An Analysis of Deep learning for Botnet Detection (July 2019)

Yongjian Zeng(20773103), University of Waterloo

***Abstract*—Botnet is defined as a number of computers or devices managed on the Internet to perform unintended malicious activities without the authorization of the system owner. It has become a growing global concern for system users. organizations, and even for countries. As one of the most powerful tools, neural network also be used to detected botnet. In this paper, we present a survey of botnet detection using convolutional neural network and recurrent neural network. These methods needed no hand-designed features but directly took raw traffic as input data. In order to build botnet detector that can work well in reality case, we choose the botnet dataset of University New Brunswick in our research. This dataset combined several botnet dataset with normal network traffic, and test set contain more kinds of botnet than training set. The experimental results illustrate that although the accuracy is lower than 90%, deep learning is an efficient method for identifying botnet traffic.**

***Index Terms*—Botnet detection; convolutional neural network; representation learning; recurrent neural network; deep learning**

I. Introduction

THE digital age has brought many benefits to society, with the Internet acting as an enabler for growth and development across practically all sectors of business and industry. Unsurprisingly, it has also become an attractive and fertile environment for criminal activity. Malware programs, which may take many forms, are now frequently used for financial theft, identity theft, espionage, and disruption of services. A particularly troublesome type of malware is a bot, an exploited system which acts as a remote tool for an attacker to control and use the resources of a target system. Typically, the attacker (called the botmaster) will do the same for multiple systems and then use them collectively, in what is known as a botnet, to launch attack.

Botnets can be deployed in a centralized, decentralized, or hybrid fashion depending on their C&C architecture. A botnet can be classified according to its communication protocol such as IRC, HTTP, P2P, and IM [1]. In the past, the IRC-based centralized C&C structure was very popular and commonly used by botnet creators. Currently, the HTTP(S)-based centralized or decentralized approach is executed. Because this protocol is the most prominent and used by Internet.

Conventionally, there are four kinds of methods to detect botnets [2]: port-based, deep packets inspection (DPI)-based, statistical-based, and behavioral-based. Port-based and DPI-based methods are rule-based approaches, which perform traffic classification by matching predefined hard-coded rules. Statistical-based and behavioral-based methods are classic machine learning approaches, which classify traffic by extracting patterns from empirical data using a set of selective features. The behavioral modeling of botnet activities in terms of network connections have been useful for discovering even evasive botnet variants [1]. Essentially, botnet's behavior is modeled by discovering the frequent and common network activities as patterns, which can also be used for detection purposes. For example, all bots must connect periodically to the command and control (C&C) server to receive orders or to update their status. These communication traces can be represented as the underlying patterns of communication activities to discover botnet traffic. However, due to the constantly evolving behavior of botnets, it is more difficult for these conventional approaches to detect botnets today [3].

Representation learning is a new rapidly developing machine learning approach in recent years that automatically learning features from raw data. Deep learning method is a typical approach of representation learning has achieved very good performance in many domains including image classification and speech recognition [4] [5]. The main goal of this paper is to attempt to apply representation learning approach to botnet traffic classification domain and demonstrate its effectiveness.

The remainder of this paper is organized as follows. Section II describes literatures that I review. Section III introduce principle about CNN and RNN. Section IV describes the methodology of the proposed method using CNN and RNN. Section V presents experiment results and analysis. Section V describes limitation of our current method

II. Literatures Review

Because deep learning method can extract features from input data automatically and find complicate pattern behind botnet, it becomes popular tool in network security. Torres proposed an RNN botnet detector with LSTM networks for building detection models based on the behavior of connections [6]. Data are separated into different connections and each connection has its own string of symbols that represents its behavior. The Attack Detection Rate (ADR) and False Alarm Rate (FAR) are

