IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 10, No. 1, March 2021, pp. 110~120
ISSN: 2252-8938, DOI: 10.11591/ijai.v10.i1.pp110-120

Toward a deep learning-based intrusion detection system for IoT against botnet attacks

Idriss Idrissi[1], Mohammed Boukabous[2], Mostafa Azizi[3], Omar Moussaoui[4], Hakim El Fadili[5]
[1,2,3,4]MATSI Research Lab., ESTO, Mohammed First University, Oujda, Morocco
[5]LIPI Research Lab., ENSAF, Sidi Mohamed Ben Abdellah University, Fez, Morocco

Article Info

Article history:

Received Jun 9, 2020
Revised Dec 30, 2020
Accepted Feb 2, 2021

Keywords:

Bot-IoT
Botnet
CNN
DL
IDS
IoT
RNN

ABSTRACT

The massive network traffic data between connected devices in the internet of things have taken a big challenge to many traditional intrusion detection systems (IDS) to find probable security breaches. However, security attacks lean towards unpredictability. There are numerous difficulties to build up adaptable and powerful IDS for IoT in order to avoid false alerts and ensure a high recognition precision against attacks, especially with the rising of Botnet attacks. These attacks can even make harmless devices becoming zombies that send malicious traffic and disturb the network. In this paper, we propose a new IDS solution, baptized BotIDS, based on deep learning convolutional neural networks (CNN). The main interest of this work is to design, implement and test our IDS against some well-known Botnet attacks using a specific Bot-IoT dataset. Compared to other deep learning techniques, such as simple RNN, LSTM and GRU, the obtained results of our BotIDS are promising with 99.94% in validation accuracy, 0.58% in validation loss, and the prediction execution time is less than 0.34 ms.

Corresponding Author:

Idriss Idrissi
MATSI Laboratory, ESTO
Mohammed First University
BP 473 Campus universitaire Al Qods, Oujda 60000, Morocco
Email: idrissi@ump.ac.ma

1. INTRODUCTION

Nowadays an enormous number of objects are dispatched around the world and are connected between them and to the internet. They vary from personal gadgets, wearables, sensors, actuators to home appliances and medical devices. As estimated by CISCO in 2025, it would be somewhere 75 billion devices connected to the Internet [1]. The IoT has raised concerns that are growing rapidly without fitting thought of the significant security challenges [2].

Nowadays, most of the security concerns are like those of regular servers, workstations and smartphones; however, security moves extraordinarily to the IoT, including mechanical security controls, hybrid frameworks, IoT-explicit business procedures, and edge devices [3]. Traditional security protection technology is limited due Zero-Day attacks and vulnerabilities and future new attacks that are continuously changing nature; setting up a steady, reliable, and precise intrusion detection is becoming mandatory for improving the IoT security [4].

Botnets or zombies are robots of infected internet-connected devices, they are used to achieve distributed denial-of-service attack (DDoS attack), password cracking, key logging (steal data), cryptocurrency mining, and give the attacker the possibility of accessing the device using command and control (C&C) software [5]. In September 2016, “Mirai” malware, an IoT Botnet attacked many sites offline like the cloud service provider “OVH” with nearly 1.1 TBps, the website of computer security consultant Brian Krebs with 620 Gbps of traffic, and many other websites like dynamic DNS provider “Dyn” [6].

The detection and prevention from different attacks are a big challenge. IDS using machine-learning methods, has gained a wide reputation [7]. IDS is an essential component in the security mechanism, it is used for the analysis and detection of the security breaches on a network [8]. IDS systems can be gathered into two categories: the first one “anomaly detection” and the second is “Misuse detection”, or gathered into three major distinct families “host-based”, “network-based” and “hybrid”. IDS investigate both traffic in the network and in the operating systems. IDS are used for effective network protection. Numerous research works are trying to apply data mining and machine learning algorithms to cyber security. In Machine learning, patterns recognition and data mining algorithms are extensively applied to distinguish the normal traffic from the malicious one.

2. ARTIFICIAL NEURAL NETWORKS (ANN)

In last recent years, one of the most resulting and efficient subsets of artificial intelligent is deep learning (DL) which it is also a subset of machine learning based on artificial neural networks (ANN) [9]; a computing system inspired from biological brain where the machine learns from many training examples, allowing it to classify other examples [10]. DL is increasingly being used. It can be applied in many data processing layers into a hierarchical architecture to make a deep model. DL accounts on its capacity to identify ideal features in raw data through successive nonlinear transformations, with every alteration achieving a more elevated level of complexity and abstraction [10]. It has been applied efficiently to many different research fields, from medical image processing, natural language processing, speech recognition, and signal recognition to many other domains of science, business and government. In all these fields, DL showed tremendously promising results [11]. In the IoT security field, the machine trains on various collected and labeled attacks and also normal traffic to learn them, were finally this machine can identify new similar attacks.

Convolutional neural networks (CNN or ConvNets): it’s a deep learning class developed in 1998 by LeCun in the LeNet architecture [12]. In recent two decades, CNN gained big success. It’s composed of an input layer, many hidden layers in between, and an output layer as shown in Figure 1 like the multi layer perceptron (MLP) [13] networks. Best known and used layers are: convolution, activation or ReLU, and pooling [14]. The convolutional layer is the most important one, it takes a convolution kernel also called a mask or a filter then pass it over the data (usually images) and transform it based on the values from the filter as shown in Figure 1. Then it calculates the feature map values using the formula (1), where the input data is represented by f and the kernel by h, m and n are respectively the indexes of rows and columns of the resultant matrix [15].

