Botnet Detection Using Recurrent Variational Autoencoder

Jeeyung Kim*, Alex Sim*, Jinoh Kim†, Kesheng Wu*
*Lawrence Berkeley National Laboratory, Berkeley, CA, USA
†Texas A&M University, Commerce, TX, USA
Email: *{jeeyungkim, asim, kwu}@lbl.gov, †jinoh.kim@tamuc.edu

***Abstract*—Botnet detection is an active research topic as botnets are a source of many malicious activities, including distributed denial-of-service (DDoS), click-fraud, spamming, and crypto-mining attacks. However, it is getting more complicated to identify botnets due to the continuous evolution of botnet software and families that harness new types of devices and attack vectors. Recent studies employing machine learning (ML) showed improved performance to detect botnets to some extent, but they are still limited and ineffective with the lack of sequential pattern analysis, which is a key to detect various classes of botnets. In this paper, we propose a novel botnet detection method, built upon Recurrent Variational Autoencoder (RVAE), that effectively captures sequential characteristics of botnet anomalies. We validate the feasibility of the proposed method with the CTU-13 dataset that have been widely employed for botnet detection studies, and show that our method is at least comparable to existing techniques in terms of detection accuracy. In addition, our experimental results show that the proposed method can detect previously unseen botnets by utilizing sequential patterns of network traffic. We will also show how our method can detect botnets in the streaming mode, which is the essential requirement to perform real-time, on-line detection.**

***Index Terms*—botnet detection, anomaly scoring, Recurrent Neural Network, Variational Autoencoder, network security**

I. Introduction

Botnets are one of the critical threats to the cyber-security as they are considered a source of many malicious activities [1]. The compromised machines (or "bots") in the botnet are driven by attackers to perform specific attacks, such as distributed denial-of-service (DDoS), click-fraud, spamming, and crypto-mining attacks, to list a few. Botnets could also harbor malware and ransomware that may be delivered to victims as a consequence of attacks. Given the increasing number of botnet-based attacks and their fatal severity, it is essential to detect botnets effectively to minimize financial and societal losses caused by such attacks.

With the growing security concern raised by botnets, there has been a significant body of studies for detecting botnets [2]–[8]. However, it is getting more challenging to identify botnets due to some reasons. First, the malicious software that infects a victim host and that operates the botnet is evolving in a way to evade existing detection tools. For example, botnets may use a combination of protocols, such as Internet Relay Chat (IRC), peer-to-peer (P2P), and HTTP, rather than relying on a static protocol [6]. Second, newly introduced botnets are more complicated utilizing diverse types of computing devices and attack vectors. In 2016, the Mirai botnet started controlling hundreds of thousands of Internet of Things (IoT) devices that are exploited to conduct a high-profile DDoS attack [9]. Another sophisticated botnet system called Smominru is known as a crypto-mining and became a rampant threat since 2018. These challenges render many of the traditional detection techniques to be limited and ineffective to identify emerging sophisticated botnets, as will be discussed next.

There exist two approaches to botnet detection: signature-based detection and anomaly-based detection. The signature-based method is configured with a set of rules that are used to detect malicious network traffic based on pattern matching. This approach requires a relatively light computational complexity (e.g., based on optimized regular expression searches), but it is only able to identify "known" botnet patterns, critically limiting this approach without the functionality to identify new types of botnets before the corresponding pattern is published [10], [11]. On the other hand, the anomaly-based technique detects botnets by focusing on discovering network traffic anomalies (e.g., too high network latency, a spike of traffic volume, or unusual system behaviors) [10]. Traditionally, many studies relied on statistical features or heuristic methods to detect botnet anomalies [7], [8]. Recently, there has been a body of studies employing machine learning (ML) to analyze botnet behaviors, with the motivation of making more generalized botnet detectors. Several studies showed that previously "unseen" types of botnet attacks can be detected based on the characterized behaviors using ML [2]–[6].

One of the limitation of the past studies suggesting ML methods for botnet detection is that they do not experiment on the same datasets, which makes hard to compare methods to each other [2]–[6]. Also critically, previous studies do not much pay attention to *sequential patterns* within network data, even though botnet traffic shows repeated patterns due to the nature of the pre-programmed characteristics of bots [12]. Some studies limitedly considered sequential characteristics within the same source IP addresses, which removes its possibility to be used as an on-line detection system [13], [14]. This is because if using sequential pattern of each source IP address, we need to wait to collect every flow related to the IP address to classify one connection as *malicious* or *non-malicious*. In addition, existing studies narrow their scope by evaluating their methods only for one of IRC, P2P, and HTTP traffic, although botnets may utilize different (and

even a combination of) protocols. Since the detection method showing effectiveness on various types of botnets can be rendered reliable and practically useful, existing techniques are limited, and thus, ineffective to detect diverse types of botnets including previously uncovered botnet families [13]–[16].

