2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering

Effective Botnet Detection Through Neural Networks on Convolutional Features

Shao-Chien Chen, Yi-Ruei Chen, and Wen-Guey Tzeng
Department of Computer Science, National Chiao-Tung University, Taiwan, 30010
Email: b10032034@gmail.com, yrchen.cs98g@nctu.edu.tw, wgtzeng@cs.nctu.edu.tw

***Abstract*—Botnet is one of the major threats on the Internet for committing cybercrimes, such as DDoS attacks, stealing sensitive information, spreading spams, etc. It is a challenging issue to detect modern botnets that are continuously improving for evading detection. In this paper, we propose a machine learning based botnet detection system that is shown to be effective in identifying P2P botnets. Our approach extracts convolutional version of effective flow-based features, and trains a classification model by using a feed-forward artificial neural network. The experimental results show that the accuracy of detection using the convolutional features is better than the ones using the traditional features. It can achieve 94.7% of detection accuracy and 2.2% of false positive rate on the known P2P botnet datasets. Furthermore, our system provides an additional confidence testing for enhancing performance of botnet detection. It further classifies the network traffic of insufficient confidence in the neural network. The experiment shows that this stage can increase the detection accuracy up to 98.6% and decrease the false positive rate up to 0.5%.**

***Index Terms*—Botnet detection, machine learning, convolutional neural networks**

I. Introduction

Botnets have become one of the most serious threats on the Internet. They command a large volume of infected computers (bots) to engage illegal activities, such as DDoS attacks, spreading spams, stealing user sensitive information, and carrying distributed computing tasks for illegal purposes. An attacker (botmaster) remotely controls the bots through command and control (C&C) channels that operate on different communication protocols with various network topologies [1], [2], [3]. Current botnets are toward P2P networks for higher connection resilience [4], [5]. The botmaster can use any bot to distribute commands to the whole botnet. A P2P botnet is difficult to detect because of its elusive nature.

Botnet detection techniques can be classified into signature-based, anomaly-based, DNS-based, and machine learning-based detection [6]. Signature-based approaches identify the known characteristics of botnets from network traffic. They are suitable for real-time detection, but fail to identify new types of botnets. Anomaly-based approaches try to discover anomalous network behaviors, such as high network latency and activities on rarely used ports. But these approaches fail to identify the botnets that blend their communication into normal network traffic. DNS-based approaches identify botnets from the DNS information of bots. Similar to signature-based ones, this kind of approaches cannot identify new botnets. Machine learning-based approaches have emerged recently since they are effective in extracting unexpected botnet patterns from network traffic. These methods transform collected network traffic into flows [7] or conversation [8], extracts useful features, and utilize learning algorithms to recognize botnets.

Contributions. In this paper, we propose a machine learning-based botnet detection system that is shown to be effective in identifying P2P botnets. The proposed system extracts flow-based features from packet headers and trains a detection model using the idea of convolutional neural networks [9] (CNN), which is widely used for image recognization [10] and speech recognization [11].

In training a detection model, our system takes the convolutional version of flow-based features as the input to a fully-connected artificial neural network (ANN). Compared with the traditional features, the convolutional ones provide more detailed information about botnets. For example, Fig. 1 demonstrates the payload size in a flow that contain 15 packets of the Sality botnet and its extracted features under the traditional and our convolutional method. The traditional

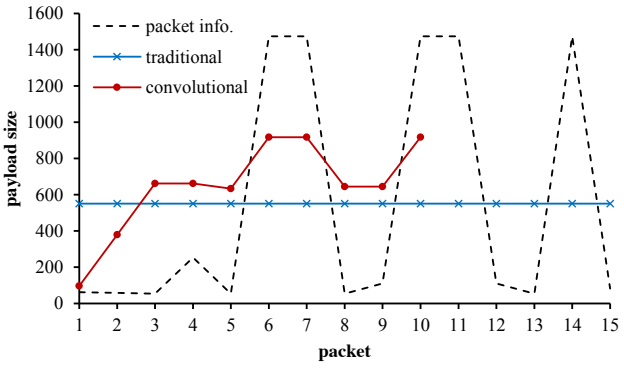

Fig. 1. An example of feature extraction (payload size) under the traditional and our convolutional method.

one extracts the payload size feature by averaging the payload size of the packets. In contrast, the convolutional one averages payload size according to a fix-sized (5-dimensional here) sliding window. It is clear that the convolutional one is closer to the original packet information than the traditional one. Our experimental result shows that using convolutional features as the input to neural networks can achieve 94.7% of detection accuracy and 2.2% of false positive rate on the known PeerRush [12] and CTU [13] P2P botnet datasets. Moreover, we compare our approach with previous approaches

2324-9013/18/31.00 ©2018 IEEE
DOI 10.1109/TrustCom/BigDataSE.2018.00062

by using their suggested feature sets. The results show that the accuracy of our detection method using the convolutional features is better than the ones using the traditional features.

Our system provides an additional confidence testing stage for enhancing accuracy in botnet detection. The confidence testing stage classifies the flows of insufficient confidence in the neural network. The experiment shows that this stage further increases the detection accuracy up to 98.6% and decreases the false positive rate up to 0.5%.