The pooling layer it is what achieves progressively down sampling to reduce the size of the succeeding layers through max pooling or average pooling to help overfitting. Max pooling divides the input into non-overlapping clusters and selects the maximum value for each cluster in the previous layer [16].

$$G[m,n] = (f * h)[m,n] = \sum_j \sum_k h[j,k] f[m-j, n-k] \qquad (1)$$

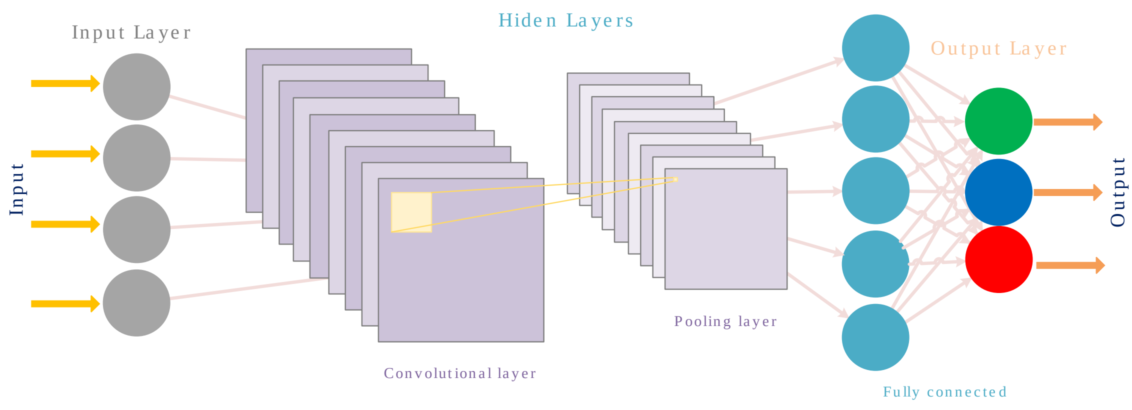


Figure 1. CNN layers

Contrariwise to the traditional feature selection algorithms it has the capability of learning better features automatically and categorize the traffic. In addition, it can achieve better classification and learn additional features with more traffic data because it shares the same convolution matrix (kernel), that would decrease the number of parameters and calculation sum of training significantly. This gives CNN a fast recognition of attack nature, contrariwise to other deep-learning algorithms, or machine learning algorithms that can be over-fitted with massive big data. Moreover, the literature shows that using CNNs in intrusion detection field gives better results than other algorithms [17-18].

Recurrent neural networks (RNN) are a class of deep neural networks that contains feedback connections as shown in Figure 2. The fully connected layer works on a flattened input where each of these inputs is connected to all neurons. The activation function of a node describes its output given a one or set of inputs. Rectified linear unit (ReLU) activation as shown in Figure 3(a), it is a ReLU used on all elements of the volume. It aims at introducing non-linearities to the network. LSTM is composed of memory blocks that are a set of recurrent connected subnetworks. These blocks are composed with a self-connected memory cells (one or many) as shown in Figure 4 which offer a memory to remember the previous data, and three units called gates: input gate (3.a), a forget gate (3.b) and an output gate (3.c) which they provide a continuous equivalent of write, read and reset operations [21]. These gates are sigmoid as shown in Figure 3(b) and tanh as shown in Figure 3(c) activation functions meaning that their output is a value between 0 and 1 for sigmoid, and between -1 and 1 for tanh. Derived from feedforward neural networks (FNN) but unlike FNN, there are loops (bidirectional data flow) and memories to remember previous computations as shown in Figure 4 [19]. And allowing preceding outputs to be used as inputs while having hidden states [20] where for each timestep t , the activation $a^{<t>}$ and the output $y^{<t>}$are expressed:

$$\left\{\begin{array}{l} a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \quad (2.a) \\ \quad and \quad y^{<t>} = g_2(W_{ya}a^{<t>} + b_y) \quad (2.b) \end{array}\right\} \qquad (2)$$

Where W_{ax}, W_{aa}, W_{ya}, b_a, b_y are coefficients that are shared temporally and g_1, g_2 activation functions

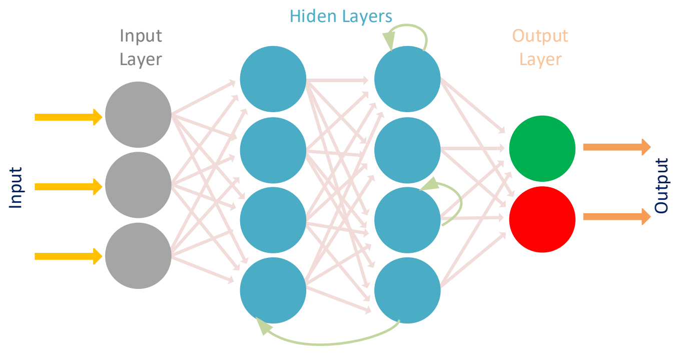


Figure 2. Recurrent neural networks layers

RNN can face the long-term dependency problem and the vanishing gradient & exploding gradient, so we cite here the best known RNN networks, the long short-term memory (LSTM) and gated recurrent unit (GRU) networks to solve these problems. The main difference to simple RNN is that the nonlinear units in the hidden layers are replaced by memory blocks. The following formula (3) represents the gates in LSTM [22].

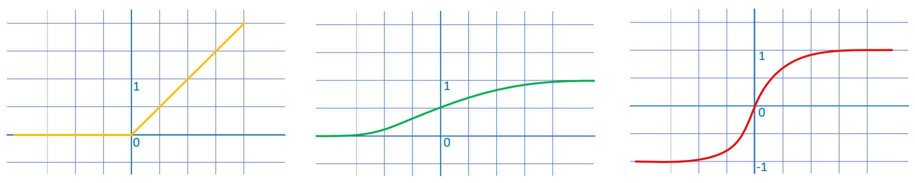
$$\left\{\begin{array}{l} i_t = \sigma(\omega_i[h_{t-1}, x_t] + b_i \ (3.a) \\ f_t = \sigma(\omega_f[h_{t-1}, x_t] + b_f \ (3.b) \\ o_t = \sigma(\omega_o[h_{t-1}, x_t] + b_o \ (3.c) \end{array}\right\} \qquad (3)$$

Where: x_t are the input gates (“i” for the input gate, “f” for the forget, and “o” for the output gate);
“σ”it is the sigmoid function;
“b_i” it is the biases for the gate(x);

"h_{t-1}" it is the output of the precedent LSTM block;
"x_t" it is the current input.

