

Attention-based deep learning approach for detecting IoT botnet-based distributed denial of service attacks

Maha Al-Jarah

*Department of Computer Engineering
Jordan University of Science and Technology
Irbid
Jordan*

Mohammad Al-Shurman

Basheer Al-Duwairi *
*Department of Network Engineering & Security
Jordan University of Science and Technology
Irbid
Jordan*

Abstract

Internet of Things (IoT) technology has evolved rapidly to become an integral part of our daily life. The numerous number of IoT devices and the limited security measures that could be applied on such devices attracted attackers to exploit these devices to act as botnets, generating massive Distributed Denial of Service (DDoS) attacks. IoT botnet-based DDoS attacks are growing dramatically both in frequency and sophistication and thus call for urgent development of powerful detection mechanism and deploy those mechanisms in an efficient way. In this paper, we propose attention-based deep learning approach for IoT botnet-based DDoS attacks. Specifically, we apply deep learning attention mechanism on two popular deep learning models LSTM and CNN for detection of IoT botnet DDoS attacks. We also examine the effect of using the deep Autoencoder for feature reduction and examine its effect on training time and data size needed to be exchanged through the network for training and updating the model, especially for solutions deployed at the Edge computing network. Performance evaluation shows that combining attention mechanism with LSTM model achieves superior performance with an accuracy reaching 100% which is due to combining the memorizing ability by the LSTM and the focusing ability of the attention in one model. In addition, results show that using Autoencoder for feature reduction resulted

* E-mail: basheer@just.edu.jo

in reducing the training time up to 66.25%, 64.39%, 72.30% and 78.64% for ALSTM, LSTM, ACNN, CNN models respectively. Also achieved reduction in size of the dataset up to 67%.

Subject Classification: 68T07 Artificial neural networks and deep learning.

Keywords: IoT, Botnet, DDoS, Deep learning, Attention, LSTM, CNN, Autoencoder.

1. Introduction

Internet of Things (IoT) has evolved rapidly as result of the advancement in several enabling technologies such as wireless communication, cloud computing, fog computing, sensors, authentication and identification technologies. IoT enables tremendous number of electronic devices and sensors to seamlessly communicate in order to provide innovative services in wide range of domains including healthcare, smart homes and cities, transportation, security and many others [1]. According to [2], the world by year 2018 has witnessed the deployment of 22 billion IoT devices, this number is expected to reach 50 billion by 2030; However, such expansion growth gives rise to additional cybersecurity risks such as IoT botnet attacks [3] [4], attackers exploit vulnerabilities in IoT devices to perform malicious activities at large scale, these vulnerabilities comes as result of using default credentials and insufficient authentications especially when fast-paced employment is needed for large number of devices in the system [5], also the limited computing and storage capacity of IoT devices makes applying a powerful security mechanisms unattainable.

IoT botnet attacks include illegal access, data theft and Distributed Denial of Services (DDoS) in which attackers overwhelm victim targets as well the network with huge amount of malicious traffic causing significant economic losses [6]. Mirai is one example of recent IoT botnet DDoS attack in which the IoT vulnerability of known credentials is exploited to generate a massive DDoS attack of about 1.1 Tbps [7]. IoT botnet DDoS attacks are becoming more developed and more harmful and thus an accurate detection mechanism is needed [8].

In this paper, we propose a mechanism for detecting IoT botnet DDoS attacks using attention based deep learning models. Deep learning is a branch of machine learning that has gained heightened scholarly attention and achieved the state of art results in various fields. One of the key advantages of deep learning models over traditional machine learning models is that deep learning does not require engineered features, instead

it's able to extract latent features automatically from raw input, its intuition is to define a computational model of large set of parameters and through a data driven training process the model's parameters are approximated to describe the complex relationships between the input data and the output, without human intervention. Deep learning models are trained to be resilient to small changes in input by determining the high level invariant features of the training samples, which can be very beneficial in detecting zero-day attacks, as long as a considerable percentage of new attacks are small mutations of previous attacks or even rely on same concepts [9].

Moreover we apply attention mechanism to improve the performance of deep learning models, the recent introduction of attention in deep learning has achieved great improvement in natural language processing, machine translation and other tasks, in general attention mechanism used to further highlight the important factors in processing the data automatically and thus produce better results [10]. Choosing among the different deep learning models for solving a problem is not an easy decision to be made, due to the wide variety of deep learning models and the rapid development this area is witnessing. Hence investigating the literature for models that achieved good performance in similar problems and developing the models using recent deep learning approaches will help in achieving better performance. The main contributions of this paper are as follows:

- Analyzing the effect of exploiting the attention mechanism on two popular deep learning models (LSTM and CNN) for the purpose of detecting IoT botnet DDoS attacks.
- Analyzing the effect of using autoencoder for the purpose of feature reduction on training time also its effect on amount of traffic needed to be exchanged for the purpose of training and updating the model especially for fast deployment of models using edge and fog computing.

The rest of the paper is organized as follows: Section 2 provides background about IoT botnets and IoT-botnets based DDoS attacks. Section 3 discusses related work. Section 4 discusses deep learning models. Section 5 presents the methodology. Section 6 presents performance evaluation while Section 7 presents conclusions and future work.

2. Background

In this section, we provide background about IoT botnets and IoT botnets-based DDoS attacks.

2.1. *IoT Botnets*

IoT botnet is defined as a network of compromised IoT devices where attackers gain illegal control over the network to cause disabling damages to the system [11]. According to Bezerra et al. [12] the typical IoT botnet consists of the following components:

- Bots: bots are IoT devices infected by malware that enable attackers of controlling them to initiate attacks against selected targets and propagate the infection to other devices.
- Command and control C&C server: The main server used by the attacker to send commands controlling the network of the botnet.
- Scanner: An automated scanning tool or customized software used on each bot for the purpose of identifying new bots, scanner process continuously scans the targeted network to find IoT devices with well known vulnerabilities and open ports that allow accessing the device. The scanner will try to log into the device by exploiting know vulnerability or through brute force, If succeeds then the scanner will send the device information to the reporting server.
- Reporting server: stores the detailed information of the compromised devices such as IP address, log in credentials and hardware architecture.
- Loader: by receiving a command from the C&C server loader will log into the compromised devices and instruct the device to download and execute malware based on the device architecture. The loader will then attempt to secure points of intrusion to protect the device from other malware.

Once the attacker has compromised enough bots, C&C specifying the target IP address and all details information of the attack, will issue an attack command to all available bots to start the attack.

2.2. *IoT botnet-based DDoS*

DDoS attacks originating from IoT botnets is a major cyber-security threat and is considered to be one of the most destructive and widely seen

cyber-attacks. In DDoS attacks, the targeted server becomes unable to respond to legitimate requests from end-users due to the high volume request rate originating from botnet IoT devices. IoT botnet-based DDoS attacks are considered particularly destructive and challenging to defend against and mitigate, because of the massive number of participating IoT devices and the widely distribution nature they have. Infected IoT devices are also able to evade traditional detection mechanisms such as network intrusion detection systems (NIDS) as their data appears to be originating from valid IP addresses [13]. Botnet collaboration is another mechanism in which botnet are arranged into groups alternate attacking the same targets which is further complicate detecting and mitigating the attack as the source keep changing over time [14].