In this paper, we propose a botnet detection method that is capable to capture *periodicity* within network data, which is a key to detect various classes of botnets showing sequential patterns, by utilizing a recurrent neural network. The proposed method also has a capability to detect botnets in an on-line manner with a new *anomaly scoring* function that represents the degree of maliciousness of network connections. The key contributions of this paper are tri-fold:

- We present a new ML model for botnet detection, which is built upon Recurrent Variational Autoencoder (RVAE) to capture sequential patterns. The proposed model learns from the normal data and detects potential anomalies that can vary over time in the context of botnet detection.
- We devise a strategy of anomaly detection which can be used in the streaming mode using the output from the RVAE network by utilizing probability density function of reconstruction errors.
- We verify that our approach could detect changing botnets by splitting the popular test data set CTU-13 into training and testing sets with different types of botnets. Tests show that we are able to detect botnets effectively when previously unseen types of botnets are used for testing compared to the existing methods.

II. Related works and Background

The various ML methods have been utilized in botnet detection. We use a fundamental structure of Recurrent VAE (RVAE) which contains both Variational Autoencoder (VAE) and Recurrent Neural Network (RNN). In the next subsection II-A, we introduce previous works utilizing ML techniques for botnet detection and discuss the limitation each work has. Subsequently, in the subsection II-B, we describe how each part of the proposed ML model works, and what problems each method was created to deal with.

A. Related Work

Variational Autoencoder: In [15], Guoc et al. introduce VAE which is an unsupervised method for detecting anomalies and also focus on explaining anomalies with a gradient-based fingerprinting technique, but it is limited it assumes that they already know the ratio of anomaly and it does not consider sequential pattern of data which can increase the performance of detection. Van et al. propose the revised VAE structure, called as Dirac Delta VAE, for achieving better anomaly detection performance in [16]. It narrows down the range of latent space which makes classifiers detect anomaly easier. However, the proposed method in the paper cannot be trained end-to-end because it separately uses classifier in the latent space. Furthermore, the authors conduct experiments using only one type of botnets for training and testing. Furthermore, there are various of studies utilizing VAE for anomaly detection of network traffic, as you can see in [17] and [18]. While many works have been done so far, most are limited in that the methods overlook sequential characteristics within network traffic.

Recurrent Neural Network: RNN has been in great use in many studies for employing sequential characteristics of network traffic data. Kapil et al. propose supervised approach to detect botnet hosts by tracking a host's network activity over time using RNN architecture and extract graph-based features of NetFlow data for botnet detection in [13]. However, extracted features are obtained by each host IP address. In addition, the method is restricted in terms of not being generalized in that it is trained and tested on the restricted botnet scenarios. Besides, Pablo et al. assign the symbol to size and the port, and embed the symbol to get distributed representation like word embedding [14]. It shows the potential to use RNN for botnet detection, but it doesn't show comparable performance in imbalanced network traffic. In [19], Egon et al. utilize text output from IDS as input of RNN. However, the method requires the specific type of output, which is text. While many works have been done so far, most are limited now that the methods cannot be applied to the on-line anomaly detection system. The methods require to collect every traffic related to the host IP addresses in order to distinguish whether they are malicious or not.

Other Machine Learning Approach: Besides VAE and RNN, many recent studies have attempted to make use of various ML approach to reduce the dependence for human heuristics. In [20], the authors regard every feature as sentence and embed it. With embedded features, classifier can be trained to detect malicious botnets. Kamaldeep et al. introduce the framework for P2P botnet detection using Random Forest (RF) in [3]. Ongun et al. present how to extract features which are good for ML model in [19]. The authors use a statistics aggregated feature processing method and validate the method with RF and Gradient Boosting.

B. Background of ML Structures

Variational Autoencoder (VAE) VAE is one of the generative models which utilizes deep neural network structure to represent transformation. Encoder which consists of neural network extracts latent variable z in accordance with input x. Decoder which also consists of neural network reconstructs x using z from the encoder. The more detailed discussion of VAE can be referred in [21].

Gated Recurrent Unit (GRU) We use RNN structure which is known as containing directed cycle beneficial to represent data with sequential pattern. Especially, we utilize GRU model which has capability to remember values with long sequences comparing to vanilla RNN. The more detailed discussion of GRU can be referred in [22].

