big data and cognitive computing

MDPI

Article

Leveraging Image Representation of Network Traffic Data and Transfer Learning in Botnet Detection

Shayan Taheri, Milad Salem and Jiann-Shiun Yuan *

Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL 32816, USA; shayan.taheri@gmail.com (S.T.); milad73s@gmail.com (M.S.)

* Correspondence: Jiann-Shiun.Yuan@ucf.edu; Tel.: +1-407-823-5719

Received: 27 October 2018; Accepted: 23 November 2018; Published: 27 November 2018

Abstract: The advancements in the Internet has enabled connecting more devices into this technology every day. The emergence of the Internet of Things has aggregated this growth. Lack of security in an IoT world makes these devices hot targets for cyber criminals to perform their malicious actions. One of these actions is the Botnet attack, which is one of the main destructive threats that has been evolving since 2003 into different forms. This attack is a serious threat to the security and privacy of information. Its scalability, structure, strength, and strategy are also under successive development, and that it has survived for decades. A bot is defined as a software application that executes a number of automated tasks (simple but structurally repetitive) over the Internet. Several bots make a botnet that infects a number of devices and communicates with their controller called the botmaster to get their instructions. A botnet executes tasks with a rate that would be impossible to be done by a human being. Nowadays, the activities of bots are concealed in between the normal web flows and occupy more than half of all web traffic. The largest use of bots is in web spidering (web crawler), in which an automated script fetches, analyzes, and files information from web servers. They also contribute to other attacks, such as distributed denial of service (DDoS), SPAM, identity theft, phishing, and espionage. A number of botnet detection techniques have been proposed, such as honeynet-based and Intrusion Detection System (IDS)-based. These techniques are not effective anymore due to the constant update of the bots and their evasion mechanisms. Recently, botnet detection techniques based upon machine/deep learning have been proposed that are more capable in comparison to their previously mentioned counterparts. In this work, we propose a deep learning-based engine for botnet detection to be utilized in the IoT and the wearable devices. In this system, the normal and botnet network traffic data are transformed into image before being given into a deep convolutional neural network, named DenseNet with and without considering transfer learning. The system is implemented using Python programming language and the CTU-13 Dataset is used for evaluation in one study. According to our simulation results, using transfer learning can improve the accuracy from 33.41% up to 99.98%. In addition, two other classifiers of Support Vector Machine (SVM) and logistic regression have been used. They showed an accuracy of 83.15% and 78.56%, respectively. In another study, we evaluate our system by an in-house live normal dataset and a solely botnet dataset. Similarly, the system performed very well in data classification in these studies. To examine the capability of our system for real-time applications, we measure the system training and testing times. According to our examination, it takes 0.004868 milliseconds to process each packet from the network traffic data during testing.

Keywords: botnet detection; convolutional neural networks; machine learning; traffic-to-image transformation

1. Introduction

A self-configuring and adaptive complex network that provides connection for uniquely identifiable objects/things to the Internet through the use of interoperable communication protocols is called the Internet of Things (IoT). These objects usually have sensing, actuating, and possibly programmability capability. They may collect information from anywhere, anytime, and from anything inside the network. One of the critical issues for this network is preserving the privacy and the security of information. It is critical to protect the user information using authentication and cryptographic mechanisms. Without doing so, the IoT devices will be vulnerable to simple intrusion attempts, such as stealing passwords. Common attacks in the IoT era include: (1) Botnet, defining as a network of systems combined together with the goal of remotely controlled and distributed malware. The botnet operators are from the control center called Command and Control (C&C) Servers. They are used to perform denial of service attacks on spam and phishing emails, exploitation of online data, and stealing confidential information. This attack is very critical in the Internet of Things world since many IoT devices can become a ThingBot that makes a big network of bots around us. Detection of ThingBots is far more difficult than traditional bots since a lot of information can be sent from them at various locations. (2) Man-In-The-Middle Attack, in which an attacker is trying to interrupt and breach communications between two separate systems. This attack is dangerous since it intercepts the communication between two parties and transmits wrong messages between them. As an example, a security threat within this context can be the presence of one or more intruding passive radio-frequency identification (RFID) tags that act as interferences. Within such a passive RFID network, security threat with respect to the reliability of the system operation is the presence of a number of intruding passive RFID tags that could act as interferers. Using these malicious tags, an attacker can use them in the network as a mean to enter interference in the operation of the other non-malicious tags, causing impossible demodulation of their signals [1]. Another threat in this category is the existence of malicious, selfish, and non-cooperative nodes within the context of wireless networks. They may block the transmission of packets inside the network, known as jamming of the packet transmissions [2]. (3) Data and Identity Theft, in which attackers try to steal data and identity through social media information, smart watches, and fitness trackers for the purpose of stealing money; (4) Social Engineering, in which the attacker manipulates people for the sake of giving up their confidential information. The manipulation can be done directly or through software; and (5) Denial of Service, in which a service becomes unavailable due to overloading computing systems. In its distributed form, a large number of these systems are targeted by a botnet, in which many devices are accessed at the same time.

Also, other vulnerabilities of the IoT devices include: (a) Mirai botnet, which was able to infect many IoT devices (primarily routers and cameras), execute the denial of service attack on a domain name service provider, and disrupt many service providers, such as Twitter or other major companies. This attack takes advantage of the kernels of the devices. It is actually one of the most prominent examples of the denial of service attack using a piece of malware to find and infect the IoT devices; (b) Cold in Finland, in which two buildings were undergoing a denial of service attack in which the heating system kept restarting. (c) Brickerbot, which is a denial of service attack that simply kills the devices. (d) Botnet Barrage, which attacked many devices in a university according to which the devices of the campus became slow or inaccessible. Clearly, botnets are among the most serious malwares for conducting cybercrimes, especially in the IoT world [3–5].

The main evolution of botnet detection was with the appearance of SDBot and AgoBot. Many botnets have been introduced, namely Bashlite, Carna, as well as Mirai. Mirai emerged as a high-profile denial of service attack. This botnet can take advantage from efficient spread across the Internet, prevalent usage of insecure passwords, and wrong assumption of having simple behavior for the botnet. Mirai has many variations, but all follow the same principle. These malwares cause a large volume of infected computers (known as bots) to engage in illegal activities inside a tuple stage of: (i) spread; (ii) command and control; and (iii) launch. Some examples can be mentioned as: distributed

denial of service attacks; spreading spams; brute force cracking; identity theft; click frauds; carrying distributed computing tasks for illegal purposes; adware installation; and stealing sensitive information of users. The bots are created and controlled remotely through channels for command and control (C&C) and by botmaster (i.e., a single attacker or a group of attackers). Different communication protocols and network topologies can be considered for these channels. The scale and controllability of botnets are evolving in the Internet. The difference between them and the other malwares is their ability to perform autonomously and the possibility of equipping them with different applications, such as communication channels for receiving commands and code updates from the central unit. Therefore, the bot and the botmaster can communicate with each other in regards to the working status and the status update. Within this communication, the IP address, domain name system (DNS), and node identification number for locating the controlling unit are known.

These malwares are used for transmission of malicious information and codes, sending spams, deceiving and generating virtual attacks, launching strong attacks, and stealing sensitive information. The activities of botnets make most of the ongoing traffic data in the Internet. Their scope of malicious activities can include economical organization, military stations, and house appliances (such as compromised refrigerators, and security cameras). The type of networks used for botnets is peer-to-peer that improves their resiliency in front of defense mechanisms. The manager of the infected computers (or bots), called botmaster to distribute the commands. It is difficult to detect the bots within this network topology due to their evasive nature. The operations of Mirai Botnet and the structure of a typical botnet are shown in Figures 1 and 2. The importance of botnet communication in real traffic data can be observed in Figure 3 (captured from [6]).