The most popular and prominent example of IoT botnet used to launch DDoS attacks is the Mirai botnet. Assembling a botnet of about 400000 IoT devices to launch several attacks with massive traffic exceeding 1 Tbps to hit down several high-profile websites such as Twitter, Netflix, and GitHub for about six hours [7]. Many variations of Mirai were developed since then to launch new attacks that elaborate existing weaknesses or to go around defences tailored for the original Mirai attack. Mirai scanner scans random public IPs through telnet protocol on TCP port 23 and port 2323. It attempts to guess the login credentials of the device by trying 62 of manufacturer default credentials list, hard-coded in the Mirai code [7]. After the shell is obtained, the characteristics of the compromised device including IP address, port, hardware architecture, username and password will be forwarded to the reporting server using a different port [7].

The C&C server will continuously connect with reporting server to check for new candidate bots, If there is any, an infect command will be issued to the loader with all the needed information. The loader will log into the device through telnet, and using a set of previously compiled binaries of the malware stored at the loader for a set of various architecture vendors of the IoT devices in order to load the proper binary according to the device architecture [15]. The loader will instruct the device to download the proper binary using GNU wget or Trivial File Transport Protocol (TFTP) and execute the binary. Once the malware is executed it will connect with the C&C server by resolving the domain name for the C&C which is hard-coded in the binary with the default entry (cnc.changeme.com) instead of static IP address, which gives the attacker the ability of using IP flux and changing its IP address continuously without changing the binaries [15].

Mirai uses several protection and prolonging mechanisms, to mention some, firstly it is considered as a file-less attack or memory only attack. As to avoid detection, immediately after execution, the malware will attempt to delete itself from the persistent file system while the malware process keep running in the memory. The malware will also attempt to protect itself from other competing worms by killing processes that use Telnet, SSH and HTTP ports and initiate its own socket with the C&C. The mechanism of locating and killing other malware processes from memory known as memory scraping[15].

From that point on, the bot will maintain waiting for commands from the C&C server and will exchange heartbeat packets every 60 secs, in this way if the device rebooted, it will automatically be disinfected but as the device information already stored at the reporting server and as the C&C is continuously checking the status of the bots it will be matter of few minutes after the device reboot to get infected again. The C&C attack command will include the attack type among 10 available attacks such as UDP, TCP, and HTTP flooding attacks, the duration of the attack, attack id, and the list of targets [7]. Another mechanism used by Mirai considered as prolonging method is the ``Don't mess with'' list, which is a list of hard coded IP addresses belong to the US Department of Defense, Postal Service, General Electric and other sensitive governmental websites, these IP addresses are excluded in the IP scan process, to avoid drawing attention of certain parties to the attack.

3. Related Work

Various approaches have been proposed for detecting IoT based botnet attacks, traditional approaches in the literature can be categorized into four groups including: host-based, network-based, anomaly-based and signature-based approaches.

3.1. *Host-based detection*

Starting with host-based approach, as the name indicates this approach relies on monitoring characteristics of each host in the system and activities associated with each host to detect abnormal behaviors. This is done using a software known as agent deployed at the inspected host and exchange data with a management server [16]. Host based approach generally used to protect critical hosts that have sensitive information or hosts that are used publicly. BotRevealer is a host based example proposed

by Khoshhalpour and Shahriari [17] to detect infected hosts at the early infection step based on known botnet life-cycle events.

According to [16] host based approach suffers from significant limitations such as the considerable configuration and customization needed to deploy detailed and up to date policies that describes the normal behavior of a host, especially on large scale environments. This approach also suffer from being prone to excessive delays to detect an event as the agents forward their data few times an hour to the management server, still the biggest concern is that IoT devices are already constrained in capabilities that applying agent software will results in degrading their performance.

3.2. *Network-based detection*

Network-based approach is more powerful approach because it monitors and analyzes network traffic of large number of hosts and have a wider view of the security situation compared to the constrained view provided by the host based approach [18][19]. Entropy calculation is a common measure used in the literature to build a normal profile of the system to be compared with in order to detect an attack [20], [21], as its noticed that change in distribution features of traffic is a common sign in traffic anomalies [22]. For example, change in distribution of the destination address in packet header to target a victim address is considered a sign of a Denial of Service attack (DoS), similarly a skewed distribution for a vulnerable port address during scanning phase of attack is another sign of attack.

Kumar et al. [20] applied entropy calculation based on the normal traffic features distribution to detect unusual traffic behavior, specifically using four features which are: source and destination IP addresses, source and destination Port addresses. In order to mitigate the computational and memory costs, monitoring, analysis and comparisons are distributed among points of connecting Internet Service Providers (ISPs) also called Point of Presence (PoPs).

Different mechanisms have been used to analyze network traffic to detect attacks one of them is event correlation analysis, Al-Duwairi et al. [23] proposed a detection and mitigation of IoT botnet based DDoS attacks based on Security Information and Event Management (SIEM) event correlation system that monitor and analyze IoT traffic to understand the normal behavior of the monitored IoT network for the purpose of alarming

the network admin on detected suspicious events and activating certain defensive rules to block traffic from the infected device.

Fuzzy logic is another mechanism used in the literature to analyze network traffic for the purpose of detecting malicious events [24] [25] [26]. For example, xia2010enhancing proposed statistical analysis of network traffic to detect occurrence of DDoS attack and built a fuzzy logic system based on set of decision rules to decide the intensity of the attack. According to Rahmani et al. [27], detection methods that rely on heavy statistical work, will compromise a bottleneck to extract features of huge traffic records, to overcome this limitation many works in the literature suggest leveraging the distributed computing technologies, such as edge computing [13] and the content distribution networks (CDNs) [28] to handle the huge amount of network traffic in distributed manner and to accelerate the detection of the attack.

3.3. *Signature based detection*

Signature-based approach depends on a database of previously known malicious signatures compared to observed events to detect malicious activities [16]. Signatures are also used to identify users of certain server, and server load thresholds are set to anticipate possible malicious activities for example upon exceeding the LLT (low load threshold) suspicious users are delayed, and upon reaching the high load threshold suspicious use res are blocked [29].

Abbas and Srikanthan [30] introduced a signature based approach to detect a set of malware of known signatures, this approach achieves high accuracy rate for known attacks, however, it will fail if any change is applied to the original malware.

According to [16] signature based approach considered to be the simplest detecting approach however it suffers from drawback of the need to continuously updating signatures of newly discovered or updated attacks, and the incapability to detect zero-day attacks in which a simple tweak may be added to avoid detection, for example to obscure known traffic characteristics of Mirai attack, some of Mirai-variant attacks use encrypted information exchanged between bots and the C&C server.

3.4. *Anomaly-based detection*

Anomaly-based detection depends on monitoring the system for a long period of time to be able to model the normal behavior of the system, when unusual activities are encountered anomaly based approaches are

able to capture the deviation from normal behavior and thus detect even novel unknown attacks [31]. The work presented in Barford et al. [32], provided an anomaly detection of network traffic using signal analysis such as applying wavelet filters that are able to identify signal characteristics as a time-frequency representation, this approach was able to address traffic cycles, both daily and weekly to detect deviations.

