A Multi-Class Neural Network Model for Rapid Detection of IoT Botnet Attacks

Haifaa Alzahrani[1], Maysoon Abulkhair[2], Entisar Alkayal[3]
Information Technology Department
King Abdulaziz University, Jeddah, Saudi Arabia

***Abstract*—The tremendous number of Internet of Things (IoT) devices and their widespread use have made our lives considerably more manageable and safer. At the same time, however, the vulnerability of these innovations means that our day-to-day existence is surrounded by insecure devices, thereby facilitating ways for cybercriminals to launch various attacks by large-scale robot networks (botnets) through IoT. In consideration of these issues, we propose a neural network-based model to detect IoT botnet attacks. Furthermore, the model provides multi-classification, which is necessary for taking appropriate countermeasures to understand and stop the attacks. In addition, it is independent and does not require specific equipment or software to fetch the required features. According to the conducted experiments, the proposed model is accurate and achieves 99.99%, 99.04% as F1 score for two benchmark datasets in addition to fulfilling IoT constraints regarding complexity and speed. It is less complicated in terms of computations, and it provides real-time detection that outperformed the state-of-the-art, achieving a detection time ratio of 1:5 and a ratio of 1:8.**

***Keywords*—Internet of Things (IoT); IoT botnets; IoT security; intrusion detection system; deep learning; neural network**

I. Introduction

The dominant features of the modern era can be illustrated by the abundant data that are collected and monitored via Internet of Things (IoT) devices, as well as by the endless functionalities enabled by this innovation. As estimated by experts [1], the number of IoT devices is expected to reach 30 billion by 2020—an important development given that these widespread and convenient technologies have strongly influenced many aspects of people's lives. At the same time, however, they have also compounded the consequences of security threats. Given the innumerable IoT devices that are constantly running and accessible over the public Internet, such innovations have become an attractive platform for cybercriminals. The hack value of IoT devices is not confined to the critical information stored, collected, or monitored by these technologies but extend to any other assets that can be breached via large-scale botnets. This problem is further exacerbated by the fact that the IoT ecosystem imposes constraints on security techniques because of limited resources with respect to central processing units (CPUs), memory, and power consumption. These shortcomings render the battle against IoT botnets a critical and challenging issue.

An equally significant concern is the higher risk that IoT devices present compared with that arising from general-purpose computers. This threat stems from numerous factors [2]. First, the requirements for IoT applications are extremely heterogeneous in terms of device types, communication protocols, and operating systems. Second, the global distribution of IoT devices translates to monitoring by different parties, thereby preventing the establishment of well-defined perimeters among these overseers. From the involvement of multiple parties comes user and device mobility, which causes continuous changes in perimeters. Third, IoT devices lack strong authentication and authorization mechanisms, as reflected by the tendency of most IoT users to employ weak passwords and default account settings. Devices equipped with IoT technology usually do not require user permission or direct interaction for the installation of software or the modification of settings, thus facilitating malware propagation through application programming interfaces (APIs) and firmware. Finally, vendors experience difficulties in patching software vulnerabilities. As a result, the conventional security techniques developed for general-purpose computers, such as antivirus programs or host-based intrusion detection systems, are inadequate measures for securing IoT.

The threat model included in this study consists of attackers with no physical access to the IoT devices connected to home routers, functioning as network gateways or other middleboxes. The actualization of a threat is described as follows: An attacker needs to exploit the vulnerabilities of different IoT devices to gain access to them, but it must first discover the existence of such devices by sending probes to certain ports. The probes initially pass through a network gateway before reaching the destination. Most IoT communications are executed through cloud API services [3] instead of proceeding directly from one local IoT device to another. In this process, therefore, a network gateway occupies a vantage point from which it can inspect every network packet. The use of this point has been increasingly emphasized in the implementation of different intrusion detection techniques. Furthermore, a network gateway provides a homogeneous and lightweight defensive mechanism and policy enforcer that protects devices from being assimilated into a botnet without interrupting their normal functionality. This study focused only on the detection techniques applicable to network gateways.

Neural networks and deep learning have demonstrated promising outcomes in many fields, especially in developing accurate anomaly-based intrusion detection systems [4]–[7]. Unfortunately, they require high computational use, and it takes a long time to train a model and detect an attack. At the same time, rectifying the problem of IoT botnets necessitates specialized solutions that take into account IoT's own constraints. An adequate number of research projects have been tailored toward the detection and prevention of IoT botnet attacks using machine or deep learning. However, to the best of our knowledge and according to the provided literature review

[8]–[13], we found there were no studies considered the IoTs requirements for real-time detection and lightness while taking into account the multi-classification issue. Although, it is a critical point to recognize the attack type and then take the appropriate countermeasures to prevent any intrusions. Motivated by these issues, our study provides an independent, accurate, real-time, and lightweight model applicable to IoT gateways that is able to multi-classify the IoT network traffic. Therefore, the main contribution of this study is to adapt the fast, accurate, stable, tiny gated recurrent neural network (FastGRNN) [14] algorithm, which is dedicated to text classification, for use with intrusion detection by treating network packets as sentences and headers as words. The objectives of this study were to

- provide a model that has accurate detection of IoT botnets,
- decrease the training time,
- decrease the detection time, and
- decrease the model complexity.

We also took into account that the model is independent and would only consider the features that are directly readable by the gateway and do not require additional equipment or a third party to fetch the features and target multi-classification.

The results proved that using FastGRNN [14] provided high speed for training the model and detecting attacks, with much less complexity compared to the state-of-the-art while also preserving a high F1 score, where it attained a score of 99.04% with the RGU dataset [8] in comparison to the gated recurrent unit (GRU) model's 97.82%, and the long short-term memory (LSTM) model's 98.60%. Furthermore, the FastGRNN-based model completed its detection within 29 seconds for the entire test set for both datasets, while the model proposed by Hwang et al. [9] took 245, 249 seconds for detection.