Gated recurrent unit (GRU) is a simplified version of LSTM, where the GRU modulates the flow of information inside the unit using gating units as shown in Figure 4, without separating the memory cells [23-24]. It merges the forget and the input gates into an "update gate", also merges cell and hidden state, GRU has fewer parameters than the LSTM. It is defined by the following formulas.

$$\left\{\begin{matrix} r_t = sigm(W_{xr}x_t + W_{hr}h_{t-1} + b_r)\ (4.a) & z_t = sigm(W_{xz}x_t + W_{hz}h_{t-1} + b_z)\ (4.b) \\ \tilde{h}_t = tanh(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h)\ (4.c) & h_t = z_t \odot h_{t-1} + (1 + z_t) \odot \tilde{h}_t\ (4.d) \end{matrix}\right\} \quad (4)$$



$f(x) = \begin{cases} 0 & \text{for } x \le 0 \\ x & \text{for } x > 0 \end{cases}$ (a)

$f(x) = \sigma(x) = \dfrac{1}{1 + e^{-x}}$ (b)

$f(x) = \tanh(x) = \dfrac{(e^x - e^{-x})}{(e^x + e^{-x})}$ (c)

Figure 3. Activation functions: (a) ReLU, (b) Sigmoid, (c) Tanh

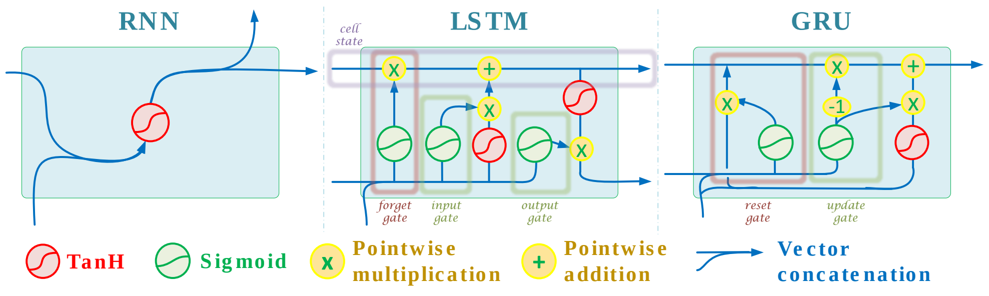


Figure 4. RNN, LSTM & GRU blocks [25]

3. RELATED WORKS

Koroniotis *et al*. [26] applied support vector machine (SVM), LSTM and RNN to evaluate the IoT-Dataset. The authors focused on binary classification on the dataset, and their prediction result was either a "normal traffic" or "some type of attacks" (for every type of attack), which is not helpful for implementing many models (for every attack type) to a working IDS contrary to a multi-label output (numerous categories of attacks) that gives one and only one model.

Ibitoye *et al*. [27] in their research compared the performance between two deep learning models self normalizing networks (SNN) and feed forward neural networks (FNN) inside the milieu of an IoT network. This comparison shows that FNN outperforms SNN, even if SNN remains better in regards to adversarial samples. Also, the authors examined the impact of feature normalization on the adversarial strength and demonstrated its bad influence to adversarial attacks resisting.

Ferrag *et al*. [28] in his paper conducted a comparative study with two datasets, Bot-IoT and CSE-CIC-IDS2018 datasets using some deep learning approaches, such RNN, CNN, Boltzmann machine, deep belief networks (DBN), deep Boltzmann machines (DBM), deep autoencoders and deep discriminative models, with 100 hidden layers to get an accuracy of 98.394%.

Mengmeng [29] proposed using FNN an intelligent binary and multiclass classification, but with just few classes to get 99% in all evaluation measures (accuracy, precision, recall and F1 score) for DDoS/DoS attacks while the normal traffic classification got an accuracy of 98%.

AlKadi [30] proposed a system named mixture localization-based outliers (MLO) on the BoT-IoT Dataset that uses utilizes gaussian-mixture models for fitting network data and a local outlier factor function for discovering abnormal patterns in network traffic data, and gotten an accuracy of 97.98%.

In fact, convolutional neural networks (CNN or ConvNets) are a class of deep neural networks that are used in many fields but mostly in pattern recognition. CNN is a class of neural networks that uses the convolution and the pooling layers instead of the fully connected hidden layers [31]. Contrariwise to the traditional feature selection algorithms it has the capability of learning better features automatically and categorize the traffic. In addition, it can achieve better classification and learn additional features with more traffic data because it shares the same convolution matrix (mask), that would decrease the number of parameters and calculation sum of training significantly. This gives CNN a fast recognition of attack nature, contrariwise to other deep-learning algorithms, or machine learning algorithms that can be over-fitted with massive big data. Moreover, the literature shows that using CNNs in intrusion detection field gives better results than other algorithms [17-18].

The related work listed above can provide a good prediction of botnet attacks that can affect an IoT system. However, these works could not recognize type of attack due to the binary classification. Hence our study constitutes an important experimental extension of the above-mentioned works by benchmarking the Bot-IoT dataset using CNN compared to different deep learning models, using the multilabel classification corresponding to various categories of attacks in the IoT.

4. PROPOSED METHOD

BotIDS is our proposed network IDS obtained by learning from the Bot-IoT dataset. This solution is planned to be placed in a fog node when it will be implemented in a real IoT environment. A such deployment gives it the power of analyzing in real time the inbound and outbound traffic through the network by sniffing it as shown in Figure 5. This location will make our BotIDS able to monitor all traffic to/from the devices both inside and outside the network, and even particularly the traffic between this insider devices in case of a lurking zombie device inside the network.