Another example is an anomaly based protocol called Watcher proposed by Bradley et al. [33], was able to detect malicious routers involved in DDoS attack. Watcher algorithm running at the router will check if each incoming packet was routed correctly and if a neighboring router misrouting packets that are not destined to this router then the neighboring router will be removed from the network. Zargar et al. [34] criticized Watchers algorithm for requiring overhead communication between the routers, and its inability of detecting spoofed packets, another concern is that attackers may exploit this approach to cause in misidentifying a targeted router as a malicious router.

Despite performing well in detecting new attacks anomaly based approach suffers from high false alarm rate as this approach tend to confuse a normal behavior of the system as an attack. [35].

3.5. *Deep Learning approaches*

Recently, deep learning approach in solving cyber-security problems gains a wide research interest. Basumallik et al. [36] proposed using convolutional neural network as filter for detecting falsified data in power monitoring systems that use Phasor Measurement Units (PMU) for state estimation, the CNN model achieved an accuracy of 98.67% outperforming other deep learning models as RNN and LSTM.

Convolutional neural network was also employed in a framework for detecting fraud behaviors [37], they used a 2-d matrices to represents features of customers credit card transactions, these matrices fed into the CNN model for training to detect the latent patterns of both legitimate and fraudulent transactions. The CNN model achieved the best performance compared to other methods as Support Vector Machine (SVM), Neural Network (NN) and Random Forest (RF).

DeppCorr is a framework that depends also on CNN model proposed by Nasr et al. [38] as a flow correlation and deanonymization to demonstrates the growing risk of the flow correlation attacks exploiting deep learning approaches as CNN achieved an accuracy of 96% compared to the traditional statistical methods achieving 4%.

Feng et al. [39] proposed a plug and play device to detect DoS, SQL and Cross-site scripting (XSS) attacks in ad-hoc networks. The device consists of a tool for capturing packets from the network and a deep learning model for anomaly based attacks detection, three models chosen for comparison including Deep Neural Network DNN, CNN and LSTM. DNN model achieved the best results for detecting DoS attack while the LSTM model achieved higher performance records for the XSS and SQL attacks.

Ahmed et al. [40] proposed a binary classification model for detecting botnet attacks using simple feed forward neural network. A preprocessing component is used to extract 15 statistical features from the network traffic flow to train the models. Multiple configuration of the model were compared and the results show neural network with 3 layers, 10 neurons each achieved the best accuracy of 99.6% outperforming other methods as SVM and Naïve Bayes (NB).

Distributed approach based on deep learning models for detecting IoT botnet attacks was proposed by parra et al. [41] by exploiting two models, LSTM model placed at the cloud and a distributed CNN model as an add-on used on each IoT device. The distributed CNN models aim at detecting phishing attacks at their origins while the LSTM model aims at detecting distributed nature attacks as DDoS attacks that can not be detected locally on the IoT devices. The approach was able to detect phishing attacks and botnet attacks with an accuracy of 94.3% and 94.8% respectively.

A recent survey proposed by Alsoufi et al. [42], investigated deep learning models used for anomaly intrusion detection in IoT environment, shows the superiority of using deep learning models in detecting anomalies in terms of accuracy and false alarm rate while considering the constrained nature of IoT resources.

The study also emphasized that using deep learning models has succeeded in discriminating latent features that not recognizable for human beings, which makes deep learning approach preferable option over other shallow learning approaches. The use of complex and high dimensional datasets in some literature in the survey was criticized for requiring high computational resources to be processed in limited time, the use of modern deep learning models is a solution for dimensional reduction while preserving the information contained in the dataset. The study concluded that combining Auto-Encoder (AE) with Convolutional Neural Network (CNN) were the most commonly used approach for enhancing performance of intrusion detection system (IDS) in the surveyed

literature. The survey authors suggest examining the less commonly used deep learning models that have not been studied in IoT environment.

In our work we use Auto-Encoder for dimensional reduction of our dataset, for the purpose of reducing computations and network overhead. We also investigate the effect of combining the recent deep learning Attention mechanism with the CNN model and compare it with using attention mechanism with the Long Short-Term Memory (LSTM) model, to the best of our knowledge, this is the first work to analyze the effect of exploiting the attention mechanism on the two popular deep learning models (LSTM and CNN) in the field of detecting IoT botnet-based DDoS attacks.

4. Deep Learning models Architecture

4.1 CNN

CNN is one of the most popular deep learning models proven to be so efficient for problems involving image data. The kernel filters used in CNN models allow them to extract fine features and recognize the spatial and hidden order relationships within the data, although CNN were specifically designed to deal with image data, CNNs used with other types of data also achieved state of art results [43] [44] [45] [46] which encouraged us to examine its performance against our problem.

The CNN model consist of the following layers:

1. **Convolutional layer:** In this layer a set of kernel filters slides across the input sample performing dot product between the input and the filter to extract feature maps which by training will represent the hidden relationships between elements of the input sample itself.

For our dataset we are going to use 1-D convolution [47], for which we represent the vector of the input sample as the following discrete function: $g(x) \in [1, f] \rightarrow \mathbb{R}$, and the kernel vector as the following function: $k(w) \in [1, n] \rightarrow \mathbb{R}$.

Having $m = ((f - n) / s) + 1$, then the convolution $q(y) \in [1, m] \rightarrow \mathbb{R}$, between the input and the kernel with the stride s , will be represented as:

$$q(y) = \sum_{x=1}^n k(n - x + 1) \cdot g((y \cdot s) - x + z) \quad (1)$$

where $z = n - s + 1$ is the offset.

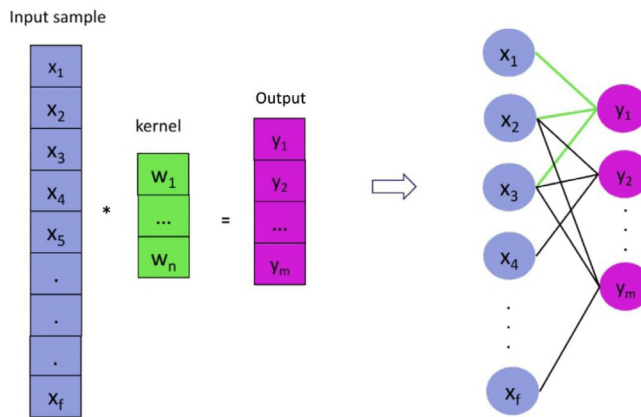


Figure 1
1D convolution layer for one filter

Number of kernels used in each convolutional layer is a hyper parameter that could be tuned during validation.

The convolution layer has two important attributes that make it efficient computationally, the first is the reduction of number of parameters as seen in the Figure 1 neurons in the consecutive layers are not fully connected instead they are connected with limited number of connections depending on the size of the filter. the second attribute is parameter sharing because we use the same filter across all the input features.

2. **Activation layer:** used to introduce non-linearity to our CNN model, we chose the Rectified Linear Units (ReLU) as the activation functions because its found to have advantages over other activation functions when used with CNN [48].