The rest of the paper is organized as follows. Section II presents a detailed background on IoT botnets, with particular attention paid to how they operate and what destructive effects they exert. The section also introduces the FastGRNN [14] algorithm. Section III consists of a literature review and a comparison of the proposed model and the current state-of-the-art models. Section IV describes the methodology and the propose model. Section V summarizes the results and findings, and Section VI concludes the paper.

II. Background

A. IoT Botnets

A botnet basically consists of compromised devices called bots, each running malicious code under a botmaster's command and control (C&C) [2]. Specifically, a bot can propagate throughout the network. To do so, it scans the entire network ranges and exploits the known vulnerabilities or weak credentials of devices. After breaking into an unprotected gadget, the bot embeds itself into the equipment and waits for instructions from a botmaster to perform malicious activities. An example of these attacks is the collaborative flooding of a target (an IoT or non-IoT device) with numerous illegitimate requests, thus preventing the device from processing legitimate ones and causing a distributed denial-of-service (DDoS) attack. The other ill-intentioned activities of IoT botnets [2] include cryptocurrency mining, password cracking, and email spam sending, keylogging.

Although the first IoT botnet, Linux.Hydra, was discovered in 2008 [15], the security community did not realize the seriousness of this issue until the emergence of the Mirai botnet [16]. In September 2016, a Mirai attack was directed against the Krebs on Security blog, generating 620 Gbps of traffic. The availability of Mirai's original source code led to the development of dozens of variants and inspired the creation of many other botnets. For instance, the following month saw a Mirai variant take down the service provider Dyn, representing the largest DDoS attack in history. This event engendered other destructive outcomes, which were summarized by [17]. Mirai was merely the tip of the iceberg, as predicted by Vlajic and Zhou [18] and indeed we are now witnessing progressively sophisticated IoT botnet attacks with considerably more critical victims. In the same year, Rapidity Networks discovered Hajime, which has a decentralized (or peer-to-peer [P2P]) architecture in contrast to the centralized structure of Mirai [19]. The year 2017 saw a demonstration of BrickerBot's ability to permanently destroy an IoT device through a permanent denial-of-service (PDoS) attack [20], and 2018 witnessed Radware's honeypot capture JenX, which uses servers to scan vulnerable IoT devices and propagates itself within such equipment. The centralized scanning mechanism of JenX enables attackers to offer botnet-for-hire and DDoS-for-hire services [21]. Other attacks were explored by [18] and [22], who inquired into potential attacks by employing IoT as a reflector of DDoS attacks, which are very difficult to trace. Adding to our understanding of cyberattacks, Soltan et al. [23] examined a possible attack in which a botnet utilizes high-wattage IoT devices to manipulate demand and thus disrupt power grid operations. Scrutinizing the distinctive behaviors of IoT botnets plays a crucial role in endeavors to combat them. Generally, the lifecycle of an IoT botnet consists of two main phases, namely, the botnet establishment and attack launch phases (Fig. 1). These stages are described below.

1. Botnet establishment
   1.1. A bot (or malware code) implements scanning and reconnaissance to find a vulnerable device. For example, Mirai sends fingerprintable scan packets to pseudo-random IPv4 addresses to identify devices accessible via Telnet (port 23 or 2323) [17]. All communications pass through a gateway.
   1.2. The bot compromises its victim by exploiting weak credentials through brute force or the exploitation of the known vulnerabilities of IoT devices or routers [15].
   1.3. On the basis of the victim's characteristics (as the processor's architecture), a compatible version of the bot is installed and executed. Sometimes, the bot may remove any other malware and rebind ports to itself to prevent any other potential malware from attacking the victimized device.
2. Attack launch
   2.1. The attacker initiates the attack command via a (C&C) server, which then relays the required information to distributed bots. As previously stated, some botnets use P2P architecture.
   2.2. The bots begin the attack after receiving the corre-

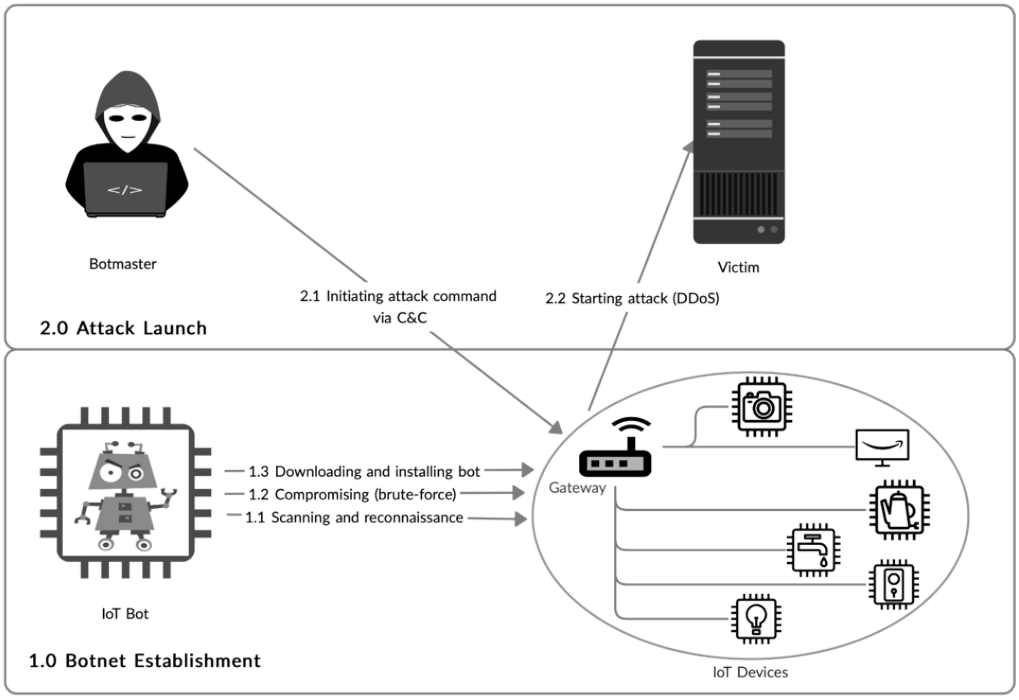

Fig. 1. The Lifecycle of IoT Botnet.

sponding commands. The attack ranges from PDoS and DDoS attacks to cryptocurrency mining and so on.