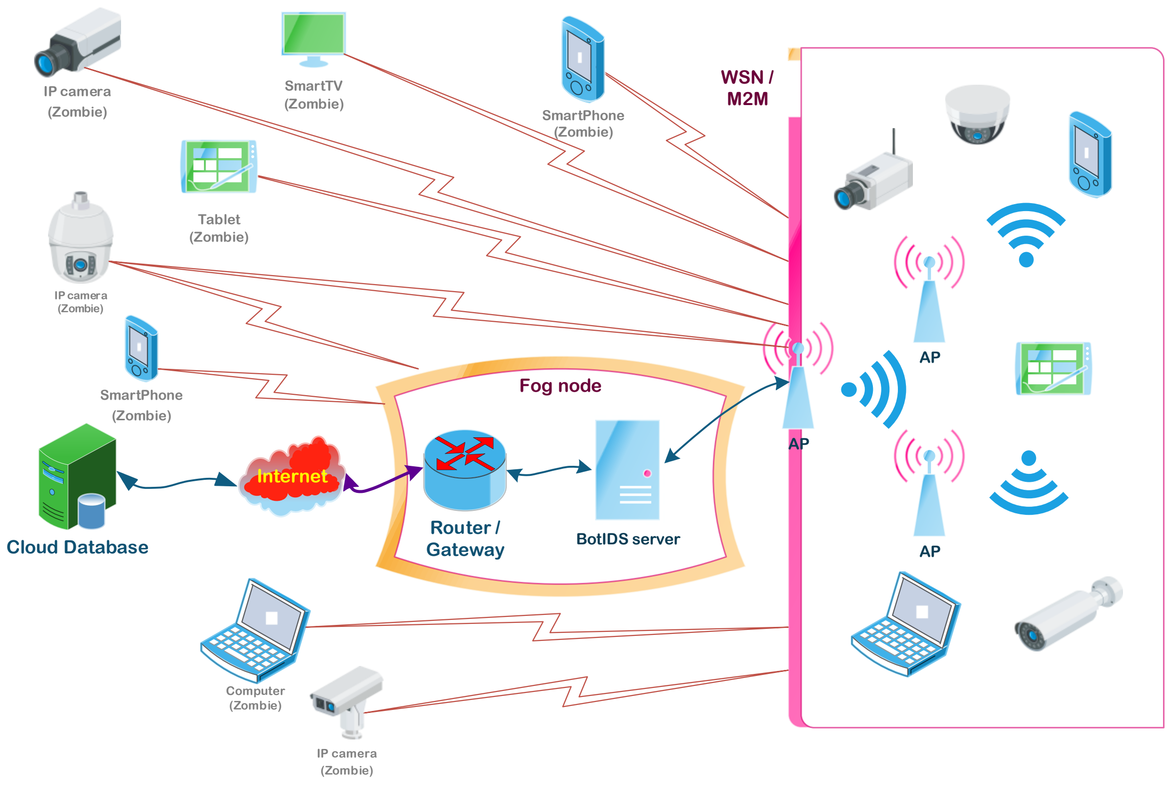


Figure 5. Architecture of proposed approach

The BotIDS is a deep learning-based method on a deep learning model that contains three phases: as shown in Figure 6

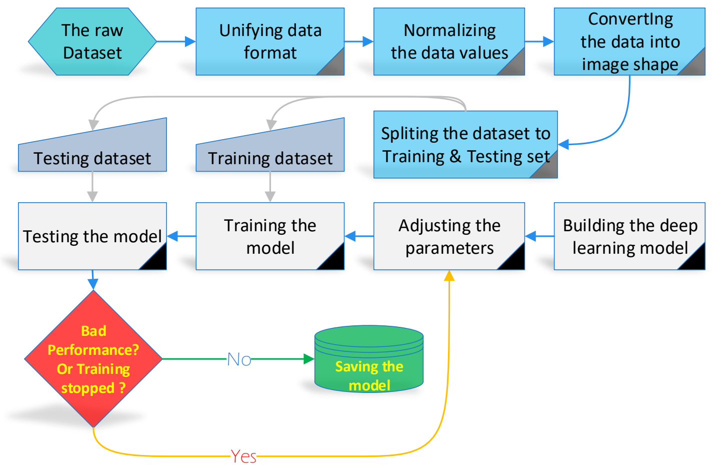

The raw Dataset
Unifying data format
Normalizing the data values
Converting the data into image shape
Testing dataset
Training dataset
Spliting the dataset to Training & Testing set
Testing the model
Training the model
Adjusting the parameters
Building the deep learning model
Bad Performance? Or Training stopped ?
No
Saving the model
Yes

Figure 6. Model building process

- 1st Phase: Dataset preprocessing; First of all, we need to alter the raw data and normalize its values with the goal of the best performance of deep learning model, and then convert it into image shape.
- 2nd Phase: Building the model; the model is at first fit on a training dataset (a part of the dataset) using parameters to achieve the improvement of the model performance, these parameters are changed in the training process to reach better performance. Secondly, the test dataset (the remaining part of the dataset) is used to validate the accuracy of the model.
- 3rd Phase: Evaluating the model by prediction; after building and generating the model, we evaluate this model with the test dataset by predicting attacks and calculating the time needed for this prediction.

4.1. Dataset preprocessing

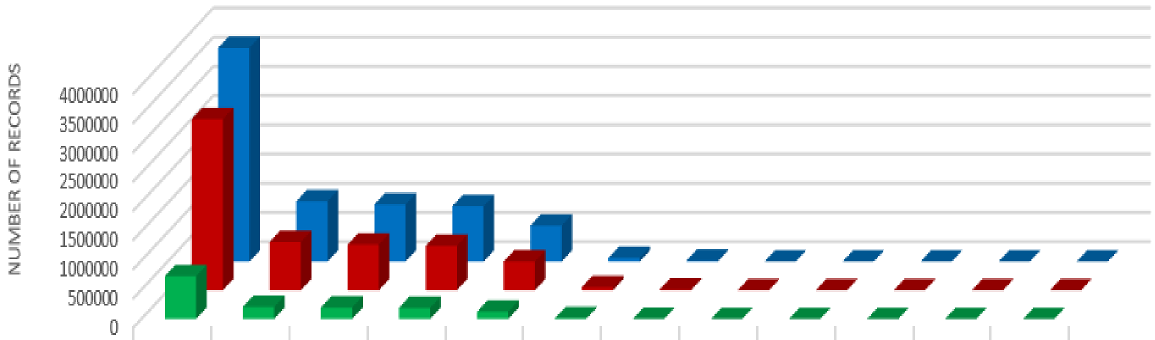
For conducting proposed work, we have used the latest Bot-IoT dataset [32] that was created specifically for IoT systems by an actual network milieu at the Cyber Range Lab of the Center of UNSW Canberra Cyber. The environment incorporates a combination of usual normal and bad traffic, with six types of attacks and 10 subcategories, namely, reconnaissance (service scanning and OS fingerprinting), DDoS (TCP, UDP and HTTP), DoS (TCP, UDP and HTTP), theft (key logging and data exfiltration).

