Adamax	85.76%	97.09%	84.48%	98.21%	94.91%	49.98%	96.38%	96.38%	92.00%	66.67%	97.70%
	ACCURACY	Adadelta	82.61%	85.80%	83.40%	60.65%	73.37%	49.89%	49.96%	50.02%	52.12%	54.76%	51.07%

Figure 13: A comparison of anomaly model's accuracy against different optimizer functions.

5. Conclusion and Future Work

Our inclination on testing the viability of transfer-learning autoencoder models of IoT devices was based on two rationales.

(1) The benign behavior of similar IoT devices on the net- work should be somewhat similar; and therefore the features learnt by the autoencoder model of these IoT devices should be similar too.

(2) Since the behavior of DDoS generating malware such as Mirai and Bashlite does not change based on device feature, the anomaly introduced by them should be similar too.

Our experimentation positively affirmed that an existing autoencoder neural network can be subjected to transfer learning with limited new data of an unknown IoT device with good accuracy. However, we did not observe a strong relation between the static features of an IoT device and its normal traffic behavior. We hypothesise that this could be due to the fact that these static features do not adequately represent the functional properties of the IoT device. Example, simply knowing whether an IoT device contains a camera only paints a black and white picture. Whereas network footprint would be impacted by the frequency of camera's use, its FPS, megapixels etc.

Our experimentation with the IPFIX formatted data has shown that while noisy DDoS traffic may be detected with a fair accuracy, this can be imporved further. We conclude that building the feature dataset as significant role in impacting the quality of the learning by the autoencoder. In general, simply focusing on the quantity and size of packets does not provide enough reference points for the neural network to learn a holistic picture.

Following can be interesting future directions;

(i) The conversion of raw packet captures into feature vectors introduces latency which can undermine the effectiveness of an anomaly detector. Minimizing the role of mid- dlewares converting raw packet-capture (PCAP) files to feature vectors and bringing them into real-time can be explored. The IPFIX framework is widely supported and has the flexibility of configuring custom attributes. This can form an interesting starting point for building a more holistic feature list.

(ii) So far, anomaly threshold is based on the mean-squared error in the reconstruction Loss and requires a program- ming logic external to the autoencoder itself. Use of RNN/LSTM can be explored to train anomaly detectors on a time-series input data stream of IoT traffic.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This is partially collaboration work with Universiti Malaysia Sabah, Malaysia.

References

[1] S. Eltanbouly, M. Bashendy, N. AlNaimi, Z. Chkirbene, and A. Erbad, "Machine learning techniques for network anomaly detection: a sur- vey," in *Proceedings of the 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, pp. 156–162, Doha, Qatar, February 2020.

[2] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning ddos detection for consumer internet of things devices," in *Proceedings of the 2018 IEEE Security and Privacy Workshops (SPW)*, pp. 29–35, IEEE, San Francisco, CA, USA, May 2018.

[3] Y. Meidan, M. Bohadana, Y. Mathov et al., "N-BaIoT-Network-Based detection of IoT botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.

[4] M. Anwar, A. Abdullah, A. Altameem et al., "Green communication for wireless body area networks: energy aware link efficient routing approach," *Sensors*, vol. 18, no. 10, p. 3237, 2018.

[5] S. A. A. Kazmi, M. K. Shahzad, A. Z. Khan, and D. R. Shin, "Smart distribution networks: a review of modern distribution concepts from a planning perspective," *Energies*, vol. 10, no. 4, p. 501, 2017.

[6] A. Marzano, D. Alexander, O. Fonseca et al., "The evolution of bashlite and mirai iot botnets," in *Proceedings of the 2018 IEEE Symposium on Computers and Communications (ISCC)*, pp. 00 813–900 818, Natal, Brazil, June 2018.

[7] Y. Meidan, M. Bohadana, Y. Mathov et al., "Uci machine learning repository - detection of iot botnet attacks n-baiot dataset," 2018, https://archive.ics.uci.edu/ml/%20datasets/detection%20of%20IoT%20botnet%20attacks%20N%20BaIoT.

[8] H. Sinanovic´ and S. Mrdovic, "Analysis of mirai malicious software," in *Proceedings of the 2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pp. 1–5, Split, Croatia, September 2017.

[9] S. Khattak, N. R. Ramay, K. R. Khan, A. A. Syed, and S. A. Khayam, "A taxonomy of botnet behavior, detection, and defense," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 898–924, 2014.