B. FastGRNN Algorithm

A recurrent neural network (RNN) is a class of neural network proposed by Jeffrey Elman in 1990 [24]. RNNs have the ability to preserve learned information from the past (or previous output) and modify it regularly with current input. This is done via a structure called hidden states, which are updated using different mechanisms or gates. A gate is simply a sigmoid neural net layer and a matrix multiplication. This ability to preserve historical data has meant that RNNs are well suited for the tasks of processing time series or sequence data, such as with neural language processing (NLP).

However, traditional RNN is prone to a vanishing gradient problem that arises when long input sequences are processed, which is the problem that LSTM [25], a different algorithm from the RNN class, was designed to resolve. The complexity of LSTM and its number of computations led to the GRU [26] which is less complicated because it has only two gates instead of the three in LSTM. Basically, GRU merges two gates, the forget and input gates, into an updated gate. In addition, it combines the cell state from LSTM with a hidden state.

FastGRNN goes further in decreasing model complexity and speeding up the learning process by adding a scalar weighted residual connection for each and every coordinate of the hidden state $\mathbf{h_t}$. As shown in Fig. 2, FastGRNN reuses the low-rank, sparse, and quantized matrices $\mathbf{W} \in \mathbf{R}^{\hat{\mathbf{D}} \times \mathbf{D}}$, and $\mathbf{U} \in \mathbf{R}^{\hat{\mathbf{D}} \times \hat{\mathbf{D}}}$ for the vector-valued gating function as well.

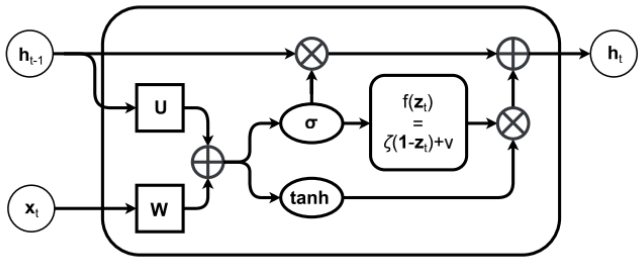

Fig. 2. FastGRNN Cell [14].

In other words, instead of directly feeding the input $\mathbf{x_t}$ and previous hidden state $\mathbf{h_{t-1}}$ into the gates or nonlinear function, these matrices squeeze those values into smaller size before passing them to the sigmoid $\boldsymbol{\sigma}$ or $\mathbf{tanh}$ function.

The learning process starts when $\mathbf{W}$ is added to $\mathbf{U}$, and the result flows into sigmoid $\boldsymbol{\sigma}$ and $\mathbf{tanh}$, resulting in $\mathbf{z_t}$ according to Equation 1 and $\mathbf{h_t}$ according to Equation 2.

$$\mathbf{z_t} = \boldsymbol{\sigma}\left(\mathbf{W}\mathbf{x_t} + \mathbf{U}\mathbf{h_{t-1}} + \mathbf{b_z}\right) \tag{1}$$

$$\tilde{\mathbf{h}}_t = \mathbf{tanh}\left(\mathbf{W}\mathbf{x_t} + \mathbf{U}\mathbf{h_{t-1}} + \mathbf{b_h}\right) \tag{2}$$

The outputs of both functions are used to calculate the final hidden state $\mathbf{h_t}$, as shown in Equation 3. Notably, $\mathbf{0} \leq \boldsymbol{\zeta}, \boldsymbol{\nu} \leq \mathbf{1}$ are trainable parameters the sigmoid function parameterizes along with $\mathbf{b} \in \mathbf{R}^{\hat{\mathbf{D}}}$.

$$\mathbf{h_t} = \left(\boldsymbol{\zeta}\left(\mathbf{1} - \mathbf{z_t}\right) + \boldsymbol{\nu}\right) \odot \tilde{\mathbf{h}}_t + \mathbf{z_t} \odot \mathbf{h_{t-1}} \tag{3}$$

III. Related Work

Anomaly detection involves the adoption of various machine or deep learning algorithms. It centers on building a model of normal behavior for a device and then leveraging the model to detect outliers that could exhibit potential attacks. Undoubtedly, deciding on appropriate features will affect the model's speed and complexity and leverage strong results in the development of considerably reliable learning models. On this basis, relevant studies were reviewed to highlight the features and how they are selected. Each study was analyzed with regard to the following criteria: detection method, whether it is multi-classification or binary, whether it is independent or dependent, whether it is real-time or offline, and whether it is lightweight or not. That information was then used as a reference in drawing the contributions of this paper.

IoTGUARD [10] observes diverse traffic types, including malicious and benign traffic, regardless of the source of flow; such traffic are fundamentally the dataset features collected from a gateway and each device log. A dataset is subjected to preprocessing steps, including oversampling and undersampling for the resolution of imbalance issues, feature extraction, analysis, and reduction techniques. Subsequently, fuzzy c-means (FCM) is used to cluster data according to self-similarities. The principal property of FCM is its ability to maintain a strong association within a cluster and weak associations with all other clusters. Weak associations facilitate task prediction because of their consideration of all clusters in determining labels for new, unknown traffic. A fuzzy interpolation scheme is then employed to ascertain the degree of malice in an attack and accordingly determine appropriate measures for various malicious traffic types. IoTGUARD has been evaluated using a dataset collected from consumer IoT devices. Aside from encompassing normal traffic, the dataset includes information on authentication attacks, botnet activities, port sweeps, port scans, spying, and worms. It achieves a high prediction accuracy with low false-positive-rate. Its operation makes minimal demands on systems because it undergoes preprocessing and reduction. However, IoTGUARD depends on features that are not directly readable and extracted via a gateway.