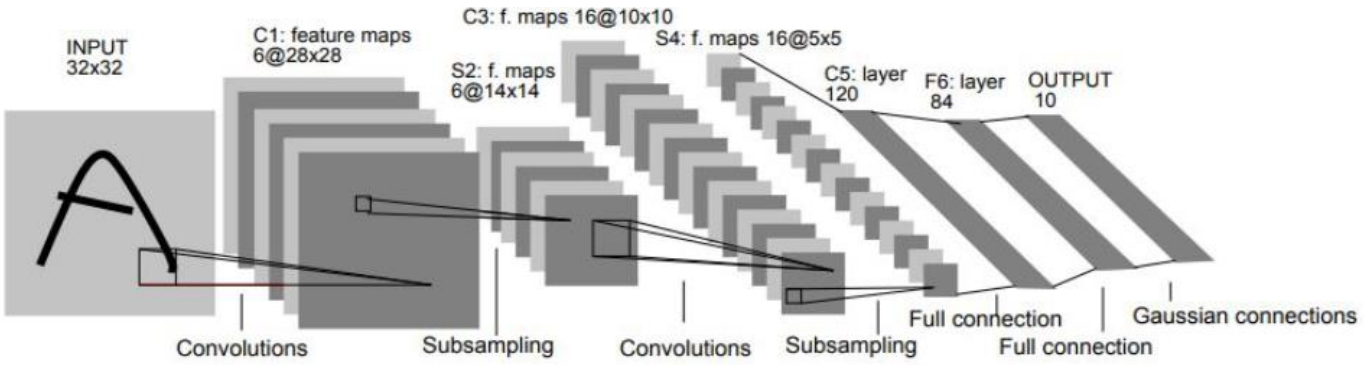

Fig. 1. Structure of LeNet-5

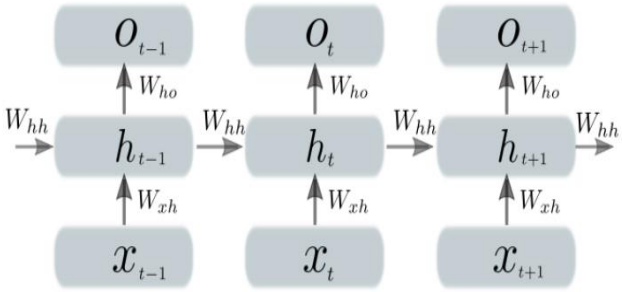

Fig. 2. Typical Structure of RNN

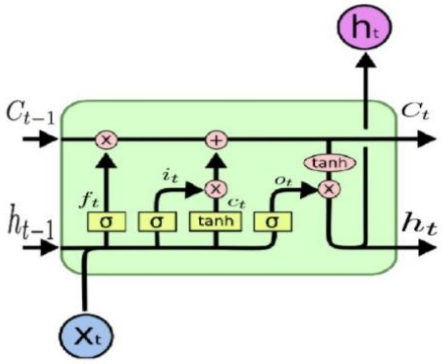

Fig. 3. Structure of modules in LSTM

about 0.98 and 0.037 respectively. Wang developed a CNN architecture similar to LeNet-5 and trained it with raw data of internet packets [7]. Raw data is treated as image data to find pattern behind. They compare the performance of model trained by data in different level and unit. The CNN networks can detect botnet with accuracy greater than 99%. Wu [8] developed a botnet detector BotCatcher which combine RNN and CNN in order to makes use of space features and time features of Internet packet. BotCatcher has high accuracy, but the model needs long time to train.

Besides the structure and categories of model, dataset also have great influence on the performance of detector. Beigi built a UNB botnet dataset by merging botnet data from three different dataset using overlay methodology [9]. There are 7 types of botnet in training set and 16 types in test set. In Beigi's experiment, he used UNB dataset to train traffic classification models proposed by Zhao which similarly employs a C4.5 decision tree algorithm with feature vectors extracted from the flows using a sliding window technique. And found the detection rate decrease dramatically from 99% to 68%. The paper illustrates that we need dataset with higher authenticity to train model that can work well in reality.

III. DEEP LEARNING

Deep learning refers to an ANN and is a subfield of machine learning. The deep term becomes very popular in artificial intelligent and refers to the number of hidden layers in the neural network. In this section, I will discuss two most popular deep learning structure: CNN and RNN, which we will use to build botnet detector.

A. CNN

Convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery [10]. It typically composed of input layer convolution layer, pooling layer, fully connected layer and output layer.

In convolution layer, weights are shared, and neurons are partially connected. So, the number of parameters in this layer decrease. the It equivalent to a feature extractor which get features as well as position information. In pooling layer, the size of image data decreases again, and high-level features are extracted. Because of convolution layers and pooling layers, it takes less time and space to train CNN model. And these layers can automatically extract features so that no feature engineering are needed.

Figure 1 shows the structure of LeNet-5, a famous CNN that can do handwritten and machine-printed character recognition.

B. RNN

A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. Unlike feedforward neural networks, RNNs can use their internal state (memory) to process

sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition.

Figure 2 show the typical structure of RNN. It can be seen that the result of the front neurons affects the output of the posterior neurons. The calculation formula is as (1):

$$h_t = \sigma_t(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \quad (1)$$

$$o_t = \sigma_o(W_{ho}h_t + b_o)$$

Long short-term memory (LSTM) [11] is a special kind of RNN. It also has this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.