Related Works. Botnet detection has been an emerging research topic in recent years. There are a large amount of techniques in the literature [1], [2], [3]. They can be categorized as signature-based, anomaly-based, DNS-based, and machine learning-based methods [6].

Signature-based approaches identify the known characteristics of botnets in traffic. Snort [14] is a classical IDS that detects botnets according to the collected signatures and the used protocols. Bro [15] uses an even engine to reduce a kernel-filtered traffic into higher-level events, and uses a policy script interpreter to interpret event handlers to express security policies. Event handlers uses syslog to update state information, synthesize new events, record information, and generate real-time notifications. Rishi [16] monitors traffic for suspicious IRC nicknames, IRC servers, and uncommon server ports, and detect bots by using n-gram analysis and scoring systems. BotHunter [17] focuses on recognizing the infection and coordination dialog during a successful malware infection. If a sequence of evidence matches the infection dialog model, it captures all the relevant events and sources in the infection process. Wurzinger et, al. [18] build a detection model through recognizing the characteristic fact of the commands from botmaster and the corresponding activities of the bots. Botcatch [19] uses signature- and behavior-analysis modules to generate detection results, respectively. It generates the final detection result through a correlation engine and adjusts detection modules through a multi-feedback engine.

Anomaly-based approaches are based on anomalous network behaviors, such as high network latency and activities on rarely used ports. Binkley and Singh [20] combine an IRC mesh detection component with a TCP scan detector for detecting IRC-based botnet meshes. The IRC component determines the IRC mesh based on IP channel names and collects statistics on individual IRC hosts in channels. Binkley [21] uses statistical detection based on four types of IRC messages: PRIVMSG, JOIN, PING, and PONG. The experiments show that the bot server has anomalous statistics comparing with the statistics on normal IRC servers. Karasaridis et, al. [22] analyze botnets using transport layer data, and produce alerts with information about controllers after the analysis of application layer data. The system indicates less than 2% false positive rates. BotSniffer [23] identifies both C&C servers and infected hosts in a local area network without any prior knowledge of signatures or C&C server addresses. It captures the spatial-temporal correlation of network traffic within the same botnet, and utilizes statistical algorithms to detect botnets. Siboni and Cohen [24] present a detection system based on the Lempel Ziv universal compression algorithm. The algorithm assigns probability for normal traffic and estimates the likelihood of new traffic for classification. Sakib and Huang [25] detect HTTP-based C&C traffic by using statistical features based on the HTTP requests from client and the responses of DNS server. They use three unsupervised anomaly detection techniques to isolate suspicious C&C communications. It achieves more than 90% detection rate under a reasonably low false positive rate.

DNS-based approaches identify botnets based on the particular DNS information of bots. Choi et, al. [26] detect botnets by observing group activities in DNS queries sent by distributed bots simultaneously. Manasrah et, al. [27] propose a mechanism that classifies the DNS traffic from single hosts and a group of hosts with periodical and duplicate queries. It can detect botnet activities with average detection rate of 89%. Choi and Lee [28] propose BotGAD (botnet group activity detector) to detect botnets from their group activities while providing real-time monitoring in large scale networks. Phoenix [29] analyzes DNS traffic to recognize botnets based on the domain-generation algorithms (DGAs). It recognizes DGA- and non-DGA-generated domains using a combination of string and IP-based features and finds the groups of DGA-generated domains that are representative of the respective botnets. The detection accuracy is 94.8% in the experiments. Nguyen et, al. [30] propose a method to detect DGA botnets using Collaborative Filtering and Density-Based Clustering. The method uses clustering and classification algorithms to remove noise and group similar domains based on similarity in characteristic distribution of domain names. It then uses collaborative filtering to identify bots in each botnet and offline malwares of the infected machines. PsyBoG [31] employees power spectral density (PSD) analysis to recognize the major frequencies in the periodic DNS queries of botnets. It only utilizes the timing information of queries regardless of the number of queries and domains. In the experiments, PsyBoG achieves 95% of detection accuracy and detects 23 unknown and 26 known botnet groups with 0.1% false positives.

Machine learning based approaches are emerging botnet detection methods, which are useful to extract unexpected patterns from traffic. Gu et, al. [32] proposes BotMiner to cluster similar communication and malicious traffic according to group activities. BotMiner then performs cross cluster correlation to identify the hosts that share both similar communication and malicious activity patterns. Liao and Chang [33] propose a P2P botnet detection. They extract flow features from P2P bots, P2P softwares, online games, and general Internet flows, and use J48, NaiveBayes, and BayesNet methods to obtain 98%, 89%, and 87% botnet detection accuracy, respectively. Felix and Hakimian [34] focus on detecting P2P botnets during the C&C phase using traffic behaviors only. They extract 13 flow-based features and 4 host-based features and use Nearest Neighbors Classifier, Linear Support Vector Machine, Artificial Neural Network, Gaussian Based Classifier, and Naive Bayes classifiers for detection. The SVM algorithm achieves the highest accuracy 98%. Hang et, al. [35] propose a