With 72 million records of data traffic simulated IoT environment. The whole data was ascended down to 5% into a “full-feature” dataset with around 3.6 million records and another version called “10 best features” is also provided with selection of best features from the “Full features” version, both versions are used for our experiment. The training and test dataset have 11 output classes which reflect the normal traffic, and the 10 types of attacks which were carried out against the IoT network.

The Bot-IOT dataset contains network connection attributes; nominal, numeric and IP addresses. We convert the nominal data to numeric data, ipv4 and ipv6 also be converted to numerical shape, and merging category and subcategories fields into one field that contains 10 types of attacks and the 11th is a normal traffic then we convert the new category attribute using “one-hot encoding”, and dropped the binary “Attack” field cause. Our focus is on a multilabel output and not a binary one.

After encoding the data, we normalized it using Scikit-learn; meaning scaling the vectors individually to unit norm, and converting the normalized output data into image data shape.

Then we split the data at first into data X (contains all the features except the “category” feature) and label Y (contains the “category” feature), and then split it into random training subset and testing subset with 75% for training set and 25% for the testing set. Figure 7 shows the number of data rows for each set and each attack type.



	Total	DoS UDP	DDoS TCP	DDoS UDP	DoS TCP	Service_Scan	OS_Fingerprint	DoS HTTP	DDoS HTTP	Normal Trafic	Keylogging	Data_Exfiltration
🟩 Testing set	731894	206485	195911	189075	123219	12929	3679	293	203	92	7	1
🟥 Training set	2927572	826475	781468	759163	492581	51351	14101	1191	786	385	66	5
🟦 Dataset	3659466	1032960	977379	948238	615800	64280	17780	1484	989	477	73	6

Figure 7. Attacks distribution in training and testing sets

4.2. Building our models

The CNN models were defined to have an input layer with the number of neurons equal to the amount of input features, four hidden layers Convolution2D layer, MaxPooling2D layer, Flatten layer, Dense layer and an output layer. For the best features dataset, the model was trained in 10 epochs (the whole dataset is passed through the neural network 10 times) with batch size of 32 (amount of training simples in a single batch is 32) and a kernel size of (1, 10). The neural network comprises 16 input neurons (in the first layer, the same number as the features), with 4 intermediate (hidden) layers with 32 (Convolution2D), 32 (MaxPooling2D), 480 (Flatten), 22 (Dense) neurons, and 11 output neurons for the multilabel classification as shown in Figure 8. For our full-feature dataset, the model was trained in 15 epochs (batch size of 32 and a kernel size of (1, 10)), and had a 40-neuron input layer, same number and consistency of hidden layers as with the best feature model and 11 output neurons for the multilabel classification. In both cases, for the input and hidden layers, the activation function that was used was ‘Relu’, while the output layer activation function was ‘Softmax’ as shown in Figure 8 and Table 1.

The other recurrent neural network (RNN) models were defined to have one input layer with the number of neurons equal to the amount of input features, four hidden layers and an output layer. For the best features dataset, the model was trained in 40 epochs with 32 in the batch size. The neural network comprised 16 input neurons (in the first layer, the same number as the features), with 4 (SimpleRNN/LSTM/GRU) intermediate (hidden) layers with 32, 64, 128, 22 neurons, and 11 output neurons for the multilabel classification as shown in Figure 9. For our full-feature dataset, the model was trained in 40 epochs (32 in the batch size), and had a 40-neuron input layer, same number and consistency of hidden layers as with the best feature model and 11 output neurons for the multilabel classification. In both cases, for the output layer activation function was ‘Softmax’ as shown in Figure 9, and Table 1.

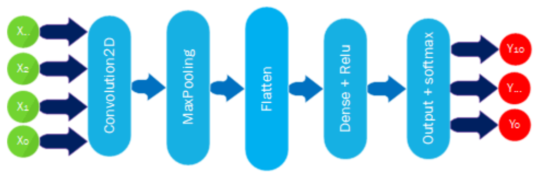


Figure 8. CNN model layers

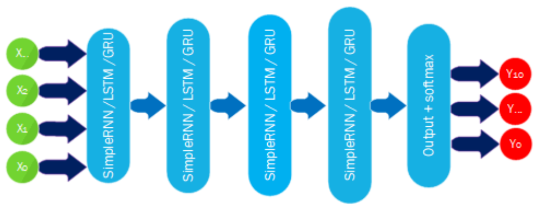


Figure 9. RNN models layers

Table 1. Models parameters for the implemented models

DL Algorithm	Convolutional neural network (CNN)		Simple recurrent neural network (RNN)		Long short-term memory (LSTM) based RNN		Gated recurrent unit (GRU) based RNN	
Dataset version	Full features	Best features	Full features	Best features	Full features	Best features	Full features	Best features
Epochs	15	10	40	40	30	25	40	20
Layers	4	4	4	4	4	4	4	4
Neurons	40 Input Convolutional layer 2D Max pooling shape layer Dense Layer (ReLU) 11 Output	40 Input Convolutional layer 2D Max pooling shape layer Dense Layer (ReLU) 11 Output 11 Output	40 Input 4 SimpleRNN Layers 11 Output	16 Input 4 SimpleRNN Layers 11 Output	40 Input 4 LSTM Layers 11 Output	16 Input 4 LSTM Layers 11 Output	40 Input 4 GRU Layers 11 Output	16 Input 4 GRU Layers 11 Output
Activation function	Hidden layers: Relu Output layers: Softmax		Output layers: 'Softmax'		Output layers: 'Softmax'		Output layers: 'Softmax'	
Optimizer	adam		adam		adam		adam	
Batch size	32		32		32		32	

5. RESULTS AND DISCUSSION

5.1. Hardware characteristics

The results we obtained are performed on a machine (laptop) with following hardware characteristics.