Recurrent Variational Autoencoder (RVAE) RVAE is the structure of combining seq2seq with VAE, whose encoder and decoder consists of auto-regressive model. As it utilizes RNN instead of MLP to generate sequential outputs, it not only takes the current input into account while generating but also its

TABLE I
NOTATIONS USED

Notation	Description
$h_{E,T}$	The last hidden state of the encoder
W_μ	Linear transformation to get $\mu(x)$
W_σ	Linear transformation to get $\sigma(x)$
z	The latent variable
$h_{D,2}$	The second hidden state of the decoder
$h_{D,t}$	The hidden state of the decoder at timestep t
W^{hh}	Linear transformation from the previous hidden state
W^{hx}	Linear transformation from input
W^s	Linear transformation from hidden state to get output
θ	The parameters of encoder
x_1	The first input of the decoder
ϕ	The parameters of decoder
$\tilde{y_t}$	Output, reconstruction at timestep t
$\tilde{y_{t_n}}$	nth feature of reconstruction at timestep t
D_{KL}	Kullback-Leibler divergence

neighborhood. The more detailed discussion of Recurrent VAE can be referred in [23] and [24]. Generally, seq2seq models are actively being used in text and music field. The advantage to the RVAE is that it utilizes sequential patterns to generate the data. With this structure, we expect that the structure performs ably in network traffic data to detect anomaly.

III. PROPOSED MODEL

In order to identify botnets, we propose a novel flow-based botnet detection system coping with periodicity of traffic flow. The overall procedure of our proposed system consists of three steps as follows:

- **Data pre-processing:** The data instances are grouped based on a predefined time interval(e.g., 60 sec for the size of time window), and they are aggregated by host IP addresses. This process also includes the process of calculating statistical features and normalizing numerical values.
- **Anomaly scoring:** At every time window, anomaly scores of every flow are calculated, which provides the degree of maliciousness of individual connections. For scoring, we establish a function that consists of RVAE and produces anomaly scores by comparing the input with the output.
- **Anomaly detection:** Based on the calculated anomaly scores, the anomaly detection function classifies individual connections into either *Malicious* or *Non-malicious*. In particular, our method does not rely on *threshold*; rather, it utilizes a couple of probability density function (PDF) which are estimated by normal and botnet instances in training dataset, respectively.

A. Anomaly Scores from Recurrent Variational Autoencoder

Notations used in this paper are in Table I. We first input network traffic data, which are pre-processed, to GRU structure. $h_{E,T}$ is used as mean and variance of Gaussian distribution which represents latent space. With the mean and the variance, z can be obtained, and the z is used as initial hidden state for the decoder.

$$\begin{aligned} \mu(x) &= W_\mu h_{E,T} \\ \sigma(x) &= W_\sigma h_{E,T} \\ z &= \mu(x) + \sigma(x) * \epsilon, \epsilon \sim N(0,1) \end{aligned} \tag{1}$$

The first input of the decoder(x_1) is zero-padded. The second hidden state of decoder follows as:

$$h_{D,2} = sigmoid(W^{hh} z + W^{hx} x_1) \tag{2}$$

Finally, the outputs we obtain from RVAE is formulated:

$$\tilde{y_t} = sigmoid(W^s h_{D,t}) \tag{3}$$

The loss function that we want to minimize:

$$J(x) = -\mathbb{E}_{q_\phi(z|x)}[log p_\theta(x|z)] + \beta * D_{KL}[q_\phi(z|x)|p_\theta(z)] \tag{4}$$

As we use binary cross entropy as error function, the anomaly score is formulated:

$$L = \sum_{n=1}^{N} (1 - y_{t_n}) log(1 - \tilde{y}_{t_n}) + y_{t_n} log \tilde{y}_{t_n} \tag{5}$$

We train the model with only *non-malicious* instances, and in evaluation phase, we calculate reconstruction errors using both *non-malicious* and *malicious* instances. As we do not use *malicious* instances during training, we can develop the model detecting any unseen botnets. Each time window, we can obtain the anomaly scores of every connection which belong to the time window. In other words, the traffic connection can be classified by the outputs(L) of the anomaly detection system. In the following section, we present how to detect botnets with anomaly scores.