graph-based solution, called Entelecheia, to detect P2P botnets in their waiting stage. It creates a graph of likely malicious flows as superflow, and uses two synergistic graph-mining steps to cluster and label botnet nodes. Entelecheia produces a median Fl score of 91.8% across various experiments. Beigi et, al. [36] revisit flow-based features employed in recent botnet detection approaches and evaluates their relative effectiveness. They categorize 21 features into byte-based, time-based, behavior-based, and packet-base ones, and selects the most useful IOPR, duration, APL, and BS, and from each category. Li et, al. [37] identify three important network behaviors: long active communication behavior (ActBehavior), connection failure behavior (FailBehavior), and network scanning behavior (ScanBehavior), and extract the features of these behaviors from flows in the network layer and transport layer. They use particle swarm optimization and K-means algorithms to uncover the host members of Botnet in the organizational network. Kirubavathi and Anitha [38] consider four flow-based features: small packet rate, packet ratio, initial packet length, and bot response packet. In the experiments, they use boosted decision tree, naive Bayesian classifier, and SVM algorithms.

II. The Construction

In this section, we provide an overview of the proposed botnet detection system and demonstrate the construction in detail.

As shown in Fig. 2, the proposed system Π consists of three stages: *prepocessing*, *training*, and *confidence testing*. The preprocessing stage transforms the original IP packets into a desired format for training. The training stage builds a detection model for differentiating botnet and normal traffic. The confidence testing stage further enhances the accuracy rate for detection.

A system overview and design explanation. In the preprocessing stage, Π transforms the network packets into flows based on the NetFlows format [7], labels each of them as malicious (bot) or benign (non-bot), and compresses the flows into a desired size. The compression is necessary since the size (packet number) of flows may range on different scales. It adjusts the size of each flow to fit the size of input layer of the subsequent ANN for training.

In the training stage, Π builds a classification model based on a modified CNN. A classical CNN consists of a feature extraction phase that contains alternating convolutional and pooling layers, and a recognition phase that is an artificial neural network (ANN) with fully-connected layers. The feature extraction phase contains one convolution layer only and no pooling layers. The recognition phase uses the extracted features to build a fully-connected ANN for mapping a flow into two indicator scores, one for bot and the other for normal.

In the confidence testing stage, Π enhances detection accuracy by further classifying the flows of insufficient confidence ANN with a decision tree.

A. Preprocessing Stage

In this stage, Π first transforms the original IP packets into flows based on the NetFlow format [7]. A flow is defined on five IP packet attributes: IP source address, IP destination address, source port, destination port, and layer 3 protocol type. All packets with the same source and destination IP addresses, source and destination ports, and protocol and within a 300-second time window [39] are grouped into a flow. We record the delivery time, payload size, and transmission direction of the packets in each flow for extracting needed features later.

Π then compresses flows with maximal size (packet number) n_p. For a flow F_i of $|F_i| = n_i$ packets, $n_i > n_p$, Π compresses F_i with the following two steps.

- Compute $\ell = \lfloor \frac{n_i}{n_p} \rfloor$, partition the sequence of packets $F_i = \langle p_1, p_2, \ldots, p_{n_i} \rangle$ into the sequence of segments of packets
$$P_i^{(j)} = \langle p_{j\cdot\ell+1}, p_{j\cdot\ell+2}, \ldots, p_{j\cdot\ell+\ell} \rangle, 0 \le j < n_p,$$
and drop the rest $(n_i - n_p \cdot \ell)$ packets. The result flow is $F_i' = \langle P_i^{(1)}, P_i^{(2)}, \ldots, P_i^{(n_p)} \rangle$;
- For each $P_i^{(j)}$, compute the averages of delivery time, payload size, and transmission direction of packets $p_{j\cdot\ell+1}$, $p_{j\cdot\ell+2}$, ..., and $p_{j\cdot\ell+\ell}$.

For example, to compress a flow $F = \langle p_1, p_2, \ldots, p_{50} \rangle$ into one of size $n_p = 14$, value $\ell = \lfloor \frac{50}{14} \rfloor = 3$ and the result F' contains the packets $P^{(0)} = \langle p_1, p_2, p_3 \rangle$, $P^{(1)} = \langle p_4, p_5, p_6 \rangle$, ..., $P^{(13)} = \langle p_{40}, p_{41}, p_{42} \rangle$. The rest 8 packets $p_{43}, p_{44}, \ldots, p_{50}$ are dropped. Assume that the packets of $P^{(1)}$ have the associated information as in Table I. The associated

TABLE I
AN EXAMPLE FOR FLOW COMPRESSION

$P^{(1)}$	p_1	p_2	p_3
delivery time	13	36	50
payload size	10	12	14
transmission direction	1	1	0

delivery time, payload size, and transmission direction of $P^{(1)}$ are computed as $\frac{13+36+50}{3} = 33$, $\frac{10+12+14}{3} = 12$, and $\frac{1+1+0}{3} = \frac{2}{3}$, respectively.

Remark. A pre-defined n_p is required for setting the size of convolutional features in the training stage. In particular, it affects accuracy and efficiency. If n_p is set too large, the compression has no affect on most flows since the sizes of the compressed flows range on many scales. Π would need to spend more time on training. In contrast, if n_p is set too small, the compression is too excessive since the associated information of the compressed flows is too much less than the original flows. According to the experimental results, we choose $n_p = 14$ for compression.