Figure 3 shows the details of modules in LSTM. In every modules of LSTM, the first step is to decide which information should be forgotten. This decision is made by a sigmoid layer called the "forget gate layer." It looks at h_{t-1} and x_t, and outputs a number between 0 and 1 for each number in the cell state C_{t-1}. 1 represents "completely keep this" while 0 represents "completely get rid of this.". The next step is to decide what new information to store in the cell state. This has two parts. First, a sigmoid layer called the "input gate layer" decides which values will update. Next, a tanh layer creates a vector of new candidate values, $\tilde{C}_t$, that could be added to the state. In the next step, it combines these two to create an update to the state. Finally, it decides what should be output. This output will be based on the cell state but will be a filtered version. First, the module runs a sigmoid layer which decides what parts of the cell state is output. Then, put the cell state through tanh (to push the values to be between −1 and 1) and multiply it by the output of the sigmoid gate, so that it only output the parts it decided to. The formulas are as follow:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (2)$$

$$i_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$o_t = \sigma \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior. Because its property, most researcher chose LSTM to build their RNN botnet detector. In my research, LSTM is used to build one of the botnet detectors too.

TABLE I
BOTNET DISTRIBUTION IN TRAINING SET

Botname	Type	Number of session
Neris	IRC	14443
Rbot	IRC	31526
Virut	HTTP	864
Weasel	P2P	4607
IRC attack	IRC	77143

TABLE II
BOTNET DISTRIBUTION IN TEST SET

Botname	Type	Number of session
Neris	IRC	19481
Rbot	IRC	165
Menti	IRC	4539
Sogou	HTTP	46
Murlo	IRC	8669
Virut	HTTP	33103
Zeus	P2P	283
Tbot	IRC	485
Zero Access	P2P	1234
Weasel	P2P	40000
Smoke Bot	P2P	66
IRC attack	IRC	10361
IRC bot black hole1	P2P	38
Black hole 2	P2P	19
Black hole 3	P2P	122
Osx trojan	P2P	15

IV. METHODOLOGY

A. *Dataset*

As we discussed before. The dataset used to train model will influence the performance of detector greatly. Because classic machine learning approaches focus on feature selection techniques, many current public traffic datasets are flow features datasets other than raw traffic datasets, e.g. famous KDD CUP1999 and NSL-KDD provide pre-defined 41 features in their dataset [12]. These datasets don't meet our requirement to raw traffic. Besides the form of data of dataset, the distribution also matters. In reality, botnet detections need to detect botnet have that never seen before. So if botnet data distributed averagely in training set and test set, firstly the experiment outcome can't reflect the ability of the detection because the model has seem patterns of all botnets from training set, it will not be a tough mission for any detector to find out same botnet in test set. Secondly, once it encounters new kind of botnet, the detector will not label it as botnet and has bad performance.

To solve those problems, we will use UNB botnet dataset. UNB botnet dataset is built by Beigi from University of New Brunswick. He merges botnet data from ISOT dataset, ISCX 2012 IDS dataset, botnet traffic generated by the Malware Capture Facility Project using overlay methodology. The resulting set was divided into training and test datasets that included 7 and 16 types of botnets, respectively. Tables 1 and 2 detail distribution and type of botnets in each dataset. All data are divided into different connections. The training dataset is 4.9 GB in size and the test dataset is 2.0 GB.

B. *Data Preprocessing*

Both training set and test set of UNB dataset is .pcap file. It means that all data is packet data that capture by capture tool such as Wireshark. In order to train CNN and RNN model, we need to preprocess data.

1) Remove Bad TCP data

In.pcap file of dataset, some data is tagged as bad TCP. Data become bad TCP data if it is retransmitted data, dup ACK data, out of order data and so on. Although Bad TCP data will give some information about the traffic, but it will also add duplicate or wrong information if we take them into consideration. So, we firstly remove bad TCP data.

2) Traffic granularity

The raw data should be separate into different unit so that we can analyze them. Network traffic split granularity include TCP connection, flow, session, service, and host [13]. Different split granularity results in different traffic units. In my research, all data is separate according to session. A session is defined as bidirectional flows, including both directions of traffic. The formal description is described as follows:

Raw traffic: All packets are defined as a set P where $P = \{p^1, p^2, p^3 \ldots . p^{|P|}\}$. Every packet p is defined as $p^i = \{x^i, b^i, t^i\}$, where x^i is a 5-tuple consist of (Source IP address, Destination IP address, Source port, Destination port, Transport layer protocol). b^i represent the size of packet and t^i is the start time of transmission.