- CPU: Intel i7 8th generation (1 socket, 6 cores, 12 threads)
- RAM: 8 GB
- GPU: NVidia GeForce GTX 1050 with cuda v10.1

In our experiments we worked with Keras (2.2.4); an open-source python deep learning library which is running on top of Google's open-source data flow software; TensorFlow (1.13.1) as a backend engine.

5.2. Evaluating the model

In Figures 10-11 (generated by Tensorboard) and Table 2, we present the accuracy training, loss training, accuracy validation, loss validation along with training time of all models that are trained over several epochs. We consider the number of epochs for which our model reaches the best results for each version of the dataset (full features and best features). We initially tested each model with a batch size of 128, which allowed us to get good timing for all models but with a poor performance, compared to a smaller batch size like 32 records that leads to an improved result but with a higher computation time.

As shown in Figure 10 and Table 2, CNN models in both dataset versions "Full Features" and "Best Features" reaches respectively 99,430 and 99,935 as accuracy in just 1395s and 823s (15 and 10 epochs for each). Compared to our CNN, GRU is a little more accurate but it took more time to be trained (12412s and 5818s in 40 and 20 epochs). The time of our CNN is almost twice compared to simple RNN for the "Full Features" version (6089s in 40 epochs), and almost equal to the "Best Features" version (5708s in 40 epochs). On the other hand, LSTM models have their best accuracy in a timing between simple RNN and GRU (with 10627s and 8302s in 30 and 15 epochs). For the Loss as shown in Figure 11 and Table 2, CNN models are always the best reaching models in both dataset versions, "Full Features" and "Best Features", with respectively 0,582% and 1,663% compared to other RNN models (between 1.7% and 2.9%). By examining these results, it is clear that the CNN model using "Full Features" is the best model according to its higher accuracy (0.9993), and the best time performance when considering the whole time for obtaining these results. In addition, when comparing the resulting metrics in all models with the two-dataset versions, we clearly see that the "Full Features" version gets best performance related to "Best Features" version. This means that deep learning algorithms can achieve better classification and learn additional features with additional traffic data as already explained in the previous section, but in computation time and energy consumption, it consumes more because more data to train are needed.

In Figure 12, we compared the time that a prediction would take for all the considered models (how much time the IDS would take to identify an attack) which could be another important metric for the IDS implementation in a real IoT environment. We then concluded that CNN has not only the best accuracy and the lowest loss, but also the lowest time to predict an attack with just a fragment of milliseconds (with an average of 0.34ms). The other models make a detection of an attack in more than twice or triple that time of CNN (0.78ms for simple RNN, 1.03ms for GRU, and 1.08ms for LSTM).

The amount of data in the dataset is considerably unbalanced regarding the different types of attacks. For example, DoS (HTTP), DDoS (HTTP), key logging, and data exfiltration attacks in the training and test sets have very small amount of data as shown in Figure 7. Therefore, the model has a limited capability to learn accurately these types of attacks. The detection average of these attacks is one of the main factors restricting the overall detection accuracy.