B. Training Stage

In this stage, Π first extracts the convolutional features from a compressed flow. This convolutional layer takes a set of features as maps. Each map is regarded as an m-dimensional window that can cover m continuous packets in a compressed flow. To extract features, Π slides (convolves) the window

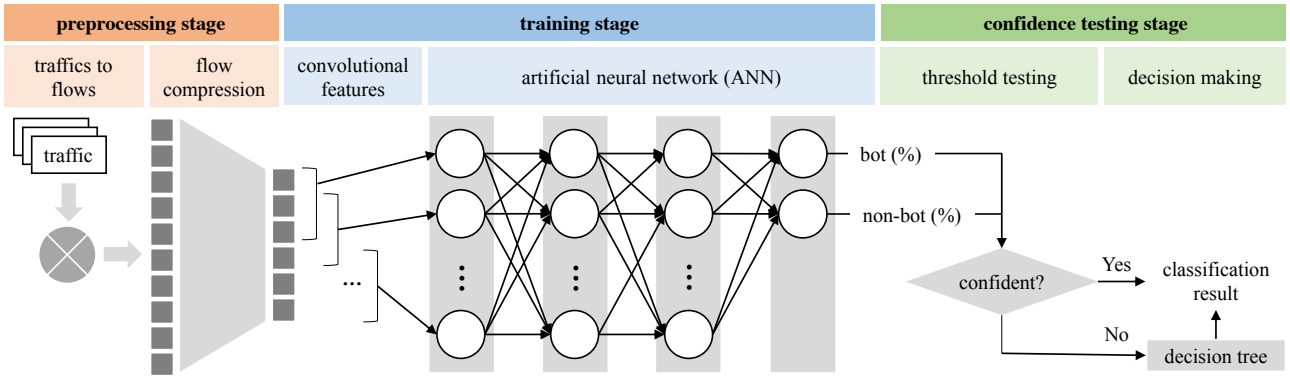

Fig. 2. An overview of the proposed botnet detection system Π.

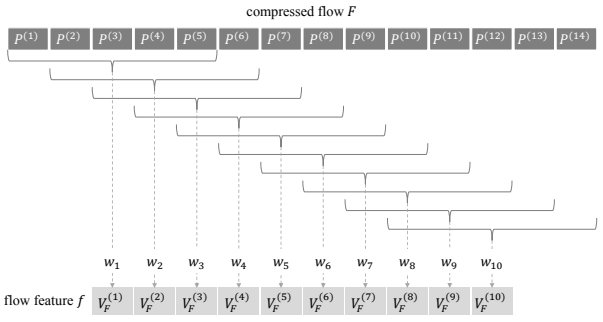

Fig. 3. Extracting convolutional features from a compressed flow F.

across a compressed flow and calculates the feature values of the covered packets accordingly.

More precisely, for each feature f and for each compressed flow $F = \langle P^{(1)}, P^{(2)}, \ldots, P^{(n)} \rangle$, Π calculates the convolutional feature values $V_F^{(i)}$ of the packets in the i-th window w_i for $1 \leq i \leq n - m + 1$, where w_i is the m-dimensional window that covers the packets $P^{(i)}$ to $P^{(i+m-1)}$. Π drops the flows with size less than m since they are too short for extracting convolutional features.

For example, Figure 3 shows the convolutional feature extraction from a compressed flow F with the compressed packets $P^{(1)}$, $P^{(2)}$, …, $P^{(14)}$. To extract a flow feature f, Π convolves a 5-dimensional window on F. Π extracts the feature information from $(P^{(1)}, P^{(2)}, \ldots, P^{(5)})$ to obtain $V_F^{(1)}$, from $(P^{(2)}, P^{(3)}, \ldots, P^{(6)})$ to obtain $V_F^{(2)}$, and so on. In our system implementation, we consider 7 useful flow features, IOPR, Duration, APL, BS, NSP, FPS, and PRR, used by Beigi et al. [36] and Kirubavathi and Anitha [38]. For instance, when we consider the IOPR feature, Π computes the ratio of the number of incoming packets to the number of outgoing packets in each of the convolutional window.

Π then builds a fully-connected ANN for reducing the convolutional feature values into two class percentages that are the possibilities of the input traffic to be bot and non-bot, respectively. Π applies max normalization [40] on the convolutional feature values for input layer and applies $Softmax$ function for the output layer.

Remark. According the experimental results, we choose $m = 5$ for the size of feature maps. For each feature f, we set the number n_f of neurons in ANN's input layer as 10, i.e., the total size of the input layer is $10 \cdot \#(features)$. According to our feature extraction procedure, the size n of a compressed flow is 14 since $n = m + (n_f - 1)$. It is the value n_p that Π needs to determine in the preprocessing stage.

C. Confidence Testing Stage

In this stage, Π classifies the flows of insufficient confidence under the classification of ANN. For a flow, ANN would output the probabilities of being bot and non-bot, respectively. The difference of these two probabilities is defined as the confidence value of correct classification. If a flow is classified with sufficient confidence in ANN, Π outputs the classification result directly. Otherwise, Π uses the decision tree method to classify the flow again. The decision tree method is a widely used classifier that is demonstrated as a good approach with high accuracy for botnet detection. Note that in the classification, Π takes the traditional features as the input using the decision tree.