Session: All packets can be divided into different session. In each subset, packets are arranged in time order, i.e. $\{p^1 = \{x^1, b^1, t^1\}, p^2 = \{x^2, b^2, t^2\} \ldots . . p^i = \{x^i, b^i, t^i\}\}$ and $t^1 < t^2 \ldots \ldots < t^n$. Every subset is defined as $S = \{x, b, d, t\}$. *x* is 3-tuple: (Source-Destination IP address pair, Source-Destination port number pair, Transport layer protocol). It means that all packets in same session have same *x*. The second element is the sum of size of packets in the session. The third is the session duration $d = t_i - t_1$. The last element is the start time of transmission. The whole dataset can be defined as $\{S^1, S^2, S^3, \ldots . . S^i\}$.

In order to divided data to sessions more accurately and efficiently, we use PcapSplitter [14] to split the dataset.

3) Transform to CNN input form

In order to trained CNN model, we need to transform each session to the form of CNN input. Usually, CNN is used to deal with image data. So, for every session, we take the first 1024 bytes data. If the session has less than 1024 bytes data, we fill 0 to the end. Then every session can be represented by a 32×32 matrix. Because each byte of data is between 0-255, every session is transformed to a grey image and each byte of original file represents a pixel. CNN can extract features automatic from the session same as extracting features from images. Finally, all data are normalized to [0, 1] from [0, 255].

4) Transform to RNN input form

In most case, RNN is used to deal with sequence data. In every session, sender and receiver send packet to each other in a time sequence. We can view them as sequence data. So, for every session, we pick the first 8 packets, and take the first 100 bytes data of every packet out. If session data we extract is less than 800 bytes, fill 0 to the end. Finally, every session is transformed to a 1×800 vector. And all are normalized to [0, 1] from [0, 255].

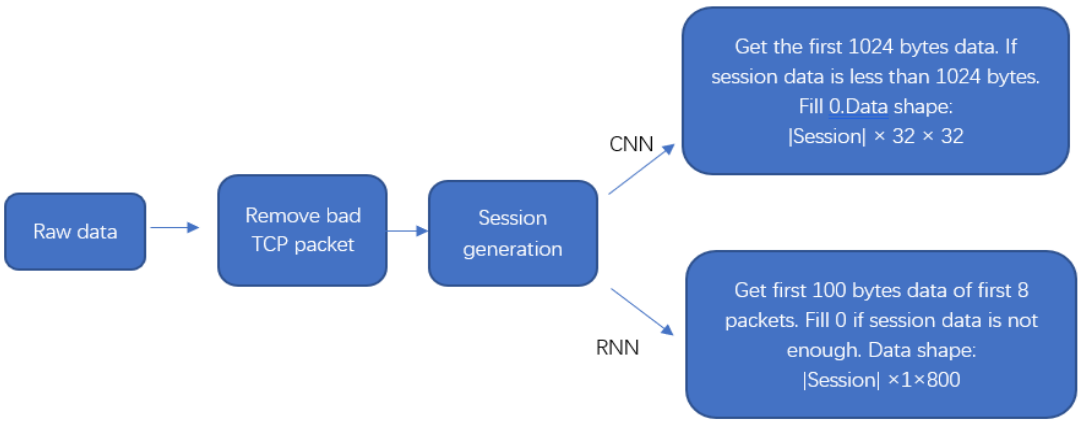

Fig. 4. Process or data preprocessing

C. Deep learning botnet detector architecture

In this section, I will discuss architectures of three deep learning botnet detector: CNN -based botnet detector, RNN-based botnet detector and hybrid botnet detector.

1) CNN-based botnet detector

The size of preprocessed dataset and the size of each image are both very similar to classic MNIST dataset [15], so a CNN architecture similar to LeNet-5 [4] should worked. In Wang's paper [7], he designed a CNN architecture which is similar to LeNet-5 used but with much more channels in each convolution layer. The input size of Wang's model is 28×28 and he randomizes the IP addresses and MAC addresses of every packet.

Learning from Wang's model, I use the same architecture for my CNN-based botnet detector, but increase the input size from 28×28 to 32×32 to let more information to be considered in each session. And I don't randomize IP addresses and MAC addresses so that these addresses can help the model to learn blacklist. The CNN-based botnet detection is as follow:

1) Input layer: CNN reads traffic image of size of 32×32×1 from preprocessing dataset.

2) Convolution layer C1: Convolution layer C1 performs a convolution operation with 32 kernels of size of 5×5. The results of C1 layer are 32 feature maps of size of 32×32.

3) Maxpooling layer M1: 2×2 maxpooling operation in M1 layer and the results are 32 feature maps of size of 16×16.