Detection of these bots can be classified into four categories using: (a) signature from data; (b) anomaly from data; (c) DNS of data; and (d) machine/deep learning-enabled engines. In the first category, the botnets have certain characteristics that can be used for their identification using network traffic. The best candidate methods to detect bots are real conditions. The anomalous network behaviors are discovered by the anomaly-based approaches. These techniques look for specific parameters, such as delay from network and the running operations on the ports with less usage. If the normal traffic data are mixed with the anomalous traffic data, then these techniques may be weak for demonstrating their functionality. The DNS information of the bots can be used by the DNS-based techniques in order to detect them. All the discussed techniques until this point are traditional in a sense that they can only detect the well-known bots. In order to detect new (and unknown) bots, machine learning-based systems need to be developed since they are capable of understanding and predicting unexpected patterns from the network traffic. Using these methods, useful features are extracted from the data and learning algorithms are utilized with the goal of recognizing bots.

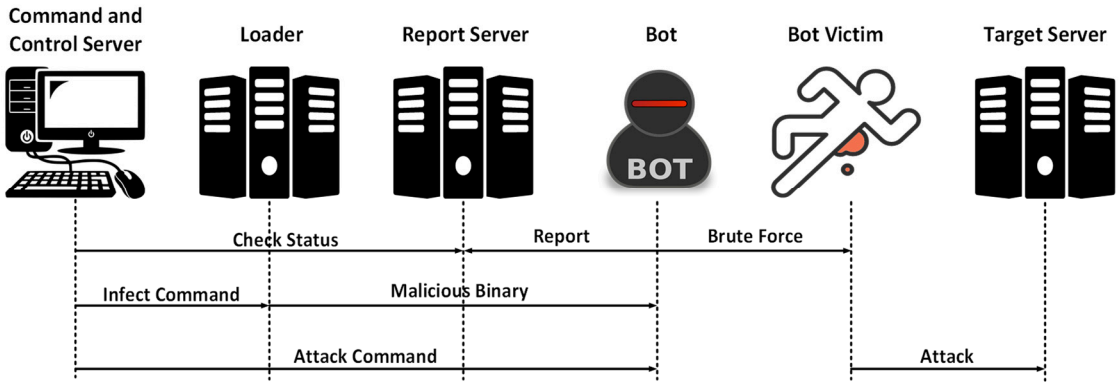


Figure 1. The operations of the Mirai botnet.

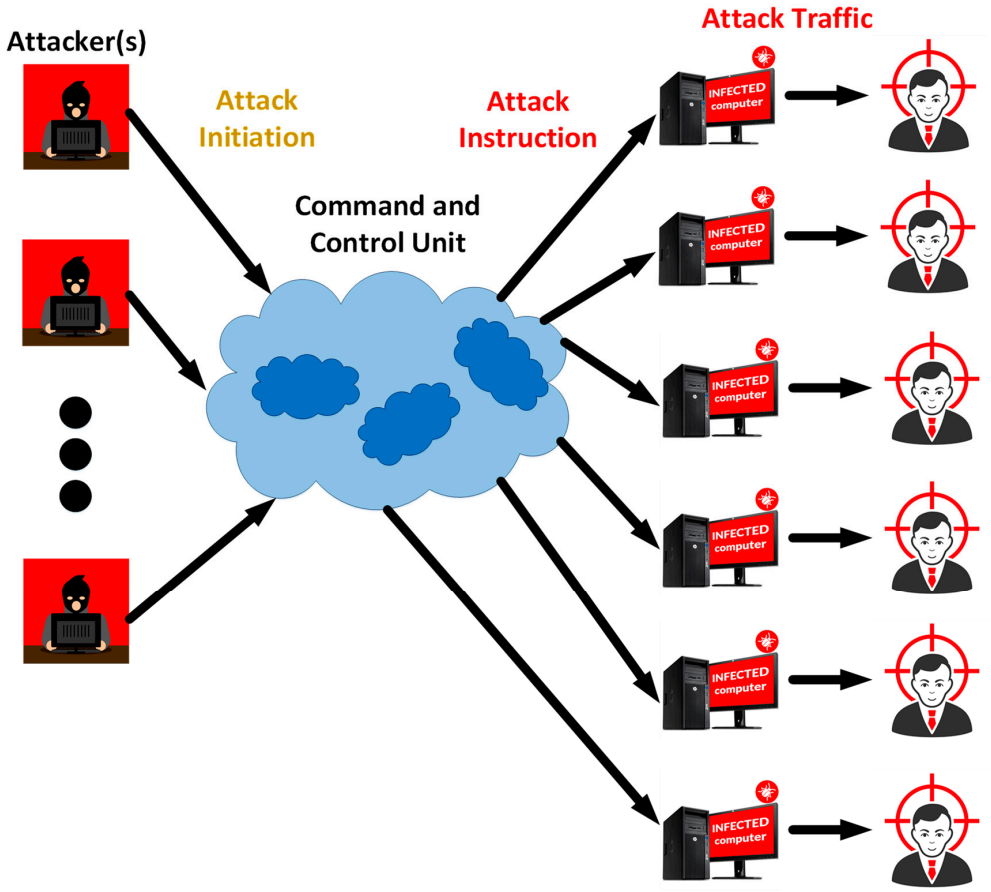


Figure 2. The network architecture for a typical botnet.

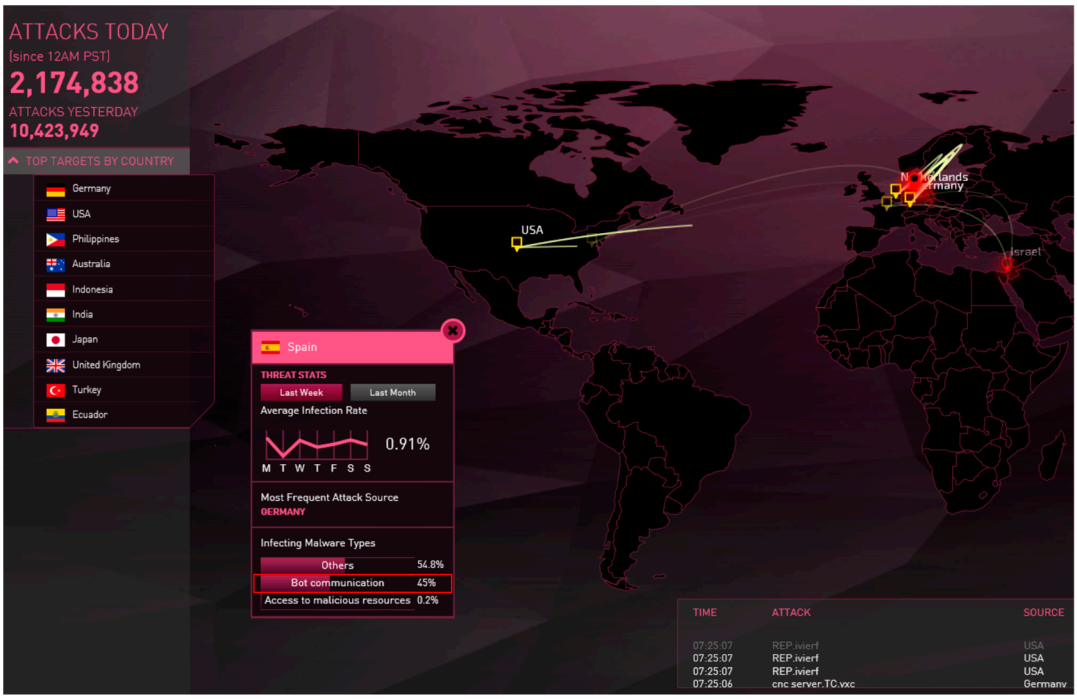


Figure 3. The ongoing cyber-attacks in real world—having 45% bot communication among all the ongoing malware traffic data in Spain last month (at the time of accessing [6]).

Two classical machine learning techniques are using: (a) statistical methods and (b) behavioral techniques. These techniques perform the task of classification through extracting patterns from empirical data using a set of selective features. These techniques may suffer from designing the best features for classification. In order to tackle this problem, a new approach has been proposed according to which the features are automatically learned from raw data, which is called representation learning. A typical approach in this regard is representation learning that demonstrated acceptable performance in many domains, including image classification and speech recognition. One of the most well-known approaches for representation learning is convolutional neural network (CNN). We can enter the traffic data into the CNN in numeric or image format.