B. Anomaly Detection

In many studies, threshold of anomaly scores is used to distinguish whether the source IP addresses in the time window is malicious or not in anomaly detection methods [15], [17], [25]. The threshold can be set in many ways. It is one of the simple and intuitive method; however, the information of the dataset is required in most cases such as the ratio of botnets or at least approximate values of anomaly scores of botnet samples. Unfortunately, there are few cases that the information about the traffic data is known in advance. Furthermore, detecting attacks of botnet with threshold hampers the anomaly detection framework performing in streaming mode. Let's say that the threshold is set to 10%, which means that samples higher than top 10% of anomaly scores are classified as botnets. Then, we have to wait until all samples in testing dataset complete to get the anomaly scores because we must sort every anomaly scores. Therefore, this method is limited to being used in timely manner.

Instead, we suggest a more efficient method using the estimated probability distribution of reconstruction errors, which enables the method be applied in streaming mode. In training phase, we collect reconstruction errors from normal and abnormal instances. Then, we find the distribution and its parameters to represent the distribution of reconstruction errors of abnormal and normal, respectively, by exploring various types of distributions and selecting the minimum sum of squared estimate errors (SSE). We call the distribution with the smallest SSE as the *best-fit PDF*. We search for the *best-fit PDF* among many different candidates such as *gamma*

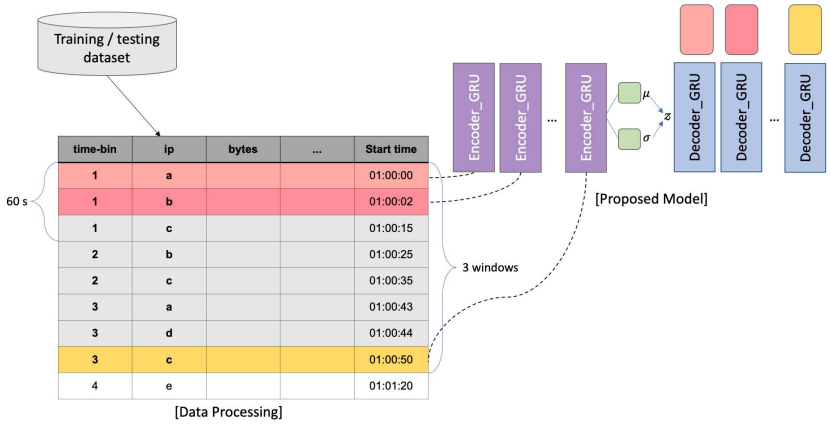


Fig. 1. The overall process of botnet detection system using RVAE with sequential dataset. In the phase of data processing, every flow sorted in chronological order is aggregated to obtain statistic features within the windows. In the botnet detection system, the encoder distills the common properties within the sequential data into latent variable z which the decoder reconstructs sequential inputs with. In the end, reconstruction loss is obtained as an output of RVAE.

distribution, *fold cauchy* distribution, *Mielke* distribution and *beta* distribution, among others. In testing phase, the estimated PDF can be utilized to obtain likelihood. Comparing the likelihoods of the two different distributions, we assume that each sample of the test data set belongs to the distribution with greater likelihood. Utilizing *best-fit PDFs* at the training stage does not require the information of network traffic dataset as well as enables the botnet detection system be used in an online manner.

IV. Experiments

We show the two aspects from the experiments. First, the proposed method has better performance than both RF and the existing standard VAE, which we call as MLP-VAE in this paper. Second, we explain how the reconstruction errors are distributed.

A. Evaluation Datasets

We use CTU-13 dataset which is widely used in the latest studies for botnet detection [14]–[17], [19], [20], [25], [26]. A botnet scenario is a particular infection of the virtual machines using a specific malware. Thirteen of these scenarios were created, and each of them was designed to be representative of some malware behavior [26]. To compare the results of MLP-VAE and RF, we reproduced nearly the same experimental settings with the settings in [15] and [19]. In [15] and [19], they prove the robustness of their methods on scenario 1, 2 and 9 of CTU-13 dataset, which consists of only botnet called Neris. The Neris is IRC based bot infecting other machines by Spam and Click Fraud. In our reproduced experiments, all methods show similar performance in every metrics, as you see in Table II. Especially, RF performs very well on the testing datasets because botnet families in the testing dataset are already used for training. In other words, RF method is able to capture dominant features to classify anomalies. However, when considering the evolving botnets, the method cannot be validated with being evaluated on dataset consisting of botnets which are previously identified.

Thus, we determine to follow the dataset separation criteria, as suggested in [26], in order to test the model in more general cases. In [26], the authors made the dataset in a way that none of the botnet families used in the training and cross-validation datasets should be used in the testing dataset. The authors state that this way ensures that the detection methods can generalize and detect new behaviors. By splitting of CTU-13 data in the suggested way, we can mimic the real situation where the operations of botnet changes over time in terms of protocols and attack types. Compared to the restricted dataset (scenario 1, 2, 9), various types of botnets that have IRC-based, P2P-based and HTTP-based communication methods and conduct attacks such as Spam, Click Fraud, Port Scan, DDos and FastFlux are included in the entire dataset.