4) Convolution layer C2: the second convolution layer C2 has 64 kernels of size 5×5. The results are 64 feature maps of size of 16×16.

5) Maxpooling layer M2: 2×2 maxpooling operation in M2. The results of M2 layers are 64 feature maps of size of 8×8.

6) Dense layer D1: Fully connected layer D1 with 1024 neurons. Before coming into D1, data need to be flattened and dropout is used.

7) Dense layer D2: Fully connected layer D2 with 10 neurons.

8) Output layer O1: Output layers with two neurons using softmax function to output the probability of each class

2) RNN-based botnet detector

CNN can only learn spatial feature, and it cannot get the dependency relation among sequence data. However, RNN model can learn features in time dimension and find the relationship between previously input data and data input later. As introduced before, the RNN-based botnet detector is based on LSTM, a powerful RNN which can tackle problem about long-term dependency.

The input for RNN-based botnet detector is 1×800 vectors, For every packet, data in IP header, TCP header, and UDP header are used to represent different fields of size of sixteen or

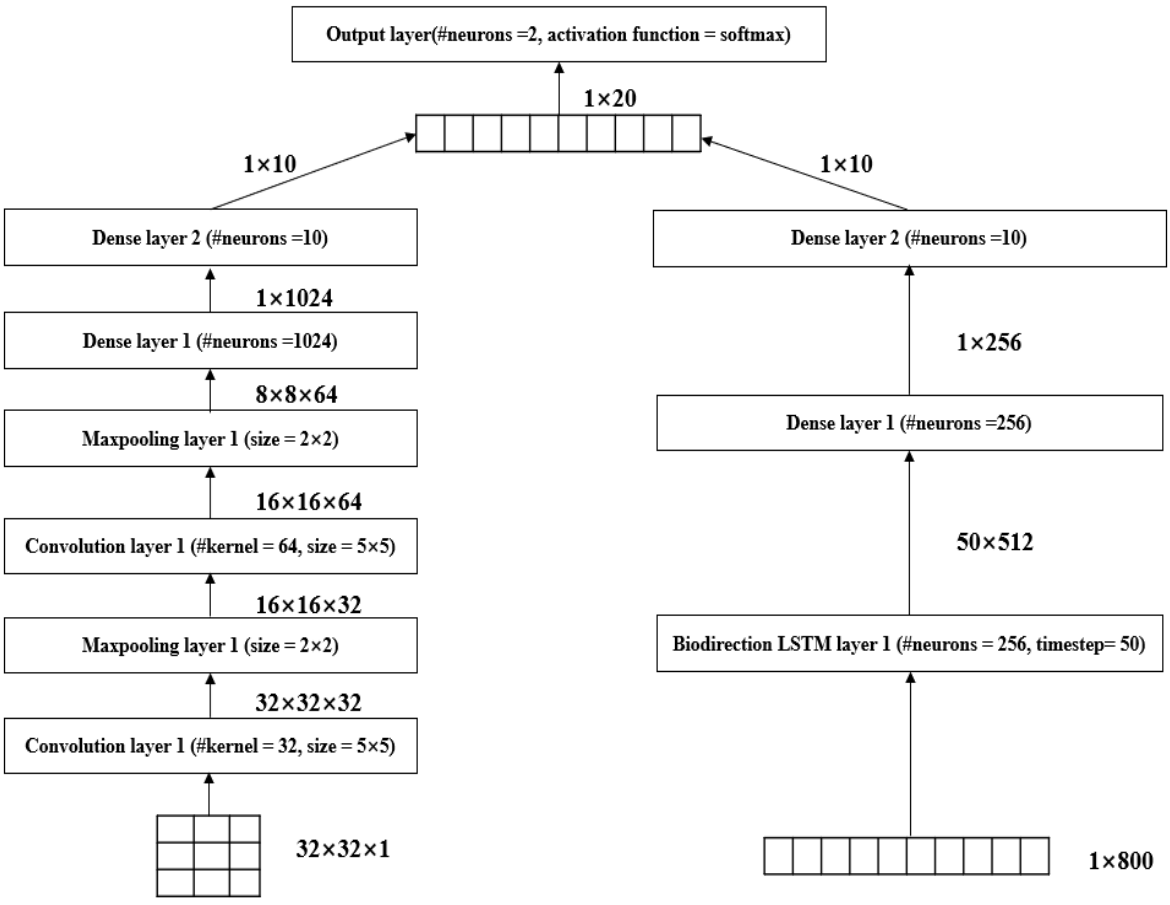

Fig. 5. Architecture of hybrid botnet detector

eight bytes. So, we set the number of timestep to 50 such that in every timestep, 16 bytes data are input to LSTM layer. Then the network can get more complete information. The detail of the RNN model is as follow:

1) Input layer: The input layer reads traffic data of size of 1 ×800 from preprocessing dataset.