Let the layer l be the activation layer that takes the output of the previous convolution layer $(l-1)$ as input to produce the activation map y^l as follow:

$$\tilde{q}(y^l) = Relu(y^l) = \max(0, y^{(l-1)}) \quad (2)$$

3. **Pooling Layer:** The pooling layer used mainly for reducing the size of the feature map and achieve invariant translation, providing an abstract version of the feature maps.

The pooling layer applied on each feature maps that result after the activation layer, in a similar way to the kernel filters but instead of

the convolution operation we use one of the two common functions either the average or the maximum.

For our model we used the maximum pooling in which we apply filters that calculate the maximum value upon patch of values in the feature map the filter is applied on.

$$p(y) = \max_{x=1}^n \tilde{q}((y.s) - x + z) \quad (3)$$

4. **Fully connected Layer:** used at the end of the CNN to generate the final class probability distribution. Its simply a multi-layer neural network with the sigmoid activation function used at the last layer instead of the Relu to give us the probability distribution.

$$\text{sigmoid}(x) = 1 / 1 + e^{-x} \quad (4)$$

4.2. LSTM

LSTM is a special kind of Recurrent Neural Networks (RNN) developed to solve problems found in RNNs. Figure 2 illustrates how RNNs are able to learn relationship between events happening at different time by sharing parameters from previous time step, RNN learn to encode information in hidden states and a recurrent connection passes the hidden state from previous time step to be considered with inputs of current time step, in this way RNN allow previous information to persist.

$$h_i = \tanh(W_x X_i + W_h h_{i-1} + b_h) \quad (5)$$

$$y_i = \text{softmax}(W_y h_i + b_y) \quad (6)$$

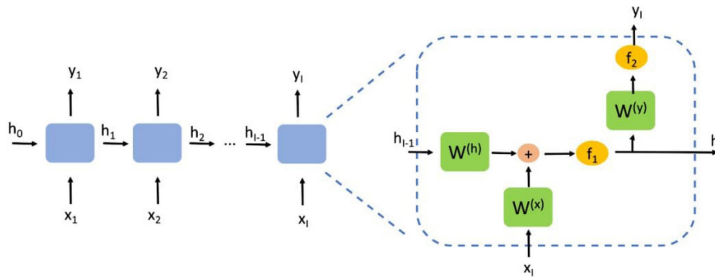


Figure 2

Internal structure of RNN cell, consisting of three trainable sets of weights $W(h)$, $W(x)$, and $W(y)$. Activation functions $\tanh$ represented by f_1 and softmax represented by f_2 , for the hidden state and the output.

As the gap between possible relevant information increases RNNs become inefficient in discovering these connections due to the problem of vanishing gradients [49], [50]. LSTM networks have the ability to learn Long term dependencies. The processing block in RNN is a single neural network repeated in each layer, as shown in Figure 3, LSTM expand this block to have four interacting gates in the single cell which are: the forget, input, update and output gates, those gates operate as valves through which the cell state flows and get carefully modified in a way that we can completely discard some information, add new one or update some with chosen updating values, the first decision to be made is what information from previous time step need to be forgotten in this step, this is done through the forget gate it consists of a sigmoid function that outputs a number between zero (for information that we need to discard) and one (for the information that we would like to keep as is).

$$f_i = \sigma(W_f \cdot [X_i, h_{i-1}] + b_f) \quad (7)$$

then the old cell state is scaled by multiplying it by the sigmoid output, but before computing the final new cell state we need to pass through the other gates.

In the next step we are going to choose some values to be updated. to decide what piece of information to update we also use *sigmoid* function, this is called the input gate.

$$n_i = \sigma(W_n \cdot [X_i, h_{i-1}] + b_n) \quad (8)$$

to create the candidate values we use *tanh* function that result in values between -1 and 1.

$$\widetilde{C}_i = \tanh(W_c \cdot [X_i, h_{i-1}] + b_c) \quad (9)$$

Now we scale the candidate values by the input gate result, and the new cell state computed as in the following equation:

$$C_i = f_i * C_{i-1} + N_i * \widetilde{C}_i \quad (10)$$

Finally the cell state will pass through the output gate in which its values will be squashed between -1 and 1 using the *tanh* function and scaled using *sigmoid* function to choose which value to keep for future processing and which to discard.

$$h_i = \tanh(C_i) * O_i \quad (11)$$

where

$$O_i = \sigma(W_o \cdot [X_i, h_{i-1}] + b_o) \quad (12)$$

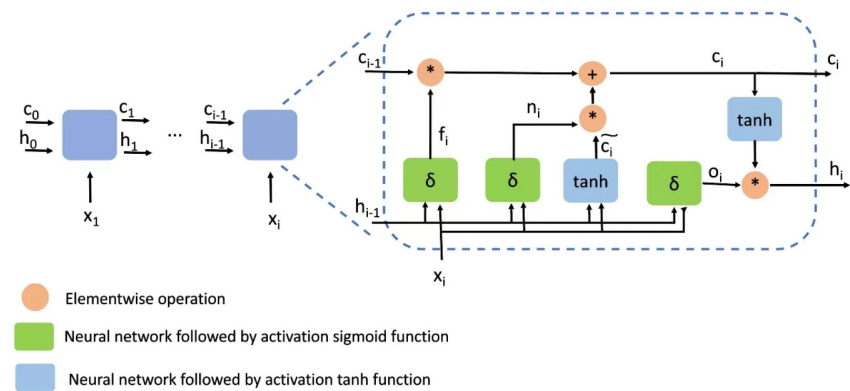


Figure 3
Internal structure of LSTM

4.3. Autoencoder

Autoencoder is a special type of neural network that is used mainly for dimensionality reduction.

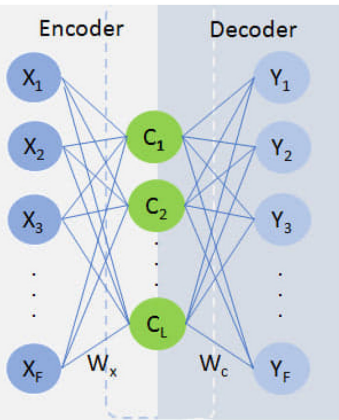


Figure 4
Autoencoder Architecture

The autoencoder architecture illustrated in Figure 4 consists of an encoder that take an input passes it through a fully connected feed forward neural network to generate the compressed feature representation of the input data or the code. For the input vector $X \in \mathbb{R}^{|F|}$, The encoding process represented in the following equation:

$$C = \sigma(W_x X + b_x) \quad (13)$$

the code vector $C \in \mathbb{R}^{|L|}$, L is the latent space size where $L < F$.

The second part is the decoder which is also a fully connected feed forward neural network that uses the code to reconstruct the original input as its output $Y \in \mathbb{R}^{|F|}$.

$$Y = \sigma(W_c C + b_c) \quad (14)$$

The objective of the autoencoder is to achieve the minimum reconstruction error between the original input and the decoder output, and according to the error we update the set of weights through backpropagation.

5. Proposed Mechanism

We propose the using of attention mechanism with LSTM and CNN models for the purpose of detecting DDoS attacks based on IoT botnets. Attention mechanism was inspired by the human attention in which we focus on certain parts of an input image or audio, while ignoring idle details or receiving them with less focus. In the following subsections, we discuss the proposed attention mechanism and the application of autoencoder for feature reduction.