Contributions: What we propose in this work is a deep learning-based engine for botnet detection system that inputs the image representation of network traffic data into a fine-tuned pre-trained CNN for botnet detection. Also, the system performance is compared with its counterparts as well as the traditional machine learning-based systems. The organization of the following parts of this paper is according to this: Section 2 presents the related works. Section 3 shows the background of the techniques used in our methodology. Our designed deep learning-based engine for botnet detection is described in Section 4. The results, evaluation, analysis, and limitations of the experimental evaluation of this method are all given in Section 5. We conclude the work in Section 6.

2. Related Works

As it was discussed before, we can divide the botnet detection techniques into four categories based on usage of: (a) data signature; (b) data anomaly; (c) DNS of data; and (d) deep/machine learning-based engines. Most of the proposed methods so far are included in the category of traditional botnets. The authors of [3] proposed using signature-based and behavior-based analysis according to which a correlation engine is employed for generating the final detection results and adjusting them using a multi-feedback engine. The network traffic has been monitored for suspicious behavior through looking at parameters such as nicknames, servers, and ports with less commonality. Another botnet detection method uses n-gram analysis and a scoring system [4]. Another work in this domain is based on a detection model being used for recognizing the characteristic fact of the commands from botmaster and his bots [5]. In [7], the dialogs are monitored based on the events and the sources of information. Once a dialog matches what exists for a successful infection, then it is called an infection.

With respect to anomaly-based approaches, the networks and servers for command and control as well as the infected hosts are identified in a network for local areas with the assumption of having no knowledge of signatures as well as the C&C server addresses. Through this process, what is captured is the spatial-temporal correlation of network traffic within the same botnet. The captured information is inputted to statistical algorithms for botnet detection [8]. An algorithm for universal compression was employed for a detection system, named the Lempel Ziv, and was proposed in [9] according to which the traffic data is given a probability and the new traffic data is estimated based on a specific likelihood. In [10], the statistical features of the requests with HTTP format from the client-side and the responses of DNS from the server-side are gathered for detection of HTTP-based C&C traffic. In this process, three unsupervised anomaly detection techniques are employed for isolation of suspicious information under transmission.

Regarding DNS-based approaches, a mechanism has been proposed in [11] that performs classification of the DNS traffic data from single hosts and a group of hosts periodically using duplicate queries. The power spectral density is leveraged in order to recognize major frequencies in the periodic DNS queries of botnets [12]. Using this technique, the timing information of queries are utilized without considering the number of queries and domains. The authors of [13] utilize clustering and classification algorithms in order to remove noise and separate domains from each other based on similarity in the characteristic distribution of domain names. The bots from each botnet and offline malwares of the machines with infection are identified using collaborative filtering.

The recent and more effective approaches to detect botnets are machine learning-based that are more beneficial for IoT botnets. In these techniques, the network traffic data are mostly analyzed at the network-level, analyzing flows of the traffic conversations. The authors of [14] showed the features based on flow such as small packet rate, initial packet length, packet ratio, and bot response packet can be given to boosted decision tree, naïve Bayesian classifier, and Support Vector Machine (SVM) algorithms for botnet detection. In [15], the authors showed 21 flow-based features categorized based on byte, time, behavior, and packet. Then, the features with the most useful information are selected for detection of botnet. There are three important behaviors for network, called long active communication, connection failure, and network scanning for feature extraction. The used data in this process are the network layer flows and the transport layer. The botmasters are identified using particle swarm optimization and K-means algorithms. The authors of [16] proposed using deep learning in developing a botnet detection model according to which the packets are converted into tokenized integer format by word embedding and inputted to a bidirectional long short term memory-based recurrent neural network (BLSTM-RNN) for learning from and detecting of botnets. The BotHunter was proposed in [6] as one of the earliest botnet behavior detection systems that tries to correlate the generated alarms by Snort (which is a free and open-source network intrusion detection system) to the behavior of an individual host. The other detection system is BotMiner [17] that considers group behavior of individual bots within the same botnet. A decision tree has been proposed by [18] in which they use reduced error pruning algorithm for improving the performance. Usage of IP addresses as a reputed feature is another aspect of their work that resulted in an increase in their detection accuracy. [19] demonstrated that detection of infected IoT devices included inside a botnet is made possible with a logistic regression-based mode. In this model, the probability of having a bot for a device is calculated. The IoTSec has been proposed as a software defined networking approach to enhance the security policies specifically in front of IoT bots [20]. The characteristics of network traffic data were studied in [21] for the purpose of botnet detection and control phase under the radar through malicious email, website, file sharing networks, and ad-hoc wireless networks. In [22], a host-based botnet detection system has been proposed in which a flow-based detection method is utilized that considers the log files of the machines. The authors of [23] developed a system for detection of botnet that leverages data mining approach for analysis of the network traffic data at the gateway level. An adaptive learning rate multi-layer feed-forward neural network has been used in [24] for botnet detection.

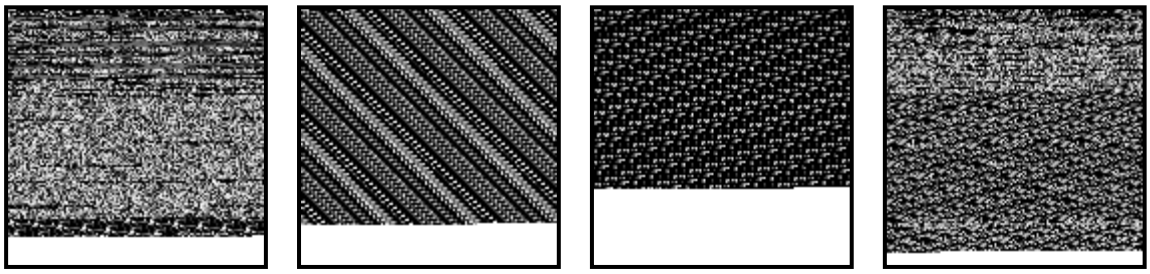
3. Background

In this section, the fundamental concepts and techniques that are employed in the proposing system are discussed. These concepts and techniques include network traffic data to image conversion, transfer learning, and the pre-trained DenseNet deep neural network are all explained.

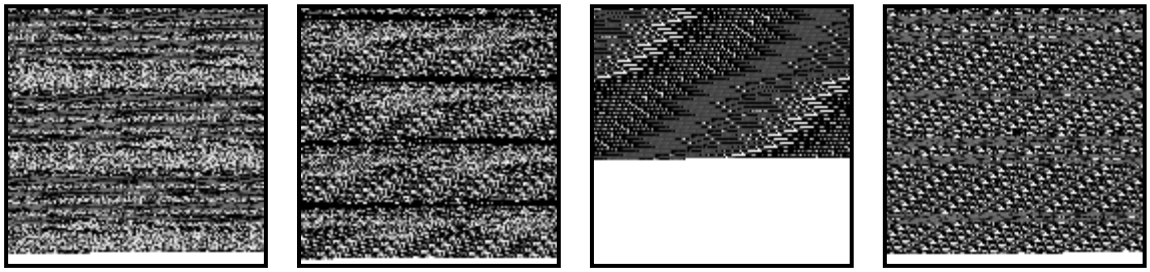
3.1. Network Traffic Data to Image Conversion

The concept of network traffic data to image conversion has recently proposed in [25] according to which the raw data go through a three-stage data processing before its image format come out. The stages in this data processing can be described as: (a) Traffic split, in which a continuous raw traffic data in the PCAP (abbreviation for packet capture) format is split to multiple discrete traffic units. Depending on the type of data representation, the output can be either PCAP for flow-based all layers or BIN (binary) for flow-based application layer (or the seventh layer). (b) Traffic clear, in which the media access control (MAC) address and the IP address are randomized in data link layer and internet protocol (IP) layer at first. This processing step is called anonymization/sanitization. This step needs to be done in cases the traffics are from different networks. Once this step is done, then it is required to remove duplicate or empty data. The duplicate data are due to the same content in some packets and the empty data are due to no application layer in some packets. Without removing these, there will be bias in training CNN. (c) Image generation, in which the outliers are removed at first. The outliers are defined as traffic data that have significantly different size from the rest of the data. Next, the

remaining data are adjusted to a specific length. Those data that are less than the chosen length are padded with 0XFF (i.e., the numerical representation of white color). The result data with the same size are transformed into images with gray-level format. In this work, for the first time the technique of converting network traffic data to images are used for botnet detection. Figure 4 shows some samples of normal and botnet network traffic data (from the CTU-13 dataset [26]) in the image format.