III. Evaluation

In this section, we evaluate the performance of the proposed system Π. The evaluation uses three experiments. The first one is for selecting system parameters, the second one is for evaluating effectiveness of our training stage, and the third one is for evaluating effectiveness of our confidence testing stage.

Datasets. In our experiments, we use PeerRush [12] and CTU [13] datasets. PeerRush contains three P2P botnet traffic and four ordinary P2P traffic. The P2P botnet traffic are from three botnets: Storm [2], Waledac [41], and Zeus [42]. The ordinary P2P traffic are collected from running five different popular P2P applications: Skype, eMule, μTorrent, Frostwire, and Vuze. CTU contains Kelihos [43] and Sality [44] P2P botnet traffic. Therefore, we use five different P2P botnet datasets and four ordinary P2P traffic in total and take $10,000$

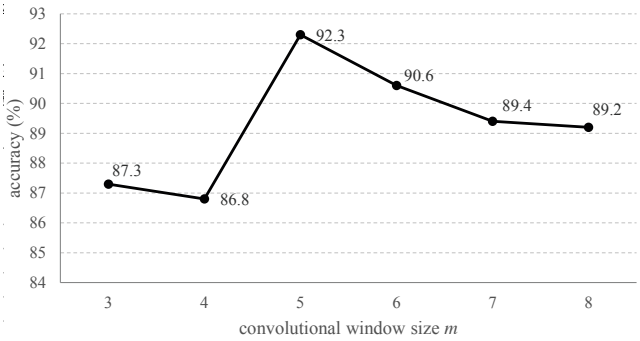

Fig. 4. Detection accuracy against different window sizes m.

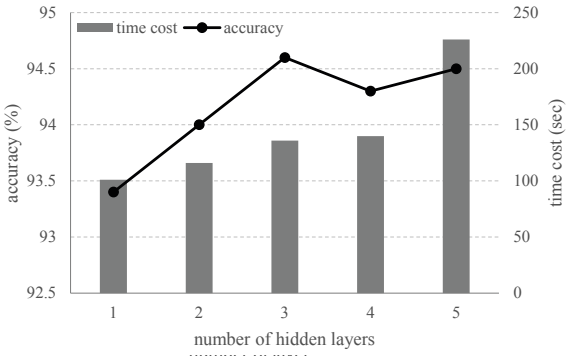

Fig. 5. Detection accuracy and training time cost against different numbers of hidden layers in ANN.

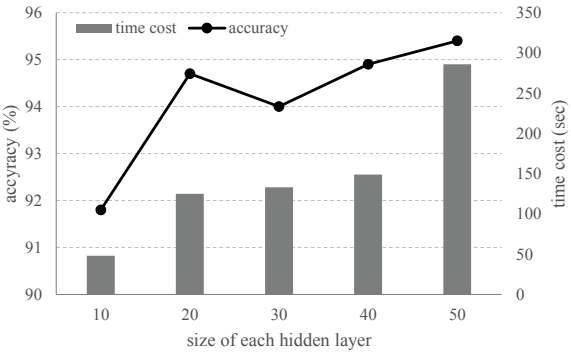

Fig. 6. Detection accuracy and training time cost against different sizes of hidden layer in ANN.

flows from each of these 9 datasets. We use the cross validation method for training.

Flow features. We extract seven features from flows. The first four features, IOPR, Duration, APL, and BS, are used by Beigi et al. [36] since they provide the highest overall detection rate. The rest three features, NSP, FPS, and PRR, are used by Kirubavathi and Anitha [38] since they are significant for botnet detection.

- *IOPR:* ratio of the number of incoming packets to the number of outgoing packets in a flow.
- *Duration:* duration of a flow.
- *APL:* average payload length per packet in a flow.
- *BS:* average bits per second in a flow.
- *NSP:* the number of small packets (10 to 320 bytes) sent and received in a flow in a specified time interval.
- *FPS:* length of the first packet in a flow.
- *RPR:* ratio of bot-response packets to total packets in a flow for a specified time interval.

Environment. All our experiments are done on an AMD Opteron Processor 6128 with 2.0 GHz and 64 GB RAM server. OS is Ubuntu 14.04.5 and the programming language is python 3.4.3. The read and write speeds of the HDD are 1.4 GB and 511 MB persecond, respectively.

A. Experiment 1 - Parameter Selection

In this experiment, we focus on selecting three parameters: the dimension m of the convolutional window, the number of layers, and the size (number of neurons) of each layer in the ANN.

To choose an appropriate m, we set ANN to have a 10-neuron input layer, 2 hidden layers, and 20 neurons in each hidden layer. The evaluation result is shown in Fig. 4. The best choice for m is 5. It achieves about 94% accuracy in botnet detection. When m gets larger, the accuracy gets lower because more packets are covered by a convolutional window, the extracted feature values are computed by averaging more packets, and thus, the convolutional features get further away from the original flow characteristic.