5.1. Attention Mechanism

Attention layer used in deep neural networks tends to victorize each input and through training process these vectors get tuned to represent relationships between this input and every other input. For using attention the first step is the embedding layer in which we express every feature by a vector, this vector could by initialized by zeros or using random weights.

The the embedding vector fed through an LSTM network to result in the hidden state h_i , for dataset containing I records [51].

$$h_i = LSTM(E_i) \quad (15)$$

The attention weights are learned using a neural network followed by softmax function as follow:

$$u_i = \tanh(Wh_i + b) \quad (16)$$

where W are the weights, b are the offsets and u_i is the importance vector.

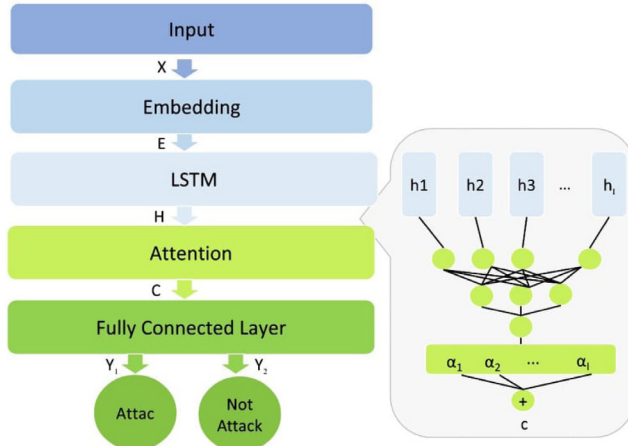


Figure 5
LSTM with Attention

The normalized probability distribution of u_i is obtained using softmax function result in the attention weights α_i

$$\alpha_i = \text{softmax}(u_i) \quad (17)$$

The context vector is obtained by the weighted sum of the attention weights and the hidden states as follow:

$$c = \sum_{i=1}^I \alpha_i h_i \quad (18)$$

As shown in Figure 5, the final prediction is obtained by propagating vector c through fully connected layer followed by sigmoid.

We also applied the same attention mechanism with the CNN model, instead of feeding the embedding vector to LSTM now we use the CNN model followed by the attention layer and the fully connected layer as well to get the final prediction results.

5.2. AutoEncoder for feature reduction

As our second contribution we propose exploiting Autoencoder for feature reduction as it will help in reducing amount of data needed to be exchanged for the purpose of training. Its also expected to expedite the process of updating the models especially for distributed solution deployed using fog computing [13] [52], the use of autoencoder to

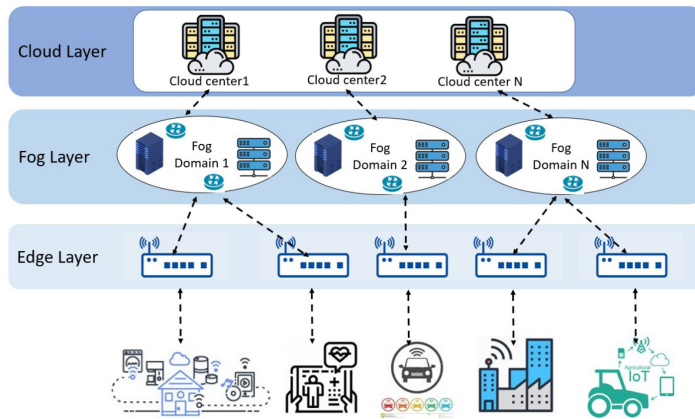


Figure 6

Distributed framework for detecting IoT attacks [52].

preprocess new data even of different shapes and generate compacted data of fixed size makes updating our models very fast.

Centralized security mechanisms will not be suitable for IoT architecture as the IoT devices are required to be distributed across wide geographical areas, Thus fog computing is considered a solution to overcome this challenge.

In [52] authors proposed a distributed framework for detecting IoT attacks using deep learning models, the framework consists of three layers, the Edge layer with huge number of IoT devices, the Fog layer that considered as an extension of the cloud data centers at distributed manner closer to the edge, and the Cloud layer that provides the powerful data processing and storage capabilities through the network as shown in Figure 6.

The edge layer generate a tremendous amount of data propagated to the fog layer where the attack detecting model is applied on the fog nodes and where the raw data is processed to extract the features, processed data then routed to the cloud where training, testing and updating of the models are done.

According to such solutions a huge amount of data needed to be exchanged through the network. To overcome this limitation we propose using an autoencoder at the fog layer to reduce amount of data to be exchanged and to make updating the models faster as the experiments shows a significant reduction in training time.

6. Performance Evaluation

The proposed models were implemented, trained and tested using the open source deep learning framework Keras, on Jupyter Notebook platform. The machine used has 8GB RAM, Processor Intel i5-6200U CPU and 64-bit Ubuntu 16.04 operating system. In the following subsections, we discuss the dataset used for training and testing, the performance metrics, and the results of the proposed models.

6.1. Dataset

We used the Bot-IoT Dataset [53], The dataset was produced at the Research Cyber Range lab of UNSW Canberra, testbed environment used to simulate a realistic IoT network where virtual machines used to simulate normal, attacking and targeted machines. We refer readers to [53] for details about the methodology used in producing the dataset. Table 1 list of the 32 features of the BoT-IoT used in our experiments, which were adopted mainly from [29].

Table 1
Features Description.

Feature	Description
pkSeqID	Row Identifier
Stime	Record start time
flgs	Flow state flags seen in transactions
flgs number	Numerical representation of feature flags
Proto	Textual representation of transaction protocols present in network flow
proto number	Numerical representation of feature protocol
saddr	Source IP address
sport	Source port number
daddr	Destination IP address
dport	Destination port number
pkts	Total count of packets in transaction
bytes	Totan number of bytes in transaction
state	Transaction state
state number	Numerical representation of feature state
ltime	Record last time

seq	Argus sequence number
dur	Record total duration
mean	Average duration of aggregated records
stddev	Standard deviation of aggregated records
sum	Total duration of aggregated records
min	Minimum duration of aggregated records
max	Maximum duration of aggregated records
spkts	Source-to-destination packet count
dpkts	Destination-to-source packet count
sbytes	Source-to-destination byte count
dbytes	Destination-to-source byte count
rate	Total packets per second in transaction
srate	Source-to-destination packets per second
drate	Destination-to-source packets per second
attack	Class label: 0 for Normal traffic, 1 for Attack Traffic
category	Traffic category
subcategory	Traffic subcategory

We wrote a python script to extract all the samples classified as DDoS attacks and the normal traffic data from the original dataset, composing one dataset of only DDoS attacks and normal traffic samples that will be shuffled and splited into training and testing datasets after the pre-processing steps. We started the preprocessing by discarding the category and the subcategory features as they are indicating if the sample is attack or not, and to avoid having redundant features we also eliminated the textual representation of some features as they have also a numerical representation as: flgs, proto, state.

The pkSeqID feature was introduced as an auto-incrementing feature in the pre-processing steps of the original Bot-IoT dataset work [53], but was not included in the dataset files, so the final number of features was 26 including the attack feature.