(a) Normal traffic data in the image format.



(**b**) Botnet traffic data in the image format

Figure 4. A few samples of converted the normal and the botnet network traffic data to image from CTU-13 dataset [26].

3.2. Transfer Learning

The process of predicting an unseen data is done by the statistical models trained on an already collected labeled or unlabeled training data. If there is sufficient labeled data for gaining knowledge, then a good classifier can be built. Within this context, the semi-supervised classification is leveraged to solve the problem of insufficient data with labels for building a good classifier through utilizing unlabeled data with large set and labeled with small set. The main assumption is having the same distribution for labeled and the unlabeled data in this domain Also, each task is learned from the scratch in these techniques.

How about when there is noticeable insufficiency in the labeled data? Transfer learning helps overcome this issue based on allowance of difference among different domains, tasks, and distributions used in training and testing. This technique helps apply knowledge learned previously (from source tasks) to tackle new challenges (target tasks) and with higher quality solutions. The knowledge extracted from certain tasks with general information is applied to a target task with unknown information.

3.3. The Neural Networks with Pre-Trained Weights

A neural network with pre-trained condition can be defined as a model that has been trained on another problem as a starting point to be used for solving a similar problem. The training happens through running images from thousands of object categories, such as keyboard, pencil, coffee, and many animals. According to this technique, there is no need to create a model from scratch. In fact, the pre-trained neural network is utilized to extract powerful information as well as features from diverse images. The acquired knowledge can be applied to other tasks. The other benefit that

comes with this approach is a faster and easier training process for other tasks since instead of training from scratch, the network only needs to be fine-tuned for another targeted task.

As an example for certain tasks such as object recognition or cars with self-driving and –learning capability, maybe months are needed to build the model from scratch. Using pre-trained networks, the faster execution time comes at the expense of less accuracy. These models help learn an algorithm or try out existing frameworks. This technique is highly desirable for applications with time limitations and computational resources do not allow us to build a ground-up model. Therefore, a pre-trained model may be used as a benchmark for improvement of existing models or testing the already built model for different applications.

One option for the pre-trained model is convolutional neural network. The common layers in a CNN are named as input layer, convolutional layers, rectified linear unit, cross channels normalization layers, average pooling layers, max pooling layers, fully-connected layers, dropout layers, softmax layers, and output classification layers. In training the network, the learning occurs from the data directly and the complexity and detail of learned information is augmented from layer to layer. Nowadays, it may not be fully practical to train a CNN from scratch especially due to time limitation. As a result, a CNN with pre-trained condition can also be used as a feature extraction module.

3.4. The Classification Measures

In order to evaluate our system, we measure accuracy, precision, recall, F1 score, and Kappa. The analytical equations are given as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{1}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

$$\text{F1 Score} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \tag{4}$$

$$\text{Kappa} = \frac{\text{Actual Accuracy} - \text{Random Accuracy}}{1 - \text{Random Accuracy}} \tag{5}$$

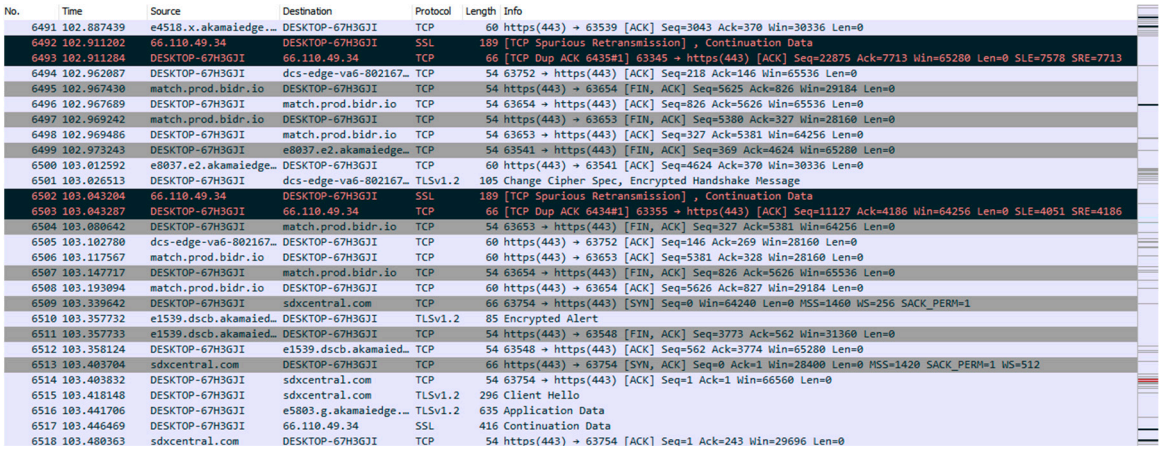
where TN is true negative (the number of data samples from normal traffic data that are correctly classified as normal), TP is true positive (the number of data samples from the botnet traffic data that are correctly classified as botnet), FP is false positive (the number of data samples from the normal traffic data that are wrongly classified as botnet), and FN is false negative (the number of data samples from the botnet traffic data that are classified as normal). These metrics will also be used in creation of confusion matrix (also known as error matrix is used to visualize the performance of an algorithm). Its main purpose is to observe whether the two classes are labeled or mislabeled. A descriptive form of this matrix is shown in Table 1.

Table 1. The confusion matrix for botnet detection.

		Actual Class	
		Botnet	**Normal**
Predicted Class	**Botnet**	True Positive (TP)	False Positive (FP)
	Normal	False Negative (FN)	True Negative (TN)

3.5. Live Network Traffic Data Acquisition

In computer forensics, we have two types of data for examinations: (a) real-time/live analysis; and (b) dead/offline/static analysis. The case of live acquisition happens when the investigation is done on a live system. The data is acquired, processed, and investigated while the system is on. However, in the dead analysis, the data have already been acquired and processed later on. So, when we consider real time analysis of data, as soon as the data become available, it is processed to understand and make conclusions. In this case, the investigation is carried out immediately and it is determined whether the network traffic data is normal or botnet. If a malware is detected, the corresponding countermeasures can be taken. Due to quick reaction from the system without significant delay, less infection and damage can happen. While the offline analysis of data can determine the system status after the problems occur. We performed capturing live network traffic data from the University of Central Florida (UCF) campus network and a local residential place. The tool that we used for capturing data is the Wireshark capture engine. The leveraged interface is Wi-Fi with a buffer size of 2 MB. Also, the used name resolutions are Resolve MAC Addresses, network names, and transport names. A new PCAP file is created every 30 min. A snapshot of the acquired network traffic data is shown in Figure 5. The UCF network traffic data are not shown due to restrictions.