Concentrating on generating more relative features, Moustafa et al. [11] proposed the use of statistical features in conjunction with an ensemble method to classify IoT network traffic. To derive the features, the authors used the Bro-IDS tool [27], and to acquire specific features, they employed a novel extractor. These features consist of flow-based, Message Queuing Telemetry Transport (MQTT), and service-based characteristics, which consist of DNS and HyperText Transfer Protocol (HTTP) features. Then, the authors applied the correntropy measure to evaluate the feature set. The most important features were selected, and the unnecessary ones were eliminated on the basis of correlation coefficients (CCs). According to the correntropy results, the difference between normal and attack vectors was very small, thereby giving rise to the need to use many classification techniques, each designed on a particular kernel, like a probability, weight or feature value. Given this issue and the need to increase the accuracy of detection, an ensemble method was used along with three classification techniques: a decision tree (DT), Naïve Bayes, and artificial neural networks (ANNs). Afterwards, AdaBoost was employed to distribute network data among the techniques. The ensemble method outperformed every individual approach in terms of accuracy and detection over two benchmark datasets, namely, UNSW-NB15 [28] and NIMS [29]. However, it required more time in detecting an attack than that needed by each individual classifier, except for ANNs. Similar to IoTGUARD [10], the ensemble method is typified by statistical features that are not directly readable by the gateway and whose extraction requires another party—deficiencies that disqualify this approach as a means of online detection.

In contrast to IoTGUARD [10] and the ensemble method established by Moustafa et al. [11], the technique proposed by Doshi et al. [12] examines only flow-based features that are directly readable by most modern gateways. The dataset that comes with the approach includes classes of DoS attacks that might be generated by a Mirai-infected device; examples of such assaults are transmission control protocol synchronize (SYN) flooding, a user datagram protocol (UDP) flood, and an HTTP GET flood. The main contribution of Doshi et al.'s [12] work is feature engineering, which guides the feature extraction process. Selected features were either found in each packet's header or generated in flows from different packets. After this, evaluation was directed toward binary classification algorithms from among the following list: K-nearest neighbors (KNN), random forest, DT, support vector machines with linear kernel (LSVM), and deep neural networks (DNN). All the algorithms performed excellently, achieving an accuracy of 99%, with the exception of the LSVM, which exhibited the worst performance, possibly because the data could not be separated in a linear manner. The study also confirmed the effectiveness of neural networks despite their use with a small dataset that consisted of only 491,855 packets. The selected features are common among all protocols, indicating that Doshi et al.'s [12] proposed method is a protocol-independent technique. It supports low memory constraints because it depends on a stateless algorithm, but the accompanying dataset reflected only one phase of Mirai propagation, that is, the launch of a DoS attack. The dataset was also imbalanced, containing 459,565 malicious packets and only 32,290 benign packets, potentially adversely affecting the results.

Given that flow-based approaches suffer significant detection delay, other researchers proposed to replace flow features with packet features. For instance, Pulse's dataset [13] comprises only the attack time, the destination IP address, the protocols used, and the packets size, as well as labels that indicate malicious or benign traffic. It is a Naïve Bayes classifier that focuses on botnets' primary behaviors, specifically network scanning, network probing, and DoS. The model was built using Weka [30], which in turn, imports the dataset collected from a testbed equipped with real IoT devices. The model is better at detecting probing attacks than it is for flood-type attacks, which might be due to insufficient feature vectors. The authors [13] chose Naïve Bayes because it outperforms other methods, but they did not specify which approaches were compared and what the results were.

In like manner, McDermott et al. [8] introduced a novel approach wherein word embedding is applied on texts in network packets and fed into a bidirectional long short-term memory-based recurrent neural network (BLSTM-RNN). The main advantage of BLSTM-RNN over LSTM-RNN is its

TABLE I. THE SUMMARY OF IOT BOTNETS DETECTION TECHNIQUES.

Work	Method	Multi-classification	Independent	Real-time	Lightweight
[10]	Fuzzy C-means (FCM) clustering	✔	✘	✘	✘
[11]	Decision tree (DT), Naïve Bayes, Artificial neural networks (ANNs), and AdaBoost	✘	✘	✘	✘
[12]	K-nearest neighbor (KNN), random forest, DT Support vector machines with linear kernel (LSVM), and deep neural networks (DNN)	✘	✔	✘	✘
[13]	Naïve Bayes	✘	✔	✘	✘
[8]	Bidirectional long short-term memory (BiLSTM)	✔	✔	✘	✘
[9]	Long short-term memory (LSTM)	✘	✔	✘	✘
Proposed model	Fast, accurate, stable, and tiny gated recurrent neural network (FastGRNN)	✔	✔	✔	✔

ability to accumulate contextual information from both the past and future. The framework consists of three modules. First, data preprocessing is completed on network packets to extract length, protocol, and payload information within the info field, after which word embedding is implemented on each token and encoded into an integer format. Next, packets are normalized and unnecessary ones are removed. Second, LSTM-RNN and BLSTM-RNN models are defined and evaluated, and third, a test dataset is used to determine the effectiveness of anomaly detection. The authors also provided the dataset named Mirai-RGU, which was generated using Mirai and IoT cameras. The traffic consisted of Mirai messages between a bot (an infected IoT) with a C&C. Additionally, four attack vectors were chosen, including User Datagram Protocol (UDP) flood, Acknowledgment (ACK) flood, DNS flood, and SYN flood attacks, as well as normal traffic generated by the cameras. A couple of experiments indicated that the accuracy and loss metrics exhibited by LSTMN-RNN and BLSTM-RNN were close but favor the latter. Nevertheless, the bidirectional model added to the overhead and increased processing time.

Similar to [13] and [8], Hwang et al. [9] eliminated the time required for accumulating network packets to generate flow-based features by directing attention exclusively to the headers of individual packets. At the same time, the authors avoided the high cost of deep-packet inspection required by [8]. The central advantage here is that packet header fields are directly readable by gateways once they arrive, thus facilitating real-time detection. The authors put forward the application of word embedding on an incoming network packet to extract its semantic meanings, then adjusting three layers of LSTM to classify the packet as normal or malicious. To evaluate the model, a dataset called Mirai-CCU collected besides Mirai-RGU [8] and ISCX2012 [31] dataset. Primarily, the performance is affected by word-embedding and attack representation in the dataset. Unfortunately, the size of the input data exceeded the size of flow-based features. Thus, the time required for training was higher than usual, reaching 17 hours at 200 epochs on some datasets.