To determine an appropriate number of hidden layers, we set $m = 5$ and other parameters of ANN as the above. The evaluation result is shown in Fig. 5. The bar chart and line graph present the training time and the detection accuracy agaist different numbers of hidden layers. The best accuracy occurs when the number of hidden layers is 3. Once the number of hidden layers is larger than 4, the training time increases significantly. Thus, we choose the number of hidden layers to be 3.

To determine an appropriate size of each hidden layer, we set $m = 5$ and the number of hidden layers to be 3. The evaluation result is shown in Fig. 6. The bar chart and line graph present the training time and detection accuracy against different sizes of hidden layers. When the size of each hidden layer increases from 40 to 50, the training time doubles. Considering the tradeoff between detection accuracy and training time, we choose the size of each hidden layer to be 20.

In the following two experiments, we set $m = 5$ for the size of convolutional window, 3 hidden layers and 20 neurons in each hidden layer.

B. Experiment 2 - Effectiveness of the Training Stage

In this experiment, we compare detection accuracy among four feature sets (a), (b), (c), and (d) that are used in the works [36], [38], [33], and [34], respectively.

(a) IOPR, Duration, APL, BS

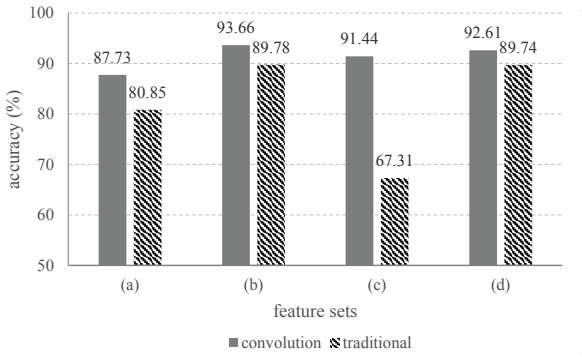

Fig. 7. A comparison of the detection accuracy on four different feature sets.

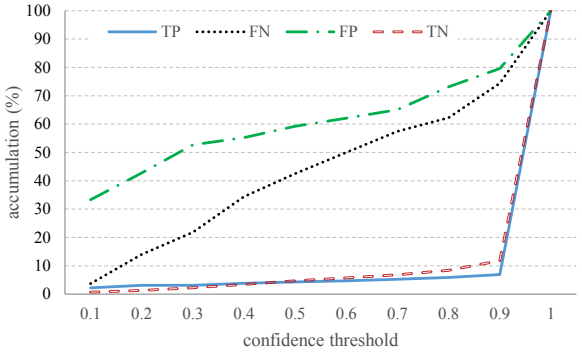

Fig. 8. Accumulation of TP, FN, FP, and TN rates against different confidence values.

(b) NSP, FPS, RPR
(c) NNP, NSP, PSP, TBT, APL
(d) PX, FPS, APL, TBT, DPL, IOPR

For a flow, NNP is the number of null packets, PSP is the percentage of small packets, TBT is the total number of bytes, PX is the total number of packets, and DPL is the total number of subsets of packets of the same length over the total number of packets.

The comparison is shown in Fig. 7. For each feature set, we compare detection accuracy of using and not using the convolutional technique. Note that in feature set (d), we do not use the "protocol" feature as in Saad et, al. [34] since it cannot be convolutionalized.

In Fig. 7, it is clear that for each feature set, using the convolutional features achieves higher accuracy than using the traditional features. It also shows that the convolutional feature sets are important for higher botnet detection rate.

C. Experiment 3 - Effectiveness of the Confidence Testing Stage

In the confidence testing stage, if the confidence value (the difference between the probabilities of being bot and non-bot by CNN) is below some threshold, Π re-classifies the flow with the decision tree method. The only thing left is to determine an approximate threshold for the cut of re-classification for a flow.

Fig. 8 shows the accumulation rates of true positive (TP), false negative (FN), false positive (FP), and true negative (TN) on $9,000$ testing flows for various confidence thresholds.

The accumulation rates of FN and FP are roughly proportional to confidence thresholds. They are not useful for choosing a suitable confidence threshold since the number of flows under incorrect classifications in ANN are almost uniformly distributed in each interval of confidence thresholds. In contrast, the accumulation rates of TP and TN show that most flows under correct classifications in ANN have confidence values higher than 0.9. That is, when the confidence threshold is set to 0.9, our system can filter out most of the flows under correct classifications. Thus, the flows under incorrect classification in ANN is classified again by using the decision tree method. Hence, we set the confidence threshold to 0.9.

TABLE II
THE PERFORMANCE OF OUR SYSTEM Π WITH CNN AND CONFIDENCE TESTING.

	TP	FN	FP	TN	Accuracy
CNN only	4730	270	201	3799	**94.7%**
CNN + CT	4924	76	46	3954	**98.6%**

Table II shows the performance of our system Π with CNN and confidence testing on $9,000$ flows. Without the confidence testing, there are $4,730$ TP, 270 FN, 201 FP, and $3,799$ TN flows. The detection accuracy is 94.7%. After we apply the confidence testing with confidence threshold 0.9, the detection accuracy increases to 98.6%. Simultaneously, the rates of TP, FN, FP, and TN are also improved.