No.	Time	Source	Destination	Protocol	Length	Info
6491	102.887439	e4518.x.akamaiedge.…	DESKTOP-67H3GJI	TCP	60	https(443) → 63539 [ACK] Seq=3043 Ack=370 Win=30336 Len=0
6492	102.911202	66.110.49.34	DESKTOP-67H3GJI	SSL	189	[TCP Spurious Retransmission] , Continuation Data
6493	102.911284	DESKTOP-67H3GJI	66.110.49.34	TCP	66	[TCP Dup ACK 6435#1] 63345 → https(443) [ACK] Seq=22875 Ack=7713 Win=65280 Len=0 SLE=7578 SRE=7713
6494	102.962087	DESKTOP-67H3GJI	dcs-edge-va6-802167…	TCP	54	63752 → https(443) [ACK] Seq=218 Ack=146 Win=65536 Len=0
6495	102.967430	match.prod.bidr.io	DESKTOP-67H3GJI	TCP	54	https(443) → 63654 [FIN, ACK] Seq=5625 Ack=826 Win=29184 Len=0
6496	102.967689	DESKTOP-67H3GJI	match.prod.bidr.io	TCP	54	63654 → https(443) [ACK] Seq=826 Ack=5626 Win=65536 Len=0
6497	102.969242	match.prod.bidr.io	DESKTOP-67H3GJI	TCP	54	https(443) → 63653 [FIN, ACK] Seq=5380 Ack=327 Win=28160 Len=0
6498	102.969486	DESKTOP-67H3GJI	match.prod.bidr.io	TCP	54	63653 → https(443) [ACK] Seq=327 Ack=5381 Win=64256 Len=0
6499	102.973243	DESKTOP-67H3GJI	e8037.e2.akamaiedge…	TCP	54	63541 → https(443) [FIN, ACK] Seq=369 Ack=4624 Win=65280 Len=0
6500	103.012592	e8037.e2.akamaiedge…	DESKTOP-67H3GJI	TCP	60	https(443) → 63541 [ACK] Seq=4624 Ack=370 Win=30336 Len=0
6501	103.026513	DESKTOP-67H3GJI	dcs-edge-va6-802167…	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
6502	103.043204	66.110.49.34	DESKTOP-67H3GJI	SSL	189	[TCP Spurious Retransmission] , Continuation Data
6503	103.043287	DESKTOP-67H3GJI	66.110.49.34	TCP	66	[TCP Dup ACK 6434#1] 63355 → https(443) [ACK] Seq=11127 Ack=4186 Win=64256 Len=0 SLE=4051 SRE=4186
6504	103.080642	DESKTOP-67H3GJI	match.prod.bidr.io	TCP	54	63653 → https(443) [FIN, ACK] Seq=327 Ack=5381 Win=64256 Len=0
6505	103.102780	dcs-edge-va6-802167…	DESKTOP-67H3GJI	TCP	60	https(443) → 63752 [ACK] Seq=146 Ack=269 Win=28160 Len=0
6506	103.117567	match.prod.bidr.io	DESKTOP-67H3GJI	TCP	60	https(443) → 63653 [ACK] Seq=5381 Ack=328 Win=28160 Len=0
6507	103.147717	DESKTOP-67H3GJI	match.prod.bidr.io	TCP	54	63654 → https(443) [FIN, ACK] Seq=826 Ack=5626 Win=65536 Len=0
6508	103.193094	match.prod.bidr.io	DESKTOP-67H3GJI	TCP	60	https(443) → 63654 [ACK] Seq=5626 Ack=827 Win=29184 Len=0
6509	103.339642	DESKTOP-67H3GJI	sdxcentral.com	TCP	66	63754 → https(443) [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
6510	103.357732	e1539.dscb.akamaied…	DESKTOP-67H3GJI	TLSv1.2	85	Encrypted Alert
6511	103.357733	e1539.dscb.akamaied…	DESKTOP-67H3GJI	TCP	54	https(443) → 63548 [FIN, ACK] Seq=3773 Ack=562 Win=31360 Len=0
6512	103.358124	DESKTOP-67H3GJI	e1539.dscb.akamaied…	TCP	54	63548 → https(443) [ACK] Seq=562 Ack=3774 Win=65280 Len=0
6513	103.403704	sdxcentral.com	DESKTOP-67H3GJI	TCP	66	https(443) → 63754 [SYN, ACK] Seq=0 Ack=1 Win=28400 Len=0 MSS=1420 SACK_PERM=1 WS=512
6514	103.403832	DESKTOP-67H3GJI	sdxcentral.com	TCP	54	63754 → https(443) [ACK] Seq=1 Ack=1 Win=66560 Len=0
6515	103.418148	DESKTOP-67H3GJI	sdxcentral.com	TLSv1.2	296	Client Hello
6516	103.441706	DESKTOP-67H3GJI	e5803.g.akamaiedge.…	TLSv1.2	635	Application Data
6517	103.446469	DESKTOP-67H3GJI	66.110.49.34	SSL	416	Continuation Data
6518	103.480363	sdxcentral.com	DESKTOP-67H3GJI	TCP	54	https(443) → 63754 [ACK] Seq=1 Ack=243 Win=29696 Len=0

Figure 5. A snapshot of the captured traffic data from a local residential network.

4. Proposed System and Methodology

The detail of our system for detection of botnet traffic data is proposed in this part. As it can be seen in Figure 6, the flow of system can be split into two main parts: (a) transformation of network traffic data into image and (b) classification of botnet and normal network traffic data. In phase (a), the raw traffic data are partitioned based on the flow. A network flow is defined as all the packets that have the same five-tuple, i.e., source IP, source port, destination IP, destination port, and transport-level protocol. According to this partitioning, a set of raw traffic data are split into a number of subsets. In each subset, the packets are arranged in time-order. Regarding the packet layers to be involved into our analysis, the intrinsic characteristics of the packet layers are reflected into application layer of TCP/IP model, while other layers contain some traffic feature information, which can be used for identification of certain network attacks. In the beginning, a continuous raw traffic is split into multiple discrete traffic units. Next, for elimination of the negative impact of the IP and the MAC address information in a flow, that information is removed through a randomization process, usually called traffic sanitization.

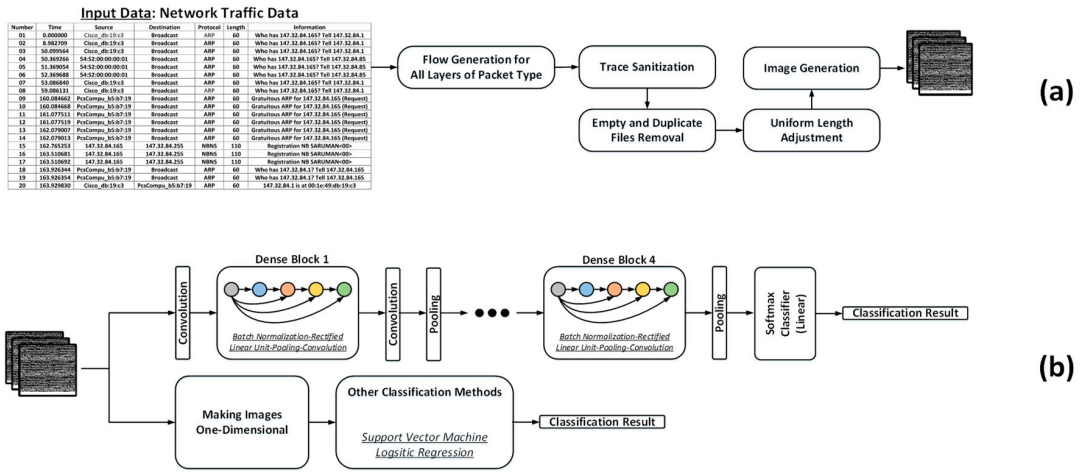


Figure 6. The flow of botnet detection using DenseNet, Support Vector Machine (SVM), and Logistic Regression: (**a**) transformation of network traffic data into image and (**b**) classification of transformed network traffic data into image using DenseNet (top) and SVM along with Logistic Regression (bottom).

According to this procedure, the MAC address and IP address are randomized in the data link layer and IP layer, respectively. After this cleaning process, it is needed to clean files for some packets that have no application layer and accordingly the result bin files will be empty. Also, a number of packets generate the same files when they have the same information and content. This fact leads to having bias during training and they need to be removed as well. After this step, the content of files are read in ASCII-encoded binary representations before being adjusted uniformly in length. After this step, the data is converted into image.

Having done so, the second phase is classification of the network traffic data, which are now images, for determination of their identity. In other words, it is required to determine whether they are botnet or normal. Three types of methods for classification have been selected, namely: (a) CNN-DenseNet (with/without transfer learning); (b) support vector machine (SVM); and (c) logistic regression. For CNN, the images are given directly as input. For transfer learning, the ImageNet weights are loaded and different set of layers of CNN are frozen during training to see their impact on the detection accuracy. In order to input the images to the SVM and the logistic regression, they are transformed into a one-dimensional vector. The flow of our system is shown algorithmically in Algorithm 1.