As discussed in this section, a growing body of the literature has recognized the importance of developing machine or deep learning models to detect IoT botnet attacks. These efforts are confronted with critical challenges that also point to gaps in this prominent research area. Distinctly, most proposed mechanisms focus on accuracy and disregard the analysis of other important metrics, such as algorithmic complexity and speed. Thus, this study proposes a lightweight model that provides real-time detection. Table I shows the proposed model in comparison to the current state-of-the-art.

IV. METHODOLOGY

The proposed classification model follows the same principle as in [9]. Thus, it treats each packet as a single sentence and each packet header field as a word because the stringent order of fields serves as a grammar rule, which is in essence creating sentence patterns for benign or malicious traffic. Therefore, word embedding is used to derive the semantics and syntactical features of packets. In the following subsections we will discuss dataset selection, feature extraction, dataset sampling, input preprocessing, proposed architecture, and experimental setup.

A. Dataset Selection

The effectiveness of neural network or deep learning models hinges primarily on the quality and size of a dataset. Research on IoT security suffers from the absence of benchmark datasets, but recent endeavors have been initiated to publish datasets meant to overcome this issue. Nevertheless, certain drawbacks remain. For example, the effectiveness of the dataset put forward in [33] is impeded by highly imbalanced records because it has only 477 legitimate traffic samples and 3,668,045 attack traffic samples. Among these recent efforts, and for the purposes of this study, the MedBIoT [32] and Mirai-RGU [8] datasets were selected. These datasets have been selected for the following reasons:

- A variety of IoT devices were used to generate the network traffic.
- There was realism in the attacks because real botnet binary codes were used to launch the attacks.
- Both phases of IoT botnet lifecycle are covered (see Section II-A).
- There was a diversity of attacks that were launched.

The MedBIoT dataset [32] is collected from a medium-sized network with 83 physical and virtual IoT devices, including switches, light bulbs, locks, and fans. Mirai [17], BashLite [15] and Torii [34] were used to initiate the malicious behavior of botnets. In contrast, the Mirai-RGU [8] dataset was generated using two Sricam AP009 IP cameras infected with Mirai source code that initiated different attacks against a raspberry Pi.

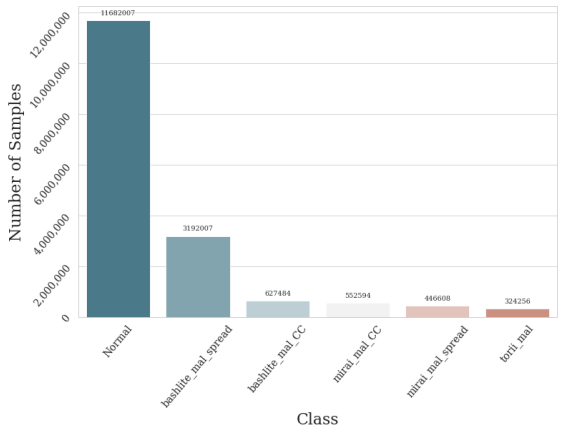

Fig. 3. MedBIoT Dataset [32] Classes Before Undersampling.

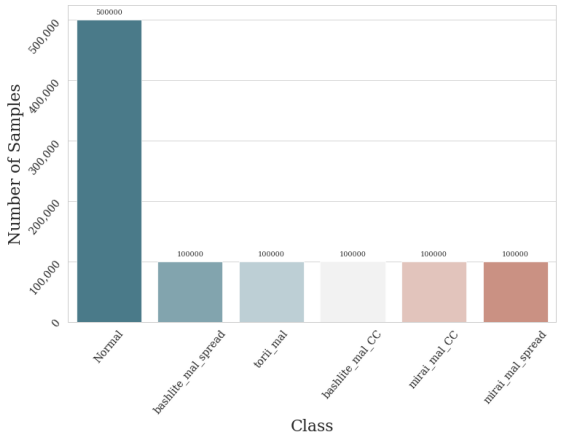

Fig. 4. MedBIoT Dataset [32] Classes After Undersampling.

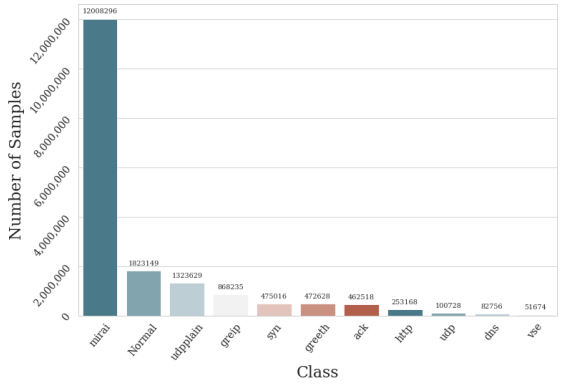

Fig. 5. Mirai-RGU Dataset [8] Classes Before Undersampling.

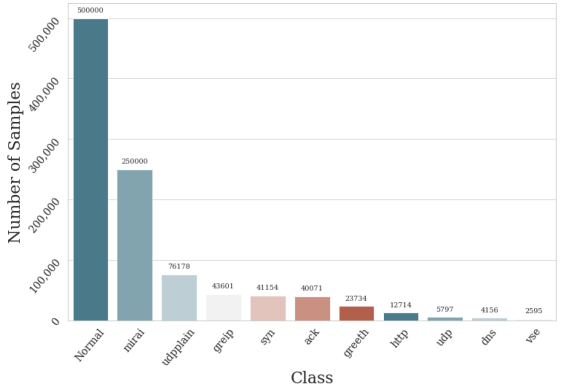

Fig. 6. Mirai-RGU Dataset [8] Classes After Undersampling.

TABLE II. PACKET HEADERS THAT USED AS FEATURES FOR THE PROPOSED INDEPENDENT MODEL.