We applied the MinMaxScaler function because our features have different value scales, we normalized the dataset to scale all the values to be bounded between[0,1] as shown in Figure 7, normalization is a common practice in machine learning that helps models to optimize [54].

```

In [20]: train_x[0]
Out[20]: array([1.52809572e+09, 0.00000000e+00, 3.00000000e+00, 2.00000000e+00,
                2.09200000e+03, 2.20000000e+01, 2.70000000e+01, 4.00000000e+00,
                4.28000000e+02, 4.00000000e+00, 1.52809573e+09, 2.00500000e+03,
                8.74533000e+00, 1.67200000e-03, 1.67200000e-03, 3.34300000e-03,
                0.00000000e+00, 3.34300000e-03, 3.00000000e+00, 1.00000000e+00,
                3.68000000e+02, 6.00000000e+01, 3.43040000e-01, 2.28693000e-01,
                0.00000000e+00])

In [21]: scaler = MinMaxScaler()
         train_x = scaler.fit_transform(train_x)
         test_x = scaler.fit_transform(test_x)

In [22]: train_x[0]
Out[22]: array([9.99999991e-01, 0.00000000e+00, 7.50000000e-01, 9.52380952e-02,
                3.77004866e-01, 1.24293785e-01, 9.31034483e-01, 2.04030278e-05,
                2.47511747e-06, 8.00000000e-01, 9.99993225e-01, 6.13363614e-03,
                2.47316397e-03, 3.35005825e-04, 8.92843793e-04, 9.54491704e-07,
                0.00000000e+00, 6.68600134e-04, 2.57536152e-05, 1.36019260e-05,
                4.34804587e-06, 8.14022857e-07, 3.43040000e-06, 1.82954400e-06,
                0.00000000e+00])

```

Figure 7

Sample of training data before and after scaling.

MinMaxScaler is a popular choice as it preserves the original distribution of the dataset [54], MinMaxScaler function subtracts the minimum value in the feature and then divides by the difference between the original maximum value and original minimum value as shown in equation 19.

$$A_{new} = (A_{old} - A_{min}) / (A_{max} - A_{min}) \quad (19)$$

6.2. Performance metrics

To evaluate the performance of our models, we used four performance metrics: Accuracy, precision, recall and F1-score. These metrics are computed according to the confusion matrix in Table 2 that includes all the possible cases in the detection results.

Table 2
Confusion matrix

	Predicted Attack	Predicted Normal
Actual Attack	True Positive (TP)	False Negative(FN)
Actual Normal	False Positive (FP)	True Negative (TN)

Accuracy represents fraction of correct predictions our model got, as shown in Equation 20.

$$Accuracy = \frac{TP+TN}{TP+TN+FN+FP} \quad (20)$$

Precision in Equation 21, measures the proportion of the attack predictions that were actually correct to be attacks among all the attack predictions that were made.

$$Precision = \frac{TP}{TP+FP} \quad (21)$$

Recall measures proportion of actual attacks that were predicted correctly among all actual attacks, as in Equation 22.

$$Recall = \frac{TP}{TP+FN} \quad (22)$$

Lastly as shown in Equation 23, F1-score is an overall measurement of having low false attack predictions and low false normal predictions.

$$F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (23)$$

For the LSTM block we used one hidden layer consists of 64 hidden units, and for the training we used binary cross entropy as the loss function and Adam as the optimizer[55], for regularization and to avoid over fitting we used dropout layer, drop out technique used only during training and its powerful as it can simulate having various architecture for one model by randomly choosing some nodes to drop out their output during training [56]. For the convolution layer in the CNN model we used the 1Dconv layer from keras with 64 filters of size 3, max pooling filter of size 2. Same setting used for the training process is used including binary cross entropy loss function, Adam optimizer and drop out layer.

6.3. Results

Table 3 and Table 4, present the performance results for all the models, showing the improvement that addition of attention mechanism introduces. Table 3 shows the performance of the models trained and tested on the original dataset without using Autoencoder, where number of features = 25 for each sample, while Table 4 shows the performance of the models trained and tested on the encoded dataset wher number of features = 5 for each sample.

Table 3**Performance of models trained and tested on original dataset with features=25.**

	ALSTM	LSTM	Improvement	ACNN	CNN	Improvement
Accuracy	100%	88%	13.636%	93.83%	84.67%	10.818%
precision	100%	100%	0%	93.53%	92.13%	1.519%
Recall	100%	76.47%	30.77 %	94.44%	76.47%	23.499%
F1-score	100%	86.67%	15.380%	93.98%	83.57%	12.456%

Table 4**Performance of models trained and tested on encoded dataset with features=5.**

	ALSTM	LSTM	Improvement	ACNN	CNN	Improvement
Accuracy	100%	88.67%	12.77%	91.33%	87.25%	4.676%
precision	100%	97.41%	2.658%	99.39%	94.22%	5.487%
Recall	100%	79.9%	25.156%	82.88%	79.9%	3.729%
F1-score	100%	87.79%	13.908%	90.39%	86.47%	4.533%

As shown in Table 3 and Table 4, first of all we can notice that introducing the attention mechanism has boosted the performance of all the models regardless of the used dataset. We can say that the LSTM model benefits more than the CNN model from the additive of the attention mechanism, the two models have different way of looking at the features and it seems that we have more details on the temporal point of view that the attention can exploit better using LSTM than the spatial details produced by the CNN model.

To discuss the effect of encoding the dataset on the performance, we noticed that the use of encoded dataset has different effect on the performance of each model, the ALSTM model was robust to the change and still give superior performance, there was a slight drop on some performance measures of the LSTM and the ACNN models, on the other hand we noticed an enhancement on the performance of the CNN model. In general when comparing the performance of the models with each others on both datasets we can say that encoded dataset did not changed their ranking; the ALSTM model has the best performance followed by the ACNN then the LSTM and finally the CNN. We can conclude from Table 3 and Table 4, that the LSTM model combined with the attention mechanism has an outstanding performance due the ability of the LSTM

of connecting between events happening at different times and the ability of the attention to focus on the significant connections.

Figure 8 and Figure 9, show the confusion matrix for the model LSTM with the attention mechanism (ALSTM), and the confusion matrix for the model LSTM without attention mechanism. While Figures 10 and 11, show the confusion matrix for the model CNN with the attention mechanism (ACNN), and the confusion matrix for the model CNN without attention mechanism. All tested on 1200 samples having 588 normal and 612 attack samples. According to the confusion matrices of the CNN models in Figure 10 and Figure 11, CNN models in general have a higher false alarms compared to the LSTM models.

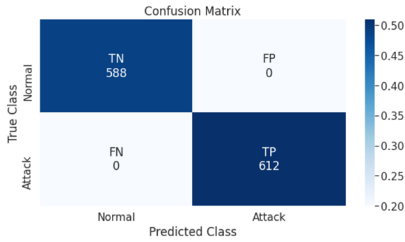


Figure 8

ALSTM model confusion matrix.