[10] L. Nkenyereye, B. Adhi Tama, M. K. Shahzad, and Y.-H. Choi, "Secure and blockchain-based emergency driven message protocol for 5G enabled vehicular edge computing," *Sensors*, vol. 20, no. 1, p. 154, 2020.

[11] Y. Fu, L. Yu, O. Hambolu et al., "Stealthy domain generation algorithms," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 6, pp. 1430–1443, 2017.

[12] Y. Zhou, Q.-s. Li, Q. Miao, and K. Yim, "Dga-based botnet detection using dns traffic," *Journal of Internet Services Information Security*, vol. 3, no. 3/4, pp. 116–123, 2013.

[13] K. N. Qureshi, E. Ahmad, M. Anwar, K. Z. Ghafoor, and G. Jeon, "Network Functions Virtualization for Mobile Core and Heterogeneous Cellular Networks," *Wireless Personal Communications*, vol. 122, 2021.

[14] K. Highnam, D. Puzio, S. Luo, and N. R. Jennings, "Real-time detection of dictionary dga network traffic using deep learning," *SN Computer Science*, vol. 2, no. 2, 2021.

[15] T. D. Tu, C. Guang, and L. Y. Xin, "Detecting bot-infected machines based on analyzing the similar periodic dns queries," in *Proceedings of the 2015 Interna- Tional Conference on Communications, Management and Telecommuni- Cations (ComManTel)*, pp. 35–40, IEEE, DaNang, Vietnam, December 2015.

[16] R. Doshi, N. Apthorpe, and N. Feamster, "Machine Learning Ddos Detection for Consumer Internet of Things Devices," in *Proceedings of the 2018 IEEE Security and Privacy Workshops (SPW)*, San Francisco, CA, USA, May 2018.

[17] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," 2020, https://arxiv.org/abs/2003.05991 w.

[18] M. Faheem, S. B. H. Shah, R. A. Butt et al., "Smart Grid Communication and information technologies in the Perspective of Industry 4.0: Opportunities and challenges," *Computer Science Review*, vol. 30, pp. 1–30, 2018.

[19] M. Anwar, A. H. Abdullah, R. A. Butt, M. W. Ashraf, and K. N. Qureshi, "Securing data communication in wireless

body area networks using digital signatures," *Technical Journal*, vol. 23, no. 2, pp. 50–55, 2018.

[20] P. Kamat and R. Sugandhi, "Anomaly detection for predictive maintenance in industry 4.0- A survey, E3S Web of Conferences," *in E3S Web of Conferences*, vol. 170, Article ID 02007, 2020.

[21] A. Sadiq, M. Anwar, R. A. Butt et al., "A review of phishing attacks and countermeasures for internet of things-based smart business applications in industry 4.0," *Human Behavior and Emerging Technologies*, vol. 2021, no. 3, pp. 854–864, 2021.

[22] A. Ahmed, K. Qureshi, M. Anwar, F. Masud, J. Imtiaz, and G. Jeon, "Link-based Penalized Trust Management Scheme for Preemptive Measures to Secure the Edge-Based Internet of Things Networks," *Wireless Networks*, 2022.

[23] K. Yang, J. Zhang, Y. Xu, and J. Chao, "Ddos Attacks Detection with Autoencoder," in *Proceedings of the NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, Budapest, Hungary, April 2020.

[24] M. Anwar, F. Masud, R. Aslam Butt, S. Mahdaliza Idrus, and M. Yazid Bajuri, "Traffic priority-aware medical data dissemination scheme for IoT based WBASN healthcare applications," *Computers, Materials & Continua*, vol. 71, no. 3, pp. 4443–4456, 2022.

[25] Y. Yang, K. Zheng, B. Wu, Y. Yang, and X. Wang, "Network intrusion detection based on supervised adversarial variational auto-encoder with regularization," *IEEE Access*, vol. 8, pp. 42 169–242 184, 2020.

[26] U. Shafiq, "Transfer learning autoencoder neural networks for anomaly detection of DDoS generating IoT devices," 2022, https://github.com/unsubshafiq/autoencoder-transferlearning.git.

[27] B. Claise, B. Trammell, and P. Aitken, "Specification of the Ip flow information export (Ipfix) protocol for the exchange of flow information," internet requests for comments, RFC editor," 2013, https://www.rfc-editor.org/rfc/rfc7011.txt.