Header	Features
Ethernet	eth_source, eth_destination, eth_type
IP	ip_version, ip_hdr_len, ip_tos, ip_length, ip_identification, ip_flags, ip_offset, ip_ttl, ip_protocol, ip_checksum, ip_source, ip_destination
TCP	tcp_source_port, tcp_destination_port, tcp_sequence, tcp_acknowledge, tcp_offset, tcp_flags_res, tcp_flags, tcp_window_size, tcp_checksum, tcp_urgent_point
UDP	udp_source_port, udp_destination_port, udp_ulen, udp_checksum

B. Feature Extraction

Both datasets consist of raw network packets as packet capture files (PCAPs). To provide an independent, lightweight, and real-time model, we needed to extract the features that are directly readable by the gateway. The required features were extracted from PCAPs using TShark [35] and converted into comma-separated values (CSVs). The extracted features were Ethernet, IP, TCP, and UDP headers, as displayed in Table II.

C. Dataset Sampling

A random undersampling technique followed to minimize the number of samples and to introduce some kind of balancing for the imbalanced classes. For MedBIoT [32], we split the dataset into two halves, normal and attacks, and then divided the attack classes equally. For the Mirai-RGU [8], we followed the attack vectors distribution of Mirai published by [17] to reflect a more realistic situation. Fig. 3, 4, 5, and 6 illustrate the undersampling effect on the classes of the datasets.

D. Input Preprocessing

Unlike LSTM-based model by [9], we did not duplicate any features. We only considered real packet headers because they require less preprocessing. To prepare a packet for embedding, all features were first converted into strings. Then, we split the dataset into training and testing sets in a ratio of 80:20, respectively. Afterward, tokenizer was applied to produce the dictionary and map each packet header with its associated integer number from the dictionary. Finally, we padded each packet to be the size of 32 words.

E. Architecture Designing

Basically, the proposed model consists of input layer, embedding layer, FastGRNN layer, dropout layer, and dense layer as illustrated in Fig. 7. First, vector of tokenized words or header fields with a size of 32 represented the input layer. The second layer was the random embedding layer that transferred each tokenized word **n** into a vector of size 64. Then, each embedded vector was passed into a FastGRNN cell with

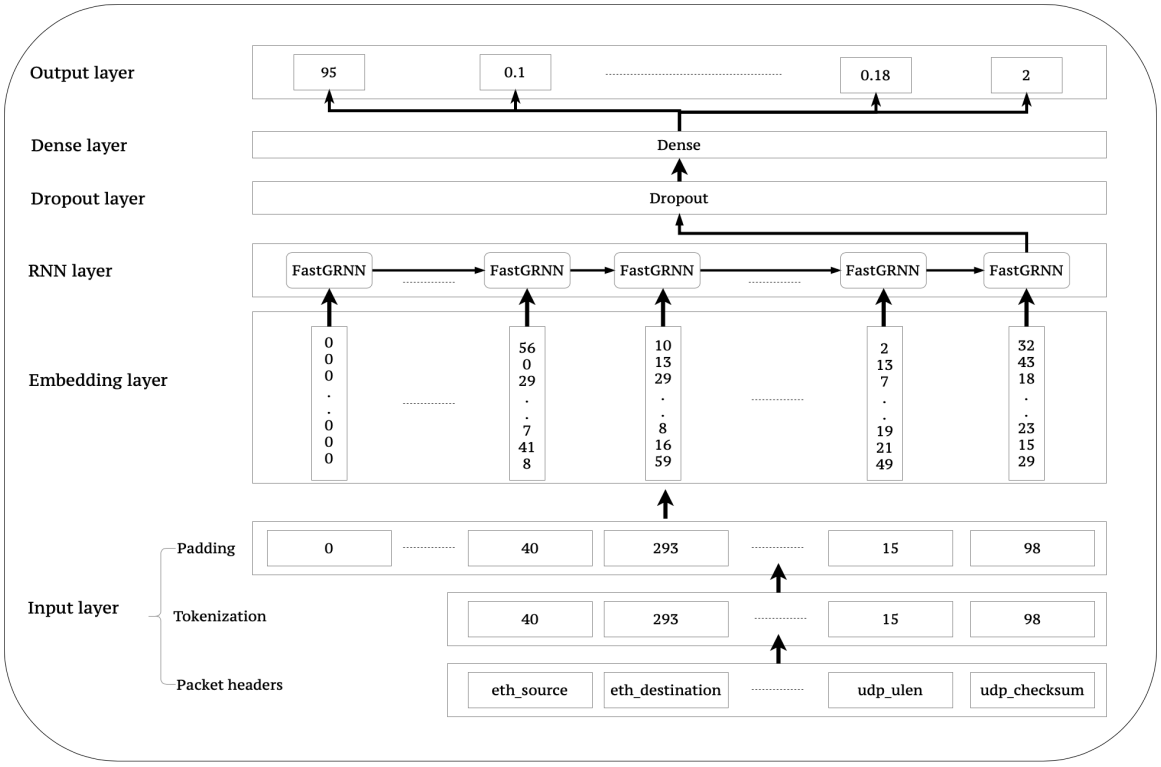

Fig. 7. The Proposed FastGRNN-based Model.

a hidden state of size 64. As mentioned in Section II-B, FastGRNN was selected due to its simplicity and lightness. Afterward, the dropout layer was used with 0.2 as the dropout rate to overcome the overfitting by dropping random neurons from the previous layer. To generate the desired output for the multi-classification task, a dense layer with Softmax as the activation function was used. Finally, to compile the model, a categorical cross entropy was used as the loss function in addition to RMSProp optimizer to adjust the learning rate.

F. Experimental Setup

The model was written in Python 3.7.3 and TensorFlow 1.15.0 [36] with Keras 2.2.4 [37]. All the experiments were done using Tesla K20 GPU, with 2496 CUDA cores and 5 GB memory besides 96 GB RAM.