IV. CONCLUSION

In this paper, we propose a machine learning-based botnet detection system that is effective in identifying P2P botnets. The system uses the idea of CNN for training and the decision tree method for further enhancing the detection rate. The experimental results show effectiveness of the convolutional features for botnet detection. Our system is shown to be useful in identifying botnets from well-known P2P botnet datasets with high accuracy and low false positive rates.

REFERENCES

[1] D. Acarali, M. Rajarajan, N. Komninos, and I. Herwono, "Survey of approaches and features for the identification of http-based botnet traffic," *Journal of Network and Computer Applications*, vol. 76, pp. 1–15, 2016.
[2] T. Holz, M. Steiner, F. Dahl, E. Biersack, and F. C. Freiling, "Measurements and mitigation of peer-to-peer-based botnets: A case study on storm worm," in *Proceedings of the First USENIX Workshop on Large-Scale Exploits and Emergent Threats, LEET '08, San Francisco, CA, USA, April 15, 2008*.
[3] P. Wang, S. Sparks, and C. C. Zou, "An advanced hybrid peer-to-peer botnet," *IEEE Transactions on Dependable and Secure Computing*, vol. 7, no. 2, pp. 113–127, 2010.
[4] M. Jelasity and V. Bilicki, "Towards automated detection of peer-to-peer botnets: On the limits of local approaches," in *Proceedings of the 2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats, LEET '09, Boston, MA, USA, April 21, 2009*, 2009.

[5] Q. Yan, Y. Zheng, T. Jiang, W. Lou, and Y. T. Hou, "Peerclean: Unveiling peer-to-peer botnets through dynamic group behavior analysis," in *Proceedings of the 2015 IEEE Conference on Computer Communications, INFOCOM 2015, Kowloon, Hong Kong, April 26 - May 1, 2015*, 2015, pp. 316–324.

[6] S. S. C. Silva, R. M. P. Silva, R. C. G. Pinto, and R. M. Salles, "Botnets: A survey," *Computer Networks*, vol. 57, no. 2, pp. 378–403, 2013.

[7] Cisco, "Cisco ios netflow." [Online]. Available: http://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html

[8] P. Narang, C. Hota, and V. N. Venkatakrishnan, "Peershark: Flow-clustering and conversation-generation for malicious peer-to-peer traffic identification," *EURASIP Journal on Information Security*, vol. 2014, p. 15, 2014.

[9] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[10] O. Abdel-Hamid, A. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 22, no. 10, pp. 1533–1545, 2014.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems, December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pp. 1106–1114.

[12] B. Rahbarinia, R. Perdisci, A. Lanzi, and K. Li, "Peerrush: Mining for unwanted p2p traffic," *Journal of Information Security and Applications*, vol. 19, no. 3, pp. 194–208, 2014.

[13] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Computers & Security*, vol. 45, pp. 100–123, 2014.

[14] "Snort." [Online]. Available: https://www.snort.org/

[15] V. Paxson, "Bro: A system for detecting network intruders in real-time," *Computer Networks*, vol. 31, no. 23-24, pp. 2435–2463, 1999.

[16] J. Goebel and T. Holz, "Rishi: Identify bot contaminated hosts by IRC nickname evaluation," in *Proceedings of the First Workshop on Hot Topics in Understanding Botnets, HotBots'07, Cambridge, MA, USA, April 10, 2007*.

[17] G. Gu, P. A. Porras, V. Yegneswaran, and M. W. Fong, "Bothunter: Detecting malware infection through ids-driven dialog correlation," in *Proceedings of the 16th USENIX Security Symposium, Boston, MA, USA, August 6-10, 2007*.

[18] P. Wurzinger, L. Bilge, T. Holz, J. Goebel, C. Kruegel, and E. Kirda, "Automatically generating models for botnet detection," in *Proceedings of Computer Security - ESORICS 2009, 14th European Symposium on Research in Computer Security, Saint-Malo, France, September 21-23, 2009*, pp. 232–249.

[19] Y. Ji, Q. Li, Y. He, and D. Guo, "Botcatch: Leveraging signature and behavior for bot detection," *Security and Communication Networks*, vol. 8, no. 6, pp. 952–969, 2015.

[20] J. R. Binkley and S. Singh, "An algorithm for anomaly-based botnet detection," in *Proceedings of the 2nd Workshop on Steps to Reducing Unwanted Traffic on the Internet, SRUTI'06, San Jose, CA, USA, July 7, 2006*.

[21] J. R. Binkley, "Anomaly-based botnet server detection," in *Proceedings of the FloCon Conference, Vancouver, Washington, October 9-12, 2006*, pp. 1–4.

[22] A. Karasaridis, B. Rexroad, and D. A. Hoeflin, "Wide-scale botnet detection and characterization," in *Proceedings of the First Workshop on Hot Topics in Understanding Botnets, HotBots'07, Cambridge, MA, USA, April 10, 2007*.

[23] G. Gu, J. Zhang, and W. Lee, "Botsniffer: Detecting botnet command and control channels in network traffic," in *Proceedings of the Network and Distributed System Security Symposium, NDSS 2008, San Diego, California, USA, 10th February - 13th February 2008*.

[24] S. Siboni and A. Cohen, "Botnet identification via universal anomaly detection," in *Proceedings of the IEEE International Workshop on Information Forensics and Security, WIFS 2014, Atlanta, GA, USA, December 3-5, 2014*, pp. 101–106.