2) LSTM layer L1: Layer L1 is bidirectional LSTM layer composed of 16 neurons. The timestep is set to 50.

3) Dense layer D1: Fully connected layer D1 with 128 neurons. Before coming into D1, data need to be flattened and dropout is used.

4) Output layer O1: Output layers with two neurons using softmax function to output the probability of each class

3) Hybrid botnet detector

As CNN can learn spatial features and RNN can learn time features from data. I proposed a hybrid botnet detector that combine RNN and CNN together in order to gain the advantages of them.

The hybrid botnet detector uses similar CNN and RNN architecture I introduced before but combine the results at the output layer. The details of architecture are shown in figure 5.

V. EXPERIMENT

A. Evaluation Criteria

To evaluate the performance of models, three metrics are used: Accuracy, Attack Detection Rate (ADR) and False Alarm Rate (FAR). Accuracy is used to evaluate the overall performance of the classifier. ADR is computed as the ratio between the number of correctly detected attacks and the total number of attacks. Whereas FAR rate is computed as the ratio between the number of normal connections that are incorrectly classified as attacks and the total number of normal connections. The formulas are as follow:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (3)$$

$$ADR = \frac{TP}{TP + FN}$$

$$FAR = \frac{FP}{TN + FP}$$

B. Experiment Design

Four models are built using Keas library in the experiment: CNN-based botnet detector, RNN-base botnet detector, Hybrid botnet detector as well as the models proposed by Wang. These models are trained and test with UNB botnet dataset, and the results are compare based on the evaluation criteria.

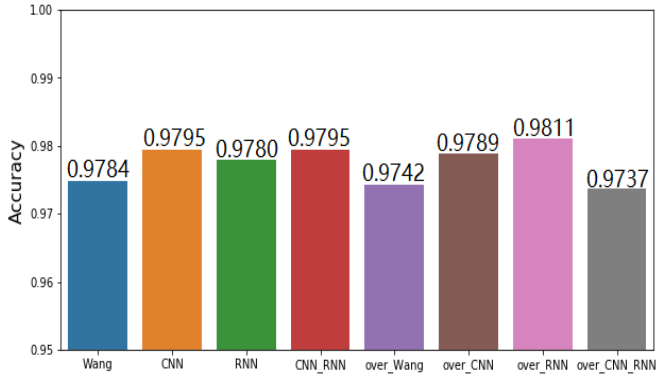

Fig. 6. Accuracy 8 botnet detectors

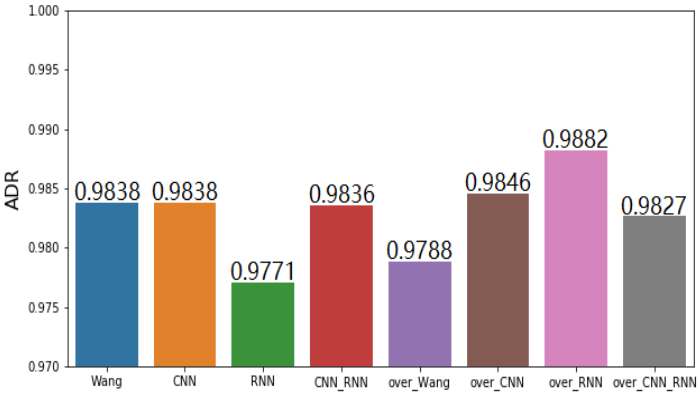

Fig. 7. ADR of 8 botnet detectors

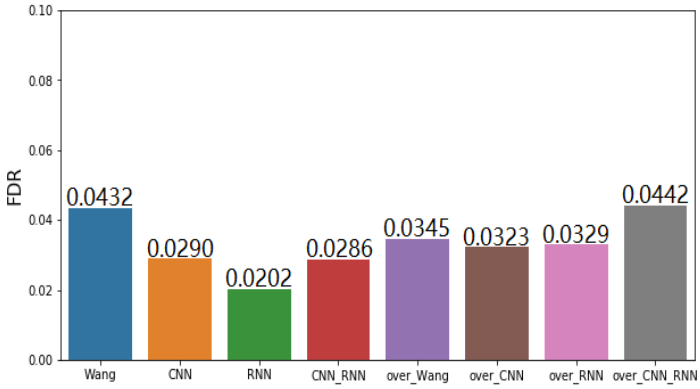

Fig. 8. FDR of 8 botnet detectors

Besides, because the botnet data of the training set is more than the data of the normal network, we implement oversampling which duplicate some normal traffic data to keep the training set balanced. Another four models will be trained by the new training set.