Algorithm 1: The Flow of Botnet Detection

01: **Input Parameters:** Network Traffic Data
02: **Output Parameters:** Normal or Botnet Traffic Data Classification Outcome
03: Data Acquisition:
04: *Data (in PCAP format) ← Acquiring normal/botnet network traffic data*
05: Data Preprocessing:
06: *Data ← All Layer Flow Generation (Data)*
07: *Data ← Trace Sanitization (Data)*
08: *Data ← Empty and Duplicate File Removal (Data)*
09: *Data ← Uniform Length Adjustment (Data)*
10: *Image ← Image Generation (Data)*
11: Botnet Detection:
12: *Fine-Tuned DenseNet Model. ← Fine-Tuning Pre-trained DenseNet on ImageNet Dataset (Training Images)*
13: *Trained SVM Model. ← SVM (Flattened Training Images)*
14: *Trained Logistic Regression Model. ← Logistic Regression (Flattened Training Images)*
15: *Made Decision from DenseNet ← Fine-Tuned DenseNet Model. (Testing Images)*
16: *Made Decision from SVM ← Trained SVM Model. (Flattened Testing Images)*
17: *Made Decision from Logistic Regression ← Trained Logistic Regression Model. (Flattened Testing Images)*

5. Experimental Results and Evaluation

In this part, we present our experiments with the systematic approaches described above to see their capabilities in botnet detection. The system is implemented using Python programming language and Keras as a backend. The dataset used in our experiments is the CTU-13 dataset, which is a dataset with given labels for botnet, normal and background traffic. We only employed the botnet and the normal traffic data from this dataset. All the data are transformed into images according to the discussed technique, letting us analyze the influence of transfer learning in botnet detection using the obtained images. In this regard, no weights are given to the DenseNet in the first run (which means no transfer learning). Afterwards, the beginning of the first, the second, the third, and the fourth dense block are selected as the starting point for fine-tuning (or retraining) the CNN. This means all the layers before the starting point are fixed (or frozen) and their transferred weights remain unchanged. The ImageNet weights are given for this experiment. In all of these experiments, the system is trained for three epochs. The evaluation results of these experiments are shown in Figure 7. As it can be seen from the figure, the best accuracy (99.98%) is achieved when 10 number of layers are fixed. Also, the demonstrated accuracy without transfer learning is noticeably low comparing to its TL-based counterparts.

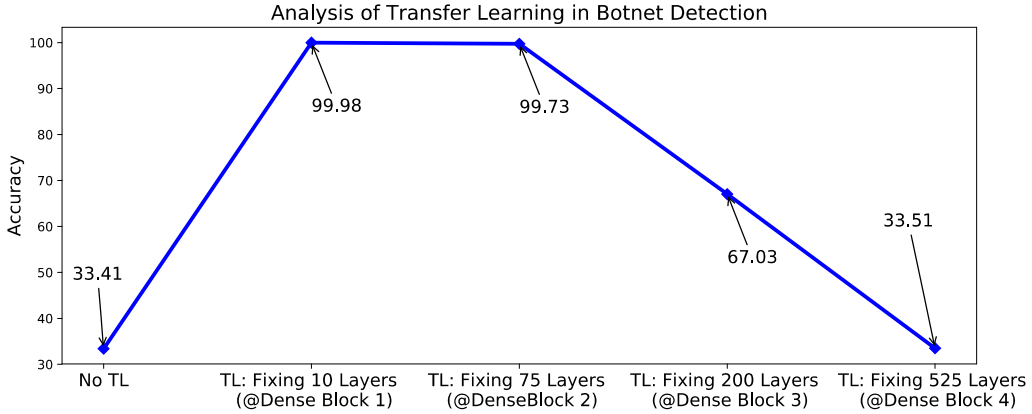


Figure 7. Results of using transfer learning in DenseNet for botnet detection.

Additionally, we tested our system (fine-tuned using the CTU-13 dataset) on another solely botnet dataset, named the Information security and object technology (ISOT) Hypertext Transfer Protocol (HTTP) Botnet Dataset (only the botnet portion) [27]. The results show that they system is strong enough to detect botnet traffic data with accuracy of 100%. Although this result may slightly be different in other runs of the code. In order to make our analysis more comprehensive, we captured live traffic data from the UCF network and a local residential place. This live data can show the performance of our system for real-world applications. The tested system achieves an accuracy of 99.35% that is promising to employ this engine in the IoT and wearable devices. In order to further assess the capability of our proposed system, we compare our results with the reported results from a number of recent published works in the area of botnet detection. This comparison is provided in Table 2. As it can be seen from the shown table, the closest counterparts to our method in terms of accuracy are [28] and clustering in [29]. However, they are less effective in the other parameters.

Table 2. The comparison of classification results among densenet, SVM, and logistic regression.

Score Merit	Accuracy (%)	Area under Curve (%)	Precision (%)	Recall (%)	F-1 Score (%)	Kappa (%)
CNN: DenseNet (Best Result) (for CTU-13 Dataset)	**99.98**	**99.99**	**99.98**	**99.98**	**99.98**	**99.96**
Support Vector Machine (SVM) (for CTU-13 Dataset)	83.15	75.80	80.73	98.07	88.56	57.72
Logistic Regression (for CTU-13 Dataset)	78.56	81.06	92.76	73.49	82.01	56.35
CNN: DenseNet (Best Result) (for Live Normal Network Traffic Data)	**99.35**	**99.98**	**99.35**	**99.35**	**99.35**	**98.70**
CNN: DenseNet (Best Result) (for ISOT HTTP Botnet Dataset, Only Botnet Traffic Data)	**100**	**99.99**	**100**	**100**	**100**	**100**
[28]	94.70	N/A	N/A	N/A	N/A	N/A
[30]: K-Nearest Neighbor (KNN) (for T1 training set)	90.20	N/A	89.50	91.00	90.30	N/A
[30]: C4.5 (for T1 training set)	90.10	N/A	89.10	91.20	90.10	N/A
[30]: Random Forest (RF) (for T1 training set)	90.80	N/A	90.70	91.00	90.80	N/A
[30]: Naïve Bayes (for T1 training set)	85.90	N/A	83.10	90.20	86.50	N/A
[30]: Clustering	98.39	N/A	86.45	84.47	85.45	N/A
[30]: Neural Network (NN)	89.38	N/A	92.50	85.70	88.97	N/A
[30]: Recurrent Neural Network (RNN)	83.09	N/A	95.41	69.53	80.44	N/A

We also performed a timing analysis on the system operation during training and testing of the pre-processed network traffic data (or the images) when the CTU-13 dataset is used. There are 403,004 number of images during training and 47,759 number of images during testing. The system execution time is around 283.22 min during training (for three epochs) and 4.65 min during testing. If we divide the execution time by the number of images, then the time it takes to process each image is 0.702772 milliseconds during training for three epochs. With considering the number of epochs, then the time is 0.234257 for one epoch. On the other side, we have the execution time of 0.0973638 milliseconds for each image during testing. Each flow has an average number of packets of 20 [31,32]. Considering this value, the time to process each packet during training and testing are 0.011713 and 0.004868 milliseconds, respectively. By taking a look at the testing time of each packet, we can say integrating even the current system in some IoT devices is not out of expectation. Figure 8 visualizes the training and the testing time of the botnet detection system.