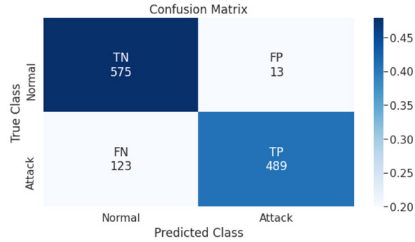


Figure 9

LSTM confusion matrix.

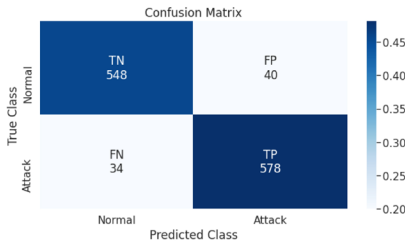


Figure 10

Confusion matrix for CNN with the Attention mechanism.

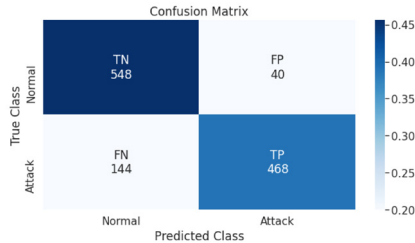


Figure 11

CNN confusion matrix without the Attention mechanism.

For the Autoencoder we used a simple model with the encoding dimension set to 5 and using the same training settings. The results in Table 5 shows reduction in training time up to 66.25% using the encoded dataset for ALSTM model, also the size of the dataset was reduced by 67%.

Table 5
Training time with and without encoding.

Avg Training time for 100 epochs(sec)	ALSTM	LSTM	ACNN	CNN
Original Dataset	226.04	119.83	151.48	129.17
Encoded Dataset	76.27	42.66	41.95	27.59
Reduction in training time(%)	66.25	64.39	72.30	78.64

7. Conclusions and Future Work

Internet of Things (IoT) is considered as the third wave of the Internet represented in an enormous amount of devices being connected to the Internet and generating massive amount of data. These characteristics along with lacking to powerful security measures and having several types of vulnerabilities had drawn the attention of the attackers to compromise the IoT in building networks of enslaved devices (botnet) through which they can generate destructive DDoS attacks.

Detecting of IoT botnet based DDoS attacks is an area of growing concern because of their rapid development in complexity and frequency and their crucial economical consequences. In this paper, we investigated the effect of combining the recent deep learning attention mechanism with two most popular and profound deep learning models; the LSTM and the CNN models. CNN is one of the most popular deep learning models proven to be so efficient for problems involving image data, the fact that CNNs used with other types of data also achieved state of art results encouraged us to examine its performance against our problem, its known for having kernel filters that allow it to extract fine features and recognize the spatial and hidden order relationships within the data, while LSTM models have the ability to learn relationship between events happening at different time by sharing parameters from previous time steps which allow the previous information to persist working as a memory. Attention mechanism was inspired by the human attention in which we focus on certain parts of an input image or audio, while ignoring idle details or receiving them with less focus. Attention layer used in deep neural networks tends to vectorize each input to represent relationships in the input data.

Results demonstrated that introducing the attention mechanism can improve the performance of both models, the LSTM model had a greater improvement when combined with the attention mechanism compared

with the CNN model. The ALSTM model achieved an Accuracy of 100% due to combining the memorizing ability by the LSTM and the focusing ability of the attention in one model, CNN models in general have a higher false alarms compared to the LSTM models.

We also investigated the effect of using an AutoEncoder model for feature reduction to reduce amount of data needed to be exchanged for the purpose of training and to expedite the process of updating the models especially for distributed solution deployed using fog computing. Conducted comparisons show the reduction in training time was up to 66.25% using the encoded dataset for ALSTM model, and reduction in amount data when using an Autoencoder for feature reduction applied on the dataset up to 67%. Using an encoded dataset did not changed the ranking of the performance for the models, the ALSTM model has the best performance followed by the ACNN then the LSTM and finally the CNN. As a future work, we plan to train the models using more than one dataset and to use the deep learning models for multi-classification to detect different types of DDoS attacks.

References

- [1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, M. Ayyash, Internet of things: A survey on enabling technologies, protocols, and applications, *IEEE Communications Surveys Tutorials* 17, 2347–2376 (2015). doi:10.1109/COMST.2015.2444095.
- [2] S. Analytics, Number of internet of things (iot) connected devices worldwide in 2018, 2025 and 2030 (in billions) (2019).
- [3] I. Lee, K. Lee, The internet of things (iot): Applications, investments, and challenges for enterprises, *Business Horizons* 58, 431–440 (2015).
- [4] S. Sicari, A. Rizzardi, L. A. Grieco, A. Coen-Porisini, Security, privacy and trust in internet of things: The road ahead, *Computer networks* 76, 146–164 (2015).
- [5] M. Hron, Are smart homes vulnerable to hacking?, URL <https://blog.avast.com/mqtt-vulnerabilitieshacking-smart-homes> (2018).
- [6] B. Krebs, Krebs on security hit with record ddos, *KrebsOnSecurity*, Sept 21 (2016).
- [7] C. Kolias, G. Kambourakis, A. Stavrou, J. Voas, Ddos in the iot: Mirai and other botnets, *Computer* 50, 80–84 (2017).

- [8] E. Bertino, N. Islam, Botnets and internet of things security, *Computer* 50, 76–79 (2017).
- [9] G. Caspi, Introducing deep learning: Boosting cybersecurity with an artificial brain (2017).
- [10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: *Advances in neural information processing systems*, pp. 5998–6008 (2017).
- [11] E. A. McCullough, Lightweight Deep Learning for Botnet DDoS Detection on IoT Access Networks, Master's thesis, Missouri State University, USA (2020).
- [12] V. H. Bezerra, V. G. T. da Costa, S. Barbon Junior, R. S. Miani, B. B. Zarpel'ao, Iotds: A one-class classification approach to detect botnets in internet of things devices, *Sensors* 19, 3188 (2019).
- [13] K. Bhardwaj, J. C. Miranda, A. Gavrilovska, Towards iot-ddos prevention using edge computing, in: *{USENIX} Workshop on Hot Topics in Edge Computing (HotEdge 18)*, pp. 1–7(2018).
- [14] A. Wang, W. Chang, S. Chen, A. Mohaisen, Delving into internet ddos attacks by botnets: characterization and analysis, *IEEE/ACM Transactions on Networking* 26, 2843–2855 (2018).
- [15] G. Kambourakis, C. Kolias, A. Stavrou, The mirai botnet and the iot zombie armies, in: *MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM)*, IEEE, pp. 267–272 (2017).
- [16] K. Scarfone, P. Mell, et al., Guide to intrusion detection and prevention systems (idps), NIST special publication 800, 94 (2007).
- [17] E. Khoshhalpour, H. R. Shahriari, Botrevealer: Behavioral detection of botnets based on botnet lifecycle, *The ISC International Journal of Information Security* 10, 55–61 (2018).
- [18] I. Butun, B. Kantarci, M. Erol-Kantarci, Anomaly detection and privacy preservation in cloud-centric internet of things, in: *2015 IEEE International Conference on Communication Workshop (ICCW)*, IEEE, pp. 2610–2615 (2015).
- [19] D. Midi, A. Rullo, A. Mudgerikar, E. Bertino, Kalis—a system for knowledge-driven adaptable intrusion detection for the internet of things, in: *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, IEEE, pp. 656–666 (2017).
- [20] K. Kumar, R. Joshi, K. Singh, A distributed approach using entropy to detect ddos attacks in isp domain, in: *2007 International Conference*