[25] M. N. Sakib and C. Huang, "Using anomaly detection based techniques to detect http-based botnet c&c traffic," in *Proceedings of the IEEE International Conference on Communications, ICC 2016, Kuala Lumpur, Malaysia, May 22-27, 2016*, pp. 1–6.

[26] H. Choi, H. Lee, H. Lee, and H. Kim, "Botnet detection by monitoring group activities in DNS traffic," in *Proceedings of the Seventh International Conference on Computer and Information Technology (CIT 2007), October 16-19, 2007, University of Aizu, Fukushima, Japan*, pp. 715–720.

[27] A. M. Manasrah, A. Hasan, O. A. Abouabdalla, and S. Ramadass, "Detecting botnet activities based on abnormal DNS traffic," *CoRR*, vol. abs/0911.0487, 2009.

[28] H. Choi and H. Lee, "Identifying botnets by capturing group activities in DNS traffic," *Computer Networks*, vol. 56, no. 1, pp. 20–33, 2012.

[29] S. Schiavoni, F. Maggi, L. Cavallaro, and S. Zanero, "Phoenix: Dga-based botnet tracking and intelligence," in *Proceedings of the 11th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, DIMVA 2014, Egham, UK, July 10-11, 2014*, pp. 192–211.

[30] T. Nguyen, T. Cao, and L. Nguyen, "DGA botnet detection using collaborative filtering and density-based clustering," in *Proceedings of the Sixth International Symposium on Information and Communication Technology, Hue City, Vietnam, December 3-4, 2015*, pp. 203–209.

[31] J. Kwon, J. Lee, H. Lee, and A. Perrig, "Psybog: A scalable botnet detection method for large-scale DNS traffic," *Computer Networks*, vol. 97, pp. 48–73, 2016.

[32] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection," in *Proceedings of the 17th USENIX Security Symposium, July 28-August 1, 2008, San Jose, CA, USA*, pp. 139–154.

[33] W.-H. Liao and C.-C. Chang, "Peer to peer botnet detection using data mining scheme," in *Proceedings of the IEEE International Conference on Internet Technology and Applications, 2010.*, pp. 1–4.

[34] S. Saad, I. Traoré, A. A. Ghorbani, B. Sayed, D. Zhao, W. Lu, J. Felix, and P. Hakimian, "Detecting P2P botnets through network behavior snalysis and machine learning," in *Proceedings of the Ninth Annual Conference on Privacy, Security and Trust, PST 2011, 19-21 July, 2011, Montreal, Québec, Canada*, pp. 174–180.

[35] H. Hang, X. Wei, M. Faloutsos, and T. Eliassi-Rad, "Entelecheia: Detecting P2P botnets in their waiting stage," in *Proceedings of the IFIP Networking Conference, 2013, Brooklyn, New York, USA, 22-24 May, 2013*, pp. 1–9.

[36] E. B. B. Samani, H. H. Jazi, N. Stakhanova, and A. A. Ghorbani, "Towards effective feature selection in machine learning-based botnet detection approaches," in *Proceedings of the IEEE Conference on Communications and Network Security, CNS 2014, San Francisco, CA, USA, October 29-31, 2014*, pp. 247–255.

[37] S. Li, Y. Kao, Z. Zhang, Y. Chuang, and D. C. Yen, "A network behavior-based botnet detection mechanism using PSO and k-means," *ACM Transactions on Management Information Systems*, vol. 6, no. 1, pp. 3:1–3:30, 2015.

[38] G. K. Venkatesh and R. Anitha, "Botnet detection via mining of traffic flow characteristics," *Computers & Electrical Engineering*, vol. 50, pp. 91–101, 2016.

[39] D. Zhao, I. Traoré, A. A. Ghorbani, B. Sayed, S. Saad, and W. Lu, "Peer to peer botnet detection based on flow intervals," in *Proceedings of the Information Security and Privacy Research - 27th IFIP TC 11 Information Security and Privacy Conference, SEC 2012, Heraklion, Crete, Greece, June 4-6, 2012.*, 2012, pp. 87–102.

[40] "How to standardize data for neural networks." [Online]. Available: https://visualstudiomagazine.com/articles/2014/01/01/how-to-standardize-data-for-neural-networks.aspx

[41] C. Nunnery, G. Sinclair, and B. B. Kang, "Tumbling down the rabbit hole: Exploring the idiosyncrasies of botmaster systems in a multi-tier botnet infrastructure," in *Proceedings of the 3rd USENIX Workshop on Large-Scale Exploits and Emergent Threats, LEET '10, San Jose, CA, USA, April 27, 2010*, 2010.

[42] Lelli, "Zeusbot/spyeye p2p updated, fortifying the botnet," 2012. [Online]. Available: http://www.symantec.com/connect/blogs/zeusbotspyeye-p2p-updated-fortifying-botnet

[43] CTU-Malware-Capture-Botnet-149-1, "Kelihos," 2015. [Online]. Available: https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-66-1/

[44] CTU-Malware-Capture-Botnet-66-1, "Sality," 2014. [Online]. Available: https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-66-1/