V. Results and Discussion

To evaluate the model against the desired objectives, we needed to calculate the correctness of classification and the required time for training and prediction. Because both datasets were imbalanced and the model targets multi-classification, F1 score was the most appropriate metric to use. The F1 score was calculated according to Equation 4. In addition, wall time was considered when calculating the time required for training and detection. Furthermore, the model was compared with the LSTM-based model proposed by [9], but because there is no published information regarding time or the MedBIoT dataset in the paper by [9], we implemented their model and trained it ourselves. Moreover, we implemented the proposed architecture once with LSTM as a replacement for FastGRNN and called it the LSTM-based model, and then we implemented it once with GRU and called it the GRU-based model. We trained both of those architectures with both datasets, and the results are summarized in Table III.

$$\boldsymbol{F_1 = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}} \tag{4}$$

As shown in Table III, our FastGRNN achieved the lowest training time for MedBIoT at only 1 hour, 18 minutes, and 51 seconds (1:18:51), while the second-lowest one was the LSTM-based model at 4 hours, 3 minutes, and 5 seconds (04:03:05). In addition, FastGRNN had the fastest detection speed of only 29 seconds for the entire test set, compared to the second-lowest time which was 53 seconds for the GRU. The reason the GRU had a longer training time than the LSTM is that the GRU needed more epochs to train before stopping. Actually, GRU takes about 25 minutes to complete one epoch, while LSTM completes an epoch in about 27 minutes. The LSTM-based model proposed by [9] had the slowest performance in training and detection due to using multiple LSTM layers and a large hidden states size, which makes the computations more expensive. For the F1 scores, all of the models had F1 scores of 99.99%, which might be due to the balancing of the attack classes.

Afterward, we followed another strategy of balancing classes with Mirai-RGU, as mentioned in Section IV-C. Again, the proposed model completed training within 2:0:41 while the second-fastest one, which was GRU, took 3:51:2. Also, work by [9] took the longest time to train, 10:42:38. Regarding the detection time, FastGRNN succeeded in reaching a detection time of 29 seconds, while GRU needed about 55 seconds.

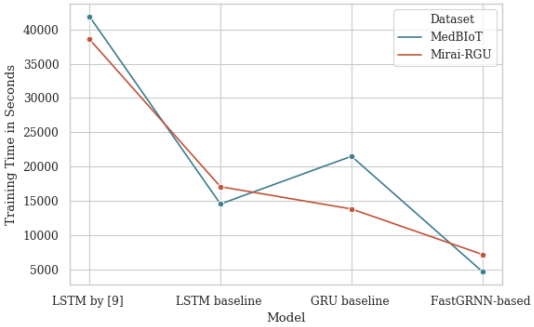

Fig. 8. The Required Training Time of The Proposed FastGRNN-based Model Compared to Other Models.

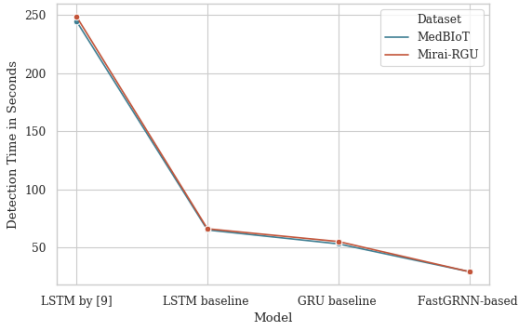

Fig. 9. The Required Detection Time of The Proposed FastGRNN-based Model Compared to Other Models.

TABLE III. EVALUATION RESULTS OF THE PROPOSED MODEL ON THE TWO DATASETS COMPARED TO OTHER MODELS.

Dataset	Model	F1 score	Training Time in hours:minutes:seconds	Detection Time in seconds
MedBIoT [32]	LSTM by [9]	99.99%	11:37:53	245
	LSTM baseline	99.99%	4:3:5	65
	GRU baseline	99.99%	5:58:49	53
	FastGRNN-based	99.99%	1:18:51	29
Mirai-RGU [8]	LSTM by [9]	99.46%	10:42:38	249
	LSTM baseline	98.60%	4:44:59	66
	GRU baseline	97.82%	3:51:2	55
	FastGRNN-based	99.04%	2:0:41	29

The proposed model outperformed GRU and LSTM in F1 score as well, with 99.04%, 97.82%, and 98.60%, respectively. Furthermore, FastGRNN achieved an F1 score close to that of LSTM. Finally, Fig. 8 and 9 illustrate the performance of the proposed model in terms of training and detection time compared to other models.

VI. CONCLUSION

IoT botnets are increasingly recognized as a serious worldwide cybersecurity concern. Investigating machine and deep learning is a continuing concern in relation to intrusion detection approaches against IoT botnets, but such exploration involves several issues. This study focused on developing a lightweight multi-classification neural network-based model with the aim of providing fast training time, real-time detection, and accuracy. According to the experiments, we proved that the proposed FastGRNN outperformed the other models when benchmarking both datasets by decreasing training and detection time while also preserving a high F1 score. Specifically, the proposed model completed training in 1:18:51 and 2:0:41 for the MedBIoT and RGU datasets, respectively. Detection was completed by FastGRNN within 29 seconds for the entire test set. Moreover, our model had competitive F1 scores of 99.99% and 99.04% for multi-classification of MedBIoT and RGU, respectively.

Finally, distinct technologies, along with IoT botnet detection measures, may be adopted. As future work, we will look into the opportunities engendered by federated learning. Because we aim to centralize the learning process using FastGRNN on the grounds of a fog or cloud and distributing a collection of network traffic data among several nodes or gateways, this direction would promote the application of collaborative intrusion detection approaches.

ACKNOWLEDGMENT

This work was supported by the Deanship of Scientific Research (DSR), King Abdulaziz University, Jeddah, under grant No. (DG1441 – 59 – 612). The authors, therefore, gratefully acknowledge the DSR technical and financial support. In addition, computation for the work described in this paper was supported by King Abdulaziz University's High Performance Computing Center (Aziz Supercomputer)[1] with NVIDIA Tesla K20 GPU.