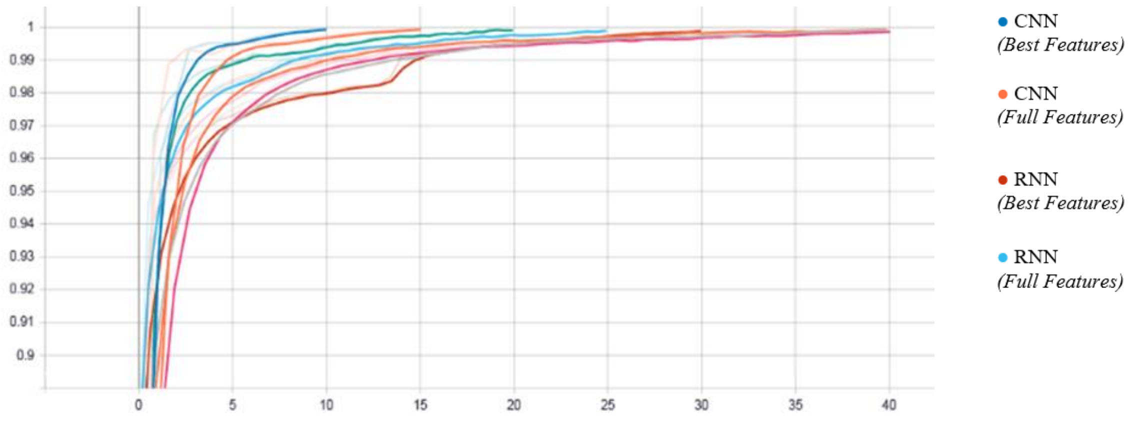


Figure 10. Accuracy scalar

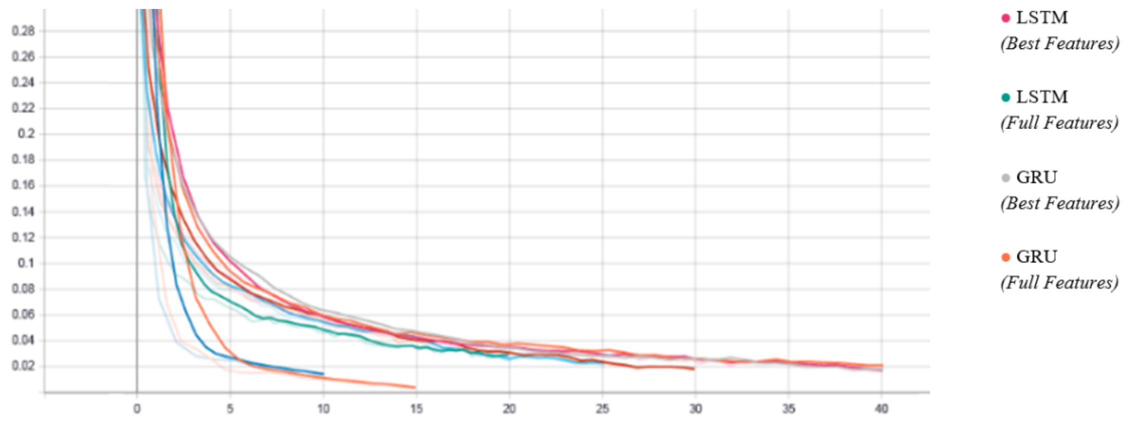


Figure 11. Loss scalar

Table 2. Training and testing results for the implemented models

DL Algorithm	Convolutional Neural Network (CNN)		Simple RNN (Recurrent Neural Network)		Long Short-Term Memory (LSTM) based RNN		Gated Recurrent Unit (GRU) based RNN	
Dataset version	Full Features	Best Features	Full Features	Best Features	Full Features	Best Features	Full Features	Best Features
Epochs	15	10	40	40	30	25	40	20
Layers	4	4	4	4	4	4	4	4
Accuracy training	0.99762	0.99086	0.98167	0.98038	0.98395	0.98285	0.98787	0.98301
Loss training	0.01132	0.03294	0.07101	0.07115	0.06155	0.06526	0.04508	0.06661
Accuracy validation	0.99935	0.99430	0.99406	0.99427	0.99401	0.99747	0.99403	0.99430
Loss validation	0.00582	0.01663	0.02021	0.01819	0.01848	0.02278	0.01762	0.02921
Timing	Total: 1395s Epoch: 138s Step: 47µs	Total: 823s Epoch: 82s Step: 30µs	Total: 6089s Epoch:153s Step: 52µs	Total: 5708s Epoch:142s Step: 52µs	Total:10627s Epoch:354s Step:121µs	Total: 8302s Epoch:332s Step:121µs	Total:12412s Epoch:310s Step:106µs	Total:5818s Epoch: 291s Step:106µs

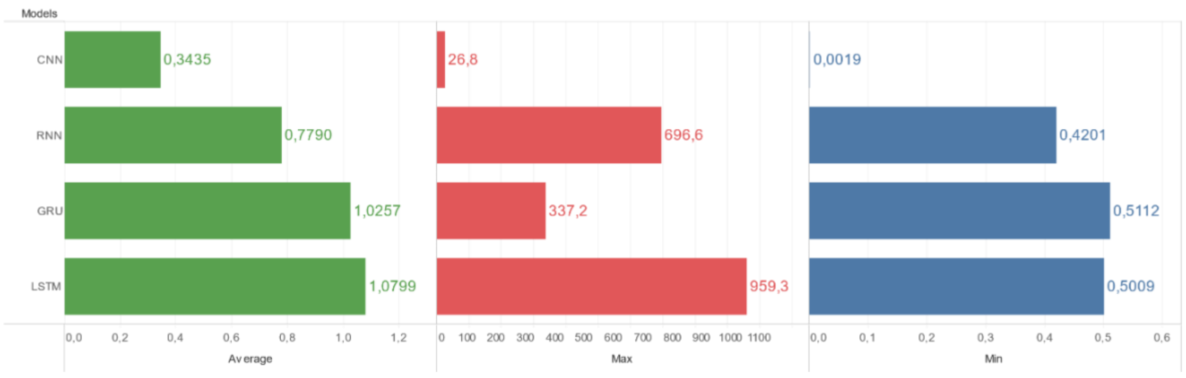


Figure 12. Prediction timing comparison

6. CONCLUSION

The number of IoT objects dispatched around the world is growing progressively, which makes its security a big challenge. Our work here focused on intrusion detection systems for IoT using four variants of deep learning models, and compared them efficiently to detect various types of IoT network attacks, usually done by botnets. After several experiments, we obtained a reasonable detection rate on our all four models. By analyzing the obtained results, we concluded that CNN is the best one for intrusion detection systems, it was able to identify successfully different types of attacks and showed the higher accuracy (with 99.94%) in comparison with other DL algorithms, such as Simple RNN, LSTM, and GRU, and it allowed a detection with lower loss rates (0.58%) and a better performance in terms of prediction time. As future works, we will apply the CNN model on a real network traffic data, and reinforcing it using deep transfer learning by adding other characteristics from other datasets or from diverse firewalls, logs, and IDS servers. In addition, we will try to use self-supervised learning to generate a powerful and updated model, and then implement an autonomous intrusion detection system.

REFERENCES

[1] "Internet of Things (IoT) - The future of IoT miniguide: The burgeoning IoT market continues - Cisco." [Online]. Available: https://www.cisco.com/c/en/us/solutions/internet-of-things/future-of-iot.html. [Accessed: 06-Jun-2020].

[2] J. Singh, T. Pasquier, J. Bacon, H. Ko, and D. Eyers, "Twenty Security Considerations for Cloud-Supported Internet of Things," *IEEE Internet Things J.*, vol. 3, no. 3, pp. 269–284, Jun. 2016. DOI: 10.1109/JIOT.2015.2460333.

[3] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao., "A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1125–1142, 2017. DOI: 10.1109/JIOT.2017.2683200.

[4] Y. Xiao, C. Xing, T. Zhang, and Z. Zhao, "An Intrusion Detection Model Based on Feature Reduction and Convolutional Neural Networks," *IEEE Access*, vol. 7, pp. 42210–42219, 2019. DOI: 10.1109/ACCESS.2019.2904620.

[5] E. Bertino, N. Islam, "Botnets and Internet of Things Security," *Computer (Long. Beach. Calif).*, vol. 50, no. 2, pp. 76–79, 2017. DOI: 10.1109/MC.2017.62.