- on Signal Processing, Communications and Networking, IEEE, pp. 331–337 (2007).
- [21] S. Yu, W. Zhou, Entropy-based collaborative detection of ddos attacks on community networks, in: 2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications (Per-Com), IEEE, pp. 566–571 (2008).
 - [22] A. Lakhina, M. Crovella, C. Diot, Mining anomalies using traffic feature distributions, ACM SIGCOMM computer communication review 35, 217–228 (2005).
 - [23] B. Al-Duwairi, W. Al-Kahla, M. A. AlRefai, Y. Abdelqader, A. Rawash, R. Fahmawi, Siem-based detection and mitigation of iot-botnet ddos attacks, *International Journal of Electrical & Computer Engineering* (2088-8708) 10 (2020).
 - [24] M. M. Shurman, O. M. Al-Jarrah, S. B. Esoh, S. H. Alnabelsi, An enhanced cross-layer approach based on fuzzy-logic for securing wireless ad-hoc networks from black hole attacks, *International Journal on Communications Antenna and Propagation (IRECAP)* (2017).
 - [25] Z. Xia, S. Lu, J. Li, J. Tang, Enhancing ddos flood attack detection via intelligent fuzzy logic, *Informatica* 34 (2010).
 - [26] S. Hemalatha, S. Qaiyum, S. L. A. Haleem, et al., Protection from distributed denial of service attack using fuzzy logic (2021). <https://www.researchsquare.com/article/rs-349667/v1>.
 - [27] H. Rahmani, N. Sahli, F. Kammoun, Joint entropy analysis model for ddos attack detection, in: 2009 Fifth International Conference on Information Assurance and Security, volume 2, IEEE, pp. 267–271 (2009).
 - [28] Z. Al-Qudah, B. Al-Duwairi, O. Al-Khaleel, Ddos protection as a service: hiding behind the giants, *International Journal of Computational Science and Engineering* 9, 292–300 (2014).
 - [29] P. Kamboj, M. C. Trivedi, V. K. Yadav, V. K. Singh, Detection techniques of ddos attacks: A survey, in: 2017 4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics (UPCON), IEEE, pp. 675–679 (2017).
 - [30] M. F. B. Abbas, T. Srikanthan, Low-complexity signature-based malware detection for iot devices, in: International Conference on Applications and Techniques in Information Security, Springer, pp. 181–189 (2017).

- [31] H. Wang, J. Gu, S. Wang, An effective intrusion detection framework based on svm with feature augmentation, *Knowledge-Based Systems* 136, 130–139 (2017).
- [32] P. Barford, J. Kline, D. Plonka, A. Ron, A signal analysis of network traffic anomalies, in: *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement*, pp. 71–82 (2002).
- [33] K. A. Bradley, S. Cheung, N. Puketza, B. Mukherjee, R. A. Olsson, Detecting disruptive routers: A distributed network monitoring approach, *IEEE network* 12, 50–60 (1998).
- [34] S. T. Zargar, J. Joshi, D. Tipper, A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks, *IEEE Communications Surveys & Tutorials* 15, 2046–2069 (2013).
- [35] A. Patcha, J.-M. Park, An overview of anomaly detection techniques: Existing solutions and latest technological trends, *Computer networks* 51, 3448–3470 (2007).
- [36] S. Basumallik, R. Ma, S. Eftekharinejad, Packet-data anomaly detection in pmu-based state estimator using convolutional neural network, *International Journal of Electrical Power & Energy Systems* 107, 690–702 (2019).
- [37] K. Fu, D. Cheng, Y. Tu, L. Zhang, Credit card fraud detection using convolutional neural networks, in: *International Conference on Neural Information Processing*, Springer, pp. 483–490 (2016).
- [38] M. Nasr, A. Bahramali, A. Houmansadr, Deepcorr: Strong flow correlation attacks on tor using deep learning, in: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1962–1976 (2018).
- [39] F. Feng, X. Liu, B. Yong, R. Zhou, Q. Zhou, Anomaly detection in ad-hoc networks based on deep learning model: A plug and play device, *Ad Hoc Networks* 84, 82–89 (2019).
- [40] A. A. Ahmed, W. A. Jabbar, A. S. Sadiq, H. Patel, Deep learning-based classification model for botnet attack detection, *Journal of Ambient Intelligence and Humanized Computing*, 1–10 (2020).
- [41] G. D. L. T. Parra, P. Rad, K.-K. R. Choo, N. Beebe, Detecting internet of things attacks using distributed deep learning, *Journal of Network and Computer Applications* 163, 102662 (2020).
- [42] M. Alsoufi, S. Razak, M. M. Siraj, A. Ali, M. Nasser, S. Abdo, et al., Anomaly intrusion detection systems in iot using deep learning

- techniques: A survey, in: *International Conference of Reliable Information and Communication Technology*, Springer, pp. 659–675 (2020).
- [43] M. Delakis, C. Garcia, text detection with convolutional neural networks., in: *VISAPP (2)*, pp. 290–294 (2008).
 - [44] H. I. Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P.-A. Muller, F. Petitjean, Inception-time: Finding alexnet for time series classification, *Data Mining and Knowledge Discovery* 34, 1936–1962 (2020).
 - [45] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, M. Hasan, B. C. Van Essen, A. A. Awwal, V. K. Asari, A state-of-the-art survey on deep learning theory and architectures, *Electronics* 8, 292 (2019).
 - [46] I. Aljarrah, D. Mohammad, Video content analysis using convolutional neural networks, in: *2018 9th International Conference on Information and Communication Systems (ICICS)*, IEEE, pp. 122–126 (2018).
 - [47] X. Zhang, J. Zhao, Y. LeCun, Character-level convolutional networks for text classification, *arXiv preprint arXiv:1509.01626* (2015).
 - [48] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, *Advances in neural information processing systems* 25, 1097–1105 (2012).
 - [49] S. Hochreiter, The vanishing gradient problem during learning recurrent neural nets and problem solutions, *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6, 107–116 (1998).
 - [50] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, *IEEE transactions on neural networks* 5, 157–166 (1994).
 - [51] G. I. Winata, O. P. Kampman, P. Fung, Attention-based lstm for psychological stress detection from spoken language using distant supervision, in: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, pp. 6204–6208 (2018).
 - [52] A. Samy, H. Yu, H. Zhang, Fog-based attack detection framework for internet of things using deep learning, *IEEE Access* 8, 74571–74585 (2020).
 - [53] N. Koroniotis, N. Moustafa, E. Sitnikova, 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* 100, 779–796 (2019).

- [54] S. Patro, K. K. Sahu, Normalization: A preprocessing stage, *arXiv preprint arXiv:1503.06462* (2015).
- [55] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, *arXiv preprint arXiv:1412.6980* (2014).
- [56] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, *The Journal of Machine learning research* 15, 1929–1958 (2014).

Received August, 2022