REFERENCES

[1] A. Blanter and M. Holman, "Internet of things 2020: a glimpse into the future," *Available at Kearney https://www. atkearney. com/documents/4634214/6398631/AT+ Kearney_Internet+ of+ Things*, 2020.

[2] E. Bertino and N. Islam, "Botnets and internet of things security," *Computer*, no. 2, pp. 76–79, 2017.

[3] J. Habibi, D. Midi, A. Mudgerikar, and E. Bertino, "Heimdall: Mitigating the internet of insecure things," *IEEE Internet of Things Journal*, vol. 4, no. 4, pp. 968–978, Aug 2017.

[4] M. F. Elrawy, A. I. Awad, and H. F. Hamed, "Intrusion detection systems for iot-based smart environments: a survey," *Journal of Cloud Computing*, vol. 7, no. 1, p. 21, 2018.

[5] L. Cui, S. Yang, F. Chen, Z. Ming, N. Lu, and J. Qin, "A survey on application of machine learning for internet of things," *International Journal of Machine Learning and Cybernetics*, vol. 9, no. 8, pp. 1399–1417, 2018.

[6] K. A. da Costa, J. P. Papa, C. O. Lisboa, R. Munoz, and V. H. C. de Albuquerque, "Internet of things: A survey on machine learning-based intrusion detection approaches," *Computer Networks*, vol. 151, pp. 147–157, 2019.

[7] M. A. Al-Garadi, A. Mohamed, A. Al-Ali, X. Du, I. Ali, and M. Guizani, "A survey of machine and deep learning methods for internet of things (iot) security," *IEEE Communications Surveys & Tutorials*, 2020.

[1] http://hpc.kau.edu.sa

[8] C. D. McDermott, F. Majdani, and A. V. Petrovski, "Botnet detection in the internet of things using deep learning approaches," in *2018 International Joint Conference on Neural Networks (IJCNN)*, July 2018, pp. 1–8.

[9] R.-H. Hwang, M.-C. Peng, V.-L. Nguyen, and Y.-L. Chang, "An lstm-based deep learning approach for classifying malicious traffic at the packet level," *Applied Sciences*, vol. 9, no. 16, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/16/3414

[10] I. Hafeez, A. Y. Ding, M. Antikainen, and S. Tarkoma, "Real-time iot device activity detection in edge networks," in *Network and System Security*, M. H. Au, S. M. Yiu, J. Li, X. Luo, C. Wang, A. Castiglione, and K. Kluczniak, Eds. Cham: Springer International Publishing, 2018, pp. 221–236.

[11] N. Moustafa, B. Turnbull, and K. R. Choo, "An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4815–4830, June 2019.

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

[13] E. Anthi, L. Williams, and P. Burnap, "Pulse: An adaptive intrusion detection for the internet of things," in *Living in the Internet of Things: Cybersecurity of the IoT - 2018*, 2018, pp. 1–4.

[14] A. Kusupati, M. Singh, K. Bhatia, A. Kumar, P. Jain, and M. Varma, "Fastgrnn: A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network," in *Advances in Neural Information Processing Systems*, 2018, pp. 9017–9028.

[15] K. Angrishi, "Turning internet of things (iot) into internet of vulnerabilities (iov): Iot botnets," *arXiv preprint arXiv:1702.03681*, 2017.

[16] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "Ddos in the iot: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[17] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the mirai botnet," in *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, Aug. 2017, pp. 1093–1110. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis

[18] N. Vlajic and D. Zhou, "Iot as a land of opportunity for ddos hackers," *Computer*, vol. 51, no. 7, pp. 26–34, July 2018.

[19] S. Edwards and I. Profetis, "Hajime: Analysis of a decentralized internet worm for iot devices," *Rapidity Networks*, vol. 16, 2016.

[20] S. Mansfield-Devine, "Weaponising the internet of things," *Network Security*, vol. 2017, no. 10, pp. 13 – 19, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1353485817301046

[21] S. Zheng and X. Yang, "Dynashield: Reducing the cost of ddos defense using cloud services," in *11th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 19)*. Renton, WA: USENIX Association, Jul. 2019. [Online]. Available: https://www.usenix.org/conference/hotcloud19/presentation/zheng

[22] M. Šimon, L. Huraj, and T. Horák, "Ddos reflection attack based on iot: A case study," in *Cybernetics and Algorithms in Intelligent Systems*, R. Silhavy, Ed. Cham: Springer International Publishing, 2019, pp. 44–52.

[23] S. Soltan, P. Mittal, and H. V. Poor, "Blackiot: Iot botnet of high wattage devices can disrupt the power grid," in *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, Aug. 2018, pp. 15–32. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/soltan

[24] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[25] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[26] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[27] V. Paxson, "Bro: a system for detecting network intruders in real-time," *Computer Networks*, vol. 31, no. 23, pp. 2435 – 2463, 1999. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128699001127

[28] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, Nov 2015, pp. 1–6.

[29] F. Haddadi and A. N. Zincir-Heywood, "Benchmarking the effect of flow exporters and protocol filters on botnet traffic classification," *IEEE Systems Journal*, vol. 10, no. 4, pp. 1390–1401, Dec 2016.

[30] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[31] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers & Security*, vol. 31, no. 3, pp. 357 – 374, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404811001672

[32] A. Guerra-Manzanares, J. Medina-Galindo, H. Bahsi, and S. Nõmm, "Medbiot: Generation of an iot botnet dataset in a medium-sized iot network." in *ICISSP*, 2020, pp. 207–218.

[33] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset," *Future Generation Computer Systems*, vol. 100, pp. 779 – 796, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X18327687

[34] R. Vishwakarma and A. K. Jain, "A survey of ddos attacking techniques and defence mechanisms in the iot network," *Telecommunication Systems*, vol. 73, no. 1, pp. 3–25, 2020.

[35] G. Combs, "Tshark-the wireshark network analyser," *URL http://www.wireshark. org*, 2017.

[36] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[37] F. Chollet *et al.*, "Keras documentation," *keras.io*, 2015.