[6] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas., "DDoS in the IoT: Mirai and other botnets," *Computer (Long. Beach. Calif).*, vol. 50, no. 7, pp. 80–84, 2017. DOI: 10.1109/MC.2017.201.

[7] A. J. Malik, W. Shahzad, and F. A. Khan., "Network intrusion detection using hybrid binary PSO and random forests algorithm," *Secur. Commun. Networks*, vol. 8, no. 16, pp. 2646–2660, 2015. https://doi.org/10.1002/sec.508.

[8] J. Jabez, B. Muthukumar., "Intrusion detection system (ids): Anomaly detection using outlier detection approach," in *Procedia Computer Science*, 2015, vol. 48, no. C, pp. 338–346. DOI: 10.1016/j.procs.2015.04.191.

[9] N. Ouerdi, I. Elfarissi, A. Azizi, and M. Azizi., "Artificial neural network-based methodology for vulnerabilities detection in EMV cards," in *Proceedings of the 2015 11th International Conference on Information Assurance and Security, IAS 2015*, pp. 85–90, 2016. DOI: 10.1109/ISIAS.2015.7492750.

[10] S. Vieira, W. H. L. Pinaya, and A. Mechelli., "Using deep learning to investigate the neuroimaging correlates of psychiatric and neurological disorders: Methods and applications," *Neuroscience and Biobehavioral Reviews*, vol. 74. Elsevier Ltd, pp. 58–75, 2017. DOI: 10.1016/j.neubiorev.2017.01.002.

[11] Y. Lecun, Y. Bengio, and G. Hinton., "Deep learning," *Nature*, vol. 521, no. 7553. Nature Publishing Group, pp. 436–444, 2015. https://doi.org/10.1038/nature14539.

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner., "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. DOI: 10.1109/5.726791.

[13] I. El Farissi, M. Azizi, and M. Moussaoui., "Detection of smart card attacks using neural networks," in *Proceedings of 2012 International Conference on Multimedia Computing and Systems, ICMCS 2012*, pp. 949–954, 2012. DOI: 10.1109/ICMCS.2012.6320286.

[14] "Convolutional Neural Network - MATLAB & Simulink." [Online]. Available: https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html. [Accessed: 05-May-2020].

[15] P. Skalski, "Gentle Dive into Math Behind Convolutional Neural Networks." [Online]. Available: https://towardsdatascience.com/gentle-dive-into-math-behind-convolutional-neural-networks-79a07dd44cf9. [Accessed: 18-May-2020].

[16] 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 Commun. Surv. Tutorials*, vol. 22, no. 3, pp. 1646 - 1685, 2020. DOI: 10.1109/COMST.2020.2988293.

[17] R. Vinayakumar, K. P. Soman, and P. Poornachandrany., "Applying convolutional neural network for network intrusion detection," in *2017 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2017*, 2017, vol. 2017, pp. 1222–1228. DOI: 10.1109/ICACCI.2017.8126009.

[18] W. Tao, W. Zhang, C. Hu, and C. Hu., "A Network Intrusion Detection Model Based on Convolutional Neural Network," in *Advances in Intelligent Systems and Computing*, vol. 895, pp. 771–783, 2020. https://doi.org/10.1007/978-3-030-16946-6_63.

[19] Y. Singh, A. S. Chauhan., "Neural networks in data mining.," *J. Theor. Appl. Inf. Technol.*, vol. 5, no. 1, pp. 36–42, 2009.

[20] A. Afshine and A. Shervine., "CS 230 - Recurrent Neural Networks Cheatsheet," *Stanford.Edu*, 2019. .

[21] M. Ibrahim Sameen, B. Pradhan., "Severity Prediction of Traffic Accidents with Recurrent Neural Networks," *Appl. Sci.*, vol. 7, no. 6, p. 476, 2017.

[22] D. Thakur, "LSTM and its equations - Medium." [Online]. Available: https://medium.com/@divyanshu132/lstm-and-its-equations-5ee9246d04af. [Accessed: 18-May-2020].

[23] R. Jozefowicz, W. Zaremba, I. Sutskever, "An empirical exploration of recurrent network architectures," *Publication: ICML'15: Proceedings of the 32nd International Conference on International Conference on Machine Learning*, vol. 37, pp. 2342–2350, 2015.

[24] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio., "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," *arXiv Prepr. arXiv1412.3555*, 2014.

[25] M. Phi., "Illustrated Guide to LSTM's and GRU's: A step by step explanation." [Online]. Available: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21. [Accessed: 19-May-2020].

[26] 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," *Futur. Gener. Comput. Syst.*, vol. 100, pp. 779–796, 2019. https://doi.org/10.1016/j.future.2019.05.041.

[27] O. Ibitoye, O. Shafiq, and A. Matrawy., "Analyzing Adversarial Attacks Against Deep Learning for Intrusion Detection in IoT Networks," 2019.

[28] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke., "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *J. Inf. Secur. Appl.*, vol. 50, p. 102419, 2020. https://doi.org/10.1016/j.jisa.2019.102419.

[29] M. Ge, X. Fu, N. Syed, Z. Baig, G. Teo, and A. Robles-Kelly., "Deep learning-based intrusion detection for IoT networks," in *Proceedings of IEEE Pacific Rim International Symposium on Dependable Computing, PRDC*, pp. 256–265, 2019. DOI: 10.1109/PRDC47002.2019.00056.

[30] O. AlKadi, N. Moustafa, B. Turnbull, and K.-K. R. Choo., "Mixture Localization-Based Outliers Models for securing Data Migration in Cloud Centers," *IEEE Access*, vol. 7, pp. 114607–114618, 2019. DOI: 10.1109/ACCESS.2019.2935142.

[31] M. Erza Aminanto, K. Kim., "Deep Learning in Intrusion Detection System: An Overview," 2016.

[32] "The BoT-IoT Dataset." [Online]. Available: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/bot_iot.php. [Accessed: 22-Feb-2020].