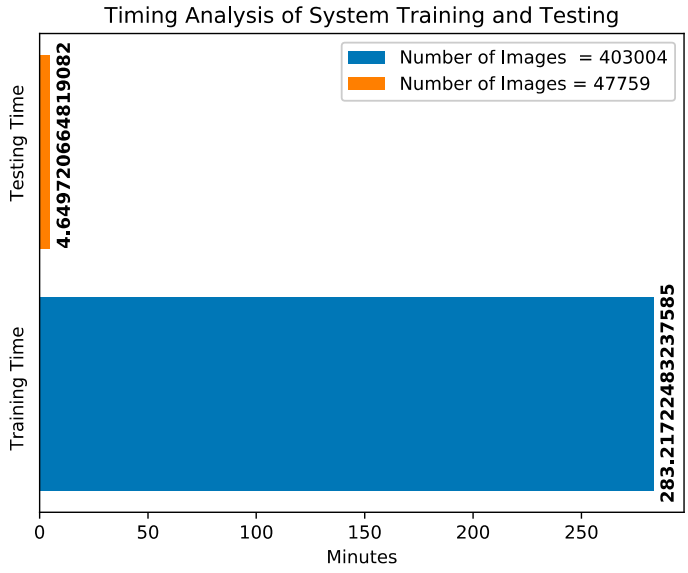
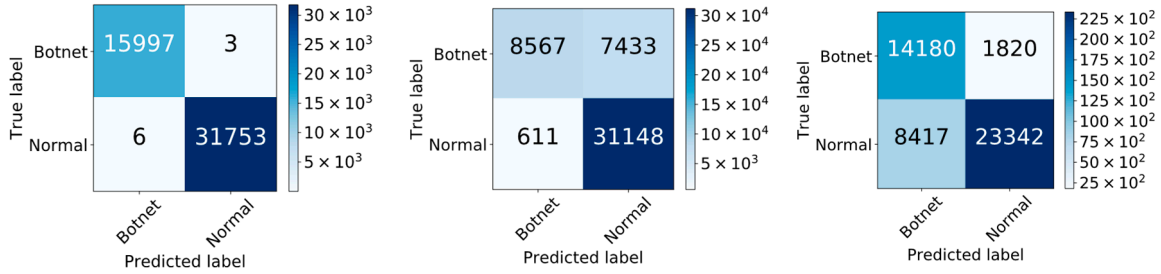


Figure 8. The execution time of the botnet detection system during training and testing.

Another analysis that has been done regarding the comparison between DenseNet (in TL when 10 layers are fixed), SVM, and Logistic Regression are based on their confusion matrices. As it can be seen from Figure 9, the highest number of true positives and true negatives are achieved when DenseNet is used. It means the normal traffic data are correctly classified as normal and reach to their destinations and the botnet traffic data are correctly classified as botnet and blocked from their targeted points. The SVM shows better performance against Logistic Regression in terms of true negative, while the logistic regression shows better performance in terms of true positives. On the other hand, the SVM caters higher false positives, while the logistic regression brings more false negatives. The interpretation of this action is having more unauthorized access when SVM is used and more denial of access (service) when logistic regression is used.



(**a**) DenseNet (**b**) SVM (**c**) Logistic Regression

Figure 9. Analysis of confusion matrix of DenseNet (**a**), SVM (**b**), and Logistic Regression (**c**) in classification of botnet and normal traffic data.

Discussion, Limitations, and Lessons Learned

The possible limitations of this system and its corresponding solutions are discussed here. When a system is built using a deep learning engine, it may contain certain limitations, such as being data hungry, having a power constraint, being complex, and being unable to deliver real-time response. These requirements are mostly referenced during the training and can be overcome by incorporating sufficient performance hardware. In other words, once the system is trained for certain number epochs and input data, then it only needs to be tested. The testing time is much less than the training time. In addition, there are many real world applications of deep learning such as Image Recognition, Speech Recognition, Medical Diagnosis, Statistical Arbitrage, Learning Associations, Classification, Extraction, and Regression [33]. The same underlying techniques used to make them suitable them for real world applications can be used for botnet detection as well. As an example, many smart phones nowadays are capable of face recognition [34,35] and equipped with AI learning and testing tasks. Furthermore, there are a number of related works published: DeepX has been presented in [36,37] which is a software accelerator for deep learning execution. Using this interface, the device resources are lowered significantly (including memory, computation, and energy) to overcome the bottleneck of mounting deep learning engines on mobile devices. A technique based on edge computing has been proposed in [38] for improving the performance of deep learning applications.

What we learned from design, development, and testing of this deep learning-based botnet detection engine is that visualization of the network traffic data provides certain informative features about the ongoing packets in the network, which may not be easily seen by the one-dimensional numerical format of the traffic data. Also, the traditional machine learning classifiers are easily defeated by the neural networks. It means that the assumptions of linearity for the decision by logistic regression and considering a hyper-plane for differentiation of the images are not as effective as the learning layers and the decision layer(s) of neural networks. Another lesson is the importance of having sufficient number of normal and botnet traffic data during training to build a high performance system. Also, we realize that it is important to check the strength of our system in front of live data since it shows how it behaves in real circumstances. The value of transferring knowledge from a different

domain into the security context is another aspect of this work that should not be neglected. Also, it has been realized that a deep neural network can be more effective than shallow neural networks when we use dataset weights. Examining the definitions of the metrics, accuracy (as a measure of errors of the system, how it biases toward categories), precision (as the fraction of retrieved network traffic data that are related to the query), recall (as the fractional of the related network traffic data that are successfully retrieved), F1 score (as a combination of the precision and recall based on its mean), and Kappa (as a measure of inter-rater agreement for qualitative items), we can interpretate the following: a deep neural network with transferred knowledge surpasses a simple neural network and traditional machine learning engines (like SVM, logistic regression, KNN, RF, C4.5, Naïve Bayes, Clustering) in having the least system errors and being biased correctly toward the targeted categories, and can deliver the highest performance in correctly retrieving related network traffic data. Transferring knowledge from a different domain can help in improving the system performance, but the most optimal layer needs to be chosen as well. According to this lesson, a specific number of layers should be fixed before starting the fine tuning process. At last, the speed of the botnet detection engine matters the most when it is employed in the IoT and wearable devices. This speed can be improved by using hardware accelerators and software algorithms.

6. Conclusions and Future Work

In this paper, we propose a deep learning-based botnet detection engine that gets raw network traffic data as input and transforms them into images. The images are input into a deep CNN, DenseNet for classification of normal and botnet traffic data. Also, the idea of transfer learning has been leveraged to evaluate the effectiveness of loading weights from a previously trained model on a different dataset. The experimental results show using transfer learning in classification of image representation of network traffic data can lead to getting accuracy up to 99.98%, Area Under Curve (AUC) up to 99.99%, precision up to 99.98%, recall up to 99.98%, F-1 score up to 99.98%, and kappa up to 99.96%. Also, the obtained true positives and true negatives are 47,750 out of 47,759. Instead of DenseNet, two other classifiers, SVM and logistic regression have been examined. They demonstrated security vulnerabilities of unauthorized access and denial of access respectively. Overall, we believe that this work is a contribution to the communities of IoT security. In future work, it has been planned to extend the ideas of one-dimensional data to image conversion and transfer learning for detection of other security vulnerabilities, namely Spyware, Phishing, and Spam in the IoT world. Also, we will use different types of neural networks for a comparative study.

Author Contributions: S.T. and M.S. came up with the ideas, ran the experiments. S.T. wrote the manuscript. J.-S.Y. provided technical feedback and revised the manuscript. All authors proofread the final manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Tsiropoulou, E.E.; Baras, J.S.; Papavassiliou, S.; Qu, G. On the Mitigation of Interference Imposed by Intruders in Passive RFID Networks. In *International Conference on Decision and Game Theory for Security*; Springer: Cham, Switzerland, 2016; pp. 62–80.
2. Sagduyu, Y.E.; Ephremides, A. A game-theoretic analysis of denial of service attacks in wireless random access. *Wirel. Netw.* **2009**, *15*, 651–666. [CrossRef]
3. Ji, Y.; Li, Q.; He, Y.; Guo, D. BotCatch: Leveraging signature and behavior for bot detection. *Secur. Commun. Netw.* **2015**, *8*, 952–969. [CrossRef]
4. Göbel, J.; Holz, T. Rishi: Identify Bot Contaminated Hosts by IRC Nickname Evaluation. *HotBots* **2007**, *7*, 8.
5. Wurzinger, P.; Bilge, L.; Holz, T.; Goebel, J.; Kruegel, C.; Kirda, E. Automatically generating models for botnet detection. In *European Symposium on Research in Computer Security*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 232–249.