The experiment is carried out on personal laptop Honor Magicbook 2019. CPU is AMD Ryzen5 3500U@2.1GHz CPU. The memory is 8G and no GPU acceleration.

C. Result and Discussion

1) Overall performance of models

Figure 6-8 shows the detail of Accuracy, Attack Detection Rate and False Alarm Rate (FAR) of eights models.

TABLE III
ACCURACY OF MODELS FOR DIFFERENT BOTNETS (%)

Botname	Wang	CNN	RNN	Hybrid	Over_Wang	Over_CNN	Over_RNN	Over_Hybrid
Black hole 2	21.05	100.00	100.00	100.00	15.78	89.47	84.73	94.73
Black hole 3	18.85	11.47	49.18	57.37	18.85	27.04	54.91	13.11
IRC	95.35	97.29	92.69	97.25	94.25	96.28	98.16	97.92
Black hole 1	47.36	42.10	36.84	26.31	42.10	39.47	44.73	39.47
Menti	99.95	99.95	99.91	99.95	99.55	99.71	99.95	99.73
Murlo	97.16	97.18	97.04	97.21	97.09	97.40	97.46	97.23
Neris	99.83	99.06	98.57	99.79	95.55	99.89	99.57	99.97
Osx trojan	80.00	93.33	93.33	20.00	80.00	86.66	86.66	73.33
Rbot	98.18	98.78	95.75	98.78	98.78	99.39	96.96	98.78
Smoke Bot	54.54	83.33	15.15	59.09	51.51	69.69	65.15	69.18
Sogou	91.30	97.82	80.43	100.00	93.47	95.65	93.47	97.82
Tbot	97.11	91.34	95.25	95.67	76.08	97.31	99.38	83.09
Virut	99.81	99.90	99.81	99.92	99.83	99.94	99.96	99.82
Weasel	99.77	99.99	99.94	99.93	99.82	99.96	99.94	99.79
Zero_access	39.46	24.55	16.77	14.99	12.56	22.69	40.43	8.50
Zeus	77.38	79.50	77.03	75.97	75.26	83.74	92.22	76.32

TABLE IV
ACCURACY OF MODELS FOR BOTNETS OF DIFFERENT PROTOCALS (%)

Type	Wang	CNN	RNN	Hybrid	Over_Wang	Over_CNN	Over_RNN	Over_Hybrid
HTTP	99.80	99.90	99.78	99.92	99.82	99.93	99.96	99.82
IRC	98.21	98.27	96.96	98.65	97.54	98.49	98.84	98.72
P2P	97.44	97.28	96.85	96.82	96.67	97.24	97.89	96.56

Regarding the ADR and FDR, the value of ADR and FDR indicate the bad performance of hybrid model, which illustrate that simply combine RNN and CNN together cannot obtain their advantages. A more robust architecture needs to be researched. Compared to models trained by original dataset, CNN-based model and RNN-based model which are trained by the oversampling dataset have higher ADR. And from the figure of FDR, we find that oversampling dataset increases the FDR of models too. That because the duplicated normal traffic data keep offering same information to model which make the model firmly believe these are the pattern of normal network traffic. That is, the model is overfitting. So, it increases the possibility that model classify data as botnet traffic once the data does not satisfy the pattern. So, the detection rate will increase, and the false detection rate will increase too. In total, models proposed in this paper have higher ADR and lower FDR than Wang's model, which indicate more powerful botnet detection ability.

For accuracy, it shows the overall performance of models. We can see that except for the hybrid model trained by the oversampling dataset, the other models have higher accuracy than Wang's model. The CNN-based model has similar architecture with Wang's model, but it increases the size of input data to let more information to be read by model, and IP addresses and MAC addresses in packet can help the model learn blacklists. The result proves that these two modifications improve the performance of detector slightly. RNN-based model trained by the oversampling dataset has the highest accuracy among these models. It tells that dependency relation between previous packets and packets arrive after shows properties of botnet, and model can use these properties to detect botnet traffic. However, accuracy of the hybrid model is low, so it needs to be explored further.

2) Detailed performance of models

Table III and Table IV show information about the detailed accuracy of models. It can be seen that botnets appear in training set, such as Neris, Virut and Weasel, can be detected by all models with high accuracy. For some other botnets, such as Menti, Sogou and Murlo, models also have high detection accuracy on them. That may because these botnets have similar pattern with botnets in training set. The change of accuracy of all models has same trend. It also reflexes the complexity of botnet activities.

From the table we can find that most of botnets with low detection accuracy are P2P-based botnet. It illustrates that P2P-based botnet have more complicate spread and infection pattern than IRC-based and HTTP-based botnet. Different botnet may have big difference to each other although they both spread based on P2P protocol. Once there are not enough data or the botnet spread with new pattern, it will be a tough work to detect them. Numbers in Table IV also shows that all models have lowest accuracy when detect P2P-based botnet.

VI. CONCLUSION

Nowadays, Botnets can be deployed in a centralized, decentralized, or hybrid fashion depending on their C&C architecture. It is difficult to detect botnets with traditional traffic classification methods which need lots of professional knowledges and complex feature extraction process. On the basis of deep learning technique, three different botnet detection model are proposed. The models need not to hand-design traffic. The raw traffic data is directly the input data of traffic classifier, and the classifier can learn features automatically. In order to train botnet detector that can work well in reality, we use UNB botnet to train our models, which have much more kind of

botnet in test set than in training set. To research whether the imbalance of training set will influence the results of training, we build a oversampling dataset by duplicate normal traffic data in training set. Then we train totally eight models and compare their performance. The results that our models can detect botnet with relatively high accuracy.

For future work, there are three aspects that can be improved about the research. Firstly, the hybrid botnet detect does not perform as well as we expect. We need further exploration on what kind of architecture can combine the advantages of CNN and RNN to build a more powerful detector. Secondly, although the UNB dataset is more authentic than many other botnet dataset, the distribution of botnet traffic and normal traffic is not real enough since there are much more normal traffic data than botnet traffic data in reality. So, another dataset is needed to better reflect the capabilities of out models. Thirdly, when classify botnet traffic, separate network traffic into different set first according to the application-layer protocol, and then train models base on these different set should build models with higher accuracy. These problems will be discussed in further research.

REFERENCES

[1] Pekta¸s A, Acarman T. Botnet detection based on network flow summary and deep learning. Int J Network Mgmt. 2018;28:e2039.
[2] E. Biersack, C. Callegari and M. Matijasevic, Data traffic monitoring and analysis. Berlin: Springer, 2013.
[3] Catania CA, Garino CG. Automatic network intrusion detection: current techniques and open issues. Comput Electr Eng.2012;38(5):1062-1072
[4] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning", Nature, vol. 521, pp. 436-444, May 2015.
[5] Z. Qingqing, L. Yong, W. Zhichao, P. Jielin and Y. Yonghong, "The Application of Convolutional Neural Network in Speech Recognition",
[6] P. Torres, C. Catania, S. Garcia and C. G. Garino, "An analysis of Recurrent Neural Networks for Botnet detection behavior," 2016 IEEE Biennial Congress of Argentina (ARGENCON), Buenos Aires, Argentina, 2016, pp. 1-6.
[7] Wei Wang, Ming Zhu, Xuewen Zeng, Xiaozhou Ye and Yiqiang Sheng, "Malware traffic classification using convolutional neural network for representation learning," 2017 International Conference on Information Networking (ICOIN), Da Nang, 2017, pp. 712-717.
[8] Wu, D & Fang, B & Cui, X & Liu, Q. (2018). BotCatcher: botnet detection system based on deep learning. Tongxin Xuebao/Journal on Communications. 39. 18-28. 10.11959.
[9] Beigi, Elaheh Biglar, et al. "Towards effective feature selection in machine learning-based botnet detection approaches." Communications and Network Security (CNS), 2014 IEEE Conference on. IEEE, 2014.
[10] Krizhevsky, Alex et al. “ImageNet Classification with Deep Convolutional Neural Networks.” Commun. ACM 60 (2012): 84-90.
[11] Hochreiter, Sepp and Jürgen Schmidhuber. “Long Short-Term Memory.” Neural Computation 9 (1997): 1735-1780.
[12] M. Tavallaee, E. Bagheri, W. Lu and A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set", Proc. 2009 IEEE Int. Conf. Comput. Intell. Security Defense Appl., pp. 53-58
[13] A. D ainotti, A. Pescape and K. Claffy, "Issues and future directions in traffic classification", Network IEEE, vol. 26, no. 1, pp. 35-40, 2012.
[14] Google, PcapPlusPlus, (2019), GitHub repository, https://github.com/seladb/PcapPlusPlus
[15] IDX File Format Specifications, Behaviour and Example, http://www.fon.hum.uva.nl/praat/manual/IDX_file_format.html, 2016.