6. Live Cyber Attack Threat Map. Available online: https://threatmap.checkpoint.com/ThreatPortal/livemap.html (accessed on 14 October 2018).
7. Gu, G.; Porras, P.A.; Yegneswaran, V.; Fong, M.W.; Lee, W. Bothunter: Detecting malware infection through ids-driven dialog correlation. In Proceedings of the USENIX Security Symposium, Boston, MA, USA, 6–10 August 2007; Volume 7, pp. 1–16.
8. Gu, G.; Zhang, J.; Lee, W. BotSniffer: Detecting botnet command and control channels in network traffic. In Proceedings of the Network and Distributed System Security Symposium, NDSS 2008, San Diego, California, USA, 10–13 February 2008.
9. Siboni, S.; Cohen, A. Botnet identification via universal anomaly detection. In Proceedings of the 2014 IEEE International Workshop on Information Forensics and Security (WIFS), Atlanta, GA, USA, 3–5 December 2014; pp. 101–106.
10. Sakib, M.N.; Huang, C. Using anomaly detection based techniques to detect HTTP-based botnet C&C traffic. In Proceedings of the 2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia, 23–27 May 2016; pp. 1–6.
11. Manasrah, A.M.; Hasan, A.; Abouabdalla, O.A.; Ramadass, S. Detecting botnet activities based on abnormal DNS traffic. *arXiv*, 2009; arXiv:0911.0487.
12. Kwon, J.; Lee, J.; Lee, H.; Perrig, A. PsyBoG: A scalable botnet detection method for large-scale DNS traffic. *Comput. Netw.* **2016**, *97*, 48–73. [CrossRef]
13. Nguyen, T.-D.; Cao, Tu.; Nguyen, Li. DGA botnet detection using collaborative filtering and density-based clustering. In Proceedings of the Sixth International Symposium on Information and Communication Technology, Hue, Vietnam, 3–4 December 2015; pp. 203–209.
14. Kirubavathi, G.; Anitha, R. Botnet detection via mining of traffic flow characteristics. *Comput. Electr. Eng.* **2016**, *50*, 91–101. [CrossRef]
15. Beigi, E.B.; Jazi, H.H.; Stakhanova, N.; Ghorbani, A.A. Towards effective feature selection in machine learning-based botnet detection approaches. In Proceedings of the 2014 IEEE Conference on Communications and Network Security (CNS), San Francisco, CA, USA, 29–31 October 2014; pp. 247–255.
16. McDermott, C.D.; Majdani, F.; Petrovski, A.V. Botnet detection in the internet of things using deep learning approaches. In Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.
17. Gu, G.; Perdisci, R.; Zhang, J.; Lee, W. Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection. In Proceedings of the 17th USENIX Security Symposium, San Jose, CA, USA, 28 July–1 August 2008.
18. Zhao, D.; Traore, I.; Sayed, B.; Lu, W.; Saad, S.; Ghorbani, A.; Garant, D. Botnet detection based on traffic behavior analysis and flow intervals. *Comput. Secur.* **2013**, *39*, 2–16. [CrossRef]
19. Prokofiev, A.O.; Smirnova, Y.S.; Surov, V.A. A method to detect internet of things botnets. In Proceedings of the 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), Moscow, Russia, 29 January–; pp. 105–108.
20. Yu, T.; Sekar, V.; Seshan, S.; Agarwal, Y.; Xu, C. Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the internet-of-things. In Proceedings of the 14th ACM Workshop on Hot Topics in Networks, Philadelphia, PA, USA, 16–17 November 2015; p. 5.
21. Saad, S.; Traore, I.; Ghorbani, A.; Sayed, B.; Zhao, D.; Lu, W.; Felix, J.; Hakimian, P. Detecting P2P botnets through network behavior analysis and machine learning. In Proceedings of the 2011 Ninth Annual International Conference on Privacy, Security and Trust (PST), Montreal, QC, Canada, 19–21 July 2011; pp. 174–180.
22. Masud, M.M.; Al-Khateeb, T.; Khan, L.; Thuraisingham, B.; Hamlen, K.W. Flow-based identification of botnet traffic by mining multiple log files. In Proceedings of the First International Conference on Distributed Framework and Applications, Penang, Malaysia, 21–22 October 2008; pp. 200–206.
23. Liao, W.; Horng, Yi. Business strategy for the telecommunication industry in digital convergence. In Proceedings of the 2010 International Conference on Internet Technology and Applications, Wuhan, China, 20–22 August 2010.
24. Venkatesh, G.K.; Nadarajan, R.A. HTTP botnet detection using adaptive learning rate multilayer feed-forward neural network. In *IFIP International Workshop on Information Security Theory and Practice*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 38–48.

25. Wang, W.; Zhu, M.; Zeng, X.; Ye, X.; Sheng, Y. Malware traffic classification using convolutional neural network for representation learning. In Proceedings of the 2017 International Conference on Information Networking (ICOIN), Da Nang, Vietnam, 11–13 January 2017; pp. 712–717.
26. Garcia, S.; Grill, M.; Stiborek, J.; Zunino, A. An empirical comparison of botnet detection methods. *Comput. Secur.* **2014**, *45*, 100–123. [CrossRef]
27. Alenazi, A.; Traore, I.; Ganame, K.; Woungang, I. Holistic Model for HTTP Botnet Detection Based on DNS Traffic Analysis. In *Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments*; Lecture Notes in Computer Science; Traore, I., Woungang, I., Awad, A., Eds.; Springer: Cham, Switzerland, 2017; Volume 10618.
28. Chen, S.; Chen, Y.; Tzeng, W. Effective botnet detection through neural networks on convolutional features. In Proceedings of the 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE), New York, NY, USA, 1–3 August 2018; pp. 372–378.
29. Bansal, A.; Mahapatra, S. A comparative analysis of machine learning techniques for botnet detection. In Proceedings of the 10th International Conference on Security of Information and Networks, Jaipur, India, 13–15 October 2017; pp. 91–98.
30. Hoang, X.D.; Nguyen, Q.C. Botnet detection based on machine learning techniques using DNS query data. *Future Int.* **2018**, *10*, 43. [CrossRef]
31. Flow-based characteristic analysis of internet application traffic. Available online: https://www.semanticscholar.org/paper/Flow-based-Characteristic-Analysis-of-Internet-Kim-Won/23bb4a5349bd26992935dd0382dc16fde3897aa8 (accessed on 26 October 2018).
32. Curtis, A.R.; Mogul, J.C.; Tourrilhes, J.; Yalagandula, P.; Sharma, P.; Banerjee, S. DevoFlow: Scaling flow management for high-performance networks. In *ACM SIGCOMM Computer Communication Review*; ACM: New York, NY, USA, 2011; Volume 41, pp. 254–265.
33. Top 9 Machine Learning Applications in Real World. Available online: https://www.datasciencecentral.com/profiles/blogs/top-9-machine-learning-applications-in-real-world (accessed on 14 October 2018).
34. Machine Learning: Real-World Applications. Available online: https://dzone.com/articles/machine-learning-real-world-applications (accessed on 14 October 2018).
35. 10 Best Face Recognition Phones 2018 With Face Unlock. Available online: http://blog.awok.com/10-best-face-recognition-phones-2018-face-unlock/ (accessed on 14 October 2018).
36. Lane, N.D.; Bhattacharya, S.; Georgiev, P.; Forlivesi, C.; Kawsar, F. An early resource characterization of deep learning on wearables, smartphones and internet-of-things devices. In Proceedings of the IoT-App '15 Proceedings of the 2015 International Workshop on Internet of Things towards Applications, Seoul, South Korea, 1 November 2015; pp. 7–12.
37. Lane, N.D.; Bhattacharya, S.; Georgiev, P.; Forlivesi, C.; Jiao, L.; Qendro, L.; Kawsar, F. Deepx: A software accelerator for low-power deep learning inference on mobile devices. In Proceedings of the 15th International Conference on Information Processing in Sensor Networks, Vienna, Austria, 11–14 April 2016; p. 23.
38. Lane, N.D.; Bhattacharya, S.; Georgiev, P.; Forlivesi, C.; Kawsar, F. Accelerated deep learning inference for embedded and wearable devices using DeepX. In Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services Companion, Singapore, 25–30 June 2016; p. 109.