B. Data Pre-processing

The CTU-13 dataset consists of NetFlow files which are composed of source and destination IP addresses and ports, time, protocol, duration, number of packets, number of bytes, state, and service. We process the data to use the aggregated flows statistic, which is the way many existing works adopt in order to obtain flow-based features [14]–[17], [19], [25]. We group NetFlow data at every time interval of T, and aggregate features within every group based on the source IP addresses to get flow-based features. With the processing method, we can detect IP address showing malicious behavior in a particular time window. Many existing works experimentally find the most appropriate time window T, which is crucial in that while too small time window might not capture traffic characteristics over a longer period of time, too large time window cannot provide timely detection in waiting the end of the window [6], [14], [15], [17], [19], [25], [26]. We did experiment as changing the duration of windows to find the ideal value for the statistical aggregation. We then sort the entire data within the time window by the time of the source IP connection, because the RNN model is sensitive to the order of the inputs. For RVAE, we use the network traffic connections collected within N windows as the sequential inputs to the model. In the case of Fig. 1, 60-second duration of three windows are used.

In terms of source/destination ports and destination IP addresses, we count the number of unique records with connected source IP addresses in the time window. For service, state, and protocol, we count the number of different values in each category with the source IP addresses in the time

TABLE II
RESULTS COMPARISON : TRAINED AND TESTED ON SCENARIO 1,2 AND 9

Model	Recall	Precision	F1	AUPRC	AUROC
RVAE	0.978	0.957	0.967	0.960	0.966
MLP-VAE	0.974	0.959	0.966	0.959	0.966
Random Forest	**1.000**	**0.998**	**0.999**	**1.000**	**1.000**

TABLE III
RESULTS COMPARISON : TRAINED AND TESTED ON THE ENTIRE DATASETS

Model	Recall	Precision	F1	AUPRC	AUROC
RVAE	**0.969**	0.892	**0.929**	**0.972**	**0.975**
MLP-VAE	0.944	0.891	0.917	0.967	0.962
Random Forest	0.424	**0.982**	0.592	0.888	0.901

window. Finally, we normalize the numerical values to be between 0 and 1. As a result, the number of features used in this experiment is smaller compared to the number of features used in [19] and [15].

C. Experimental Setting

For splitting datasets for training, validation and testing, we followed the suggested separation criteria, as we mentioned in the section IV-A. The architecture of MLP-VAE follows what is used in [15], [# of features $\rightarrow$ 512 $\rightarrow$ 512 $\rightarrow$ 1024 $\rightarrow$ 100]. For RNN architecture, we use 2-layer bidirectional GRU. We use the 512 dimensions of hidden states, and 100 dimensions of latent variable as MLP-VAE. We also apply *ReLU* activation to MLP-VAE as well as RVAE. We train for 500 epochs with Adam optimizer and 128 batch-size. Also, learning rate is set as 0.01 for both VAE models. We use the 5 different evaluation metrics to validate our performance; Area Under the Receiver Operating Characteristics (AUROC), Area Under the Precision-Recall Curve (AUPRC), Precision, Recall and F1 score. We save the model showing the best value of AUPRC in 5-fold cross validation sets.

V. RESULTS AND DISCUSSION

We did experiments to validate the proposed method in different aspects. For quantitative validation, we compare the performance of our method with other methods on different metrics. For qualitative validation, we plot the distribution of the reconstruction errors of normal and botnets cases. Moreover, we plot estimated the *best-fit PDF*. To compare methods with the same processing and detection method, we reproduce MLP-VAE and RF in [15] and [19], respectively.

Performance comparison among methods In Table II, RFmethod shows the nearly perfect performance in every metric, even though VAE models show the comparable performance. It is because that the training/testing datasets which are based on scenario 1, 2 and 9 share the same characteristics. RF is effective in finding dominant features in these restricted datasets. However, as we mention in the section IV-A, validating the models on the characterized datasets is not what we focus on in this paper. In Table III, we show the results from the training and testing on the generalized dataset that we mentioned in the section IV-A. In this experiment, we pre-processed our data by using 60-second duration of window and using three windows. There is an overall performance degrade with RF which is affected by the dominant features of the training dataset. Nonetheless, VAE methods validate its reliability by showing the robust performance with the generalized dataset. In addition, we find that RVAE method outperforms MLP-VAE method overall based on the same features and the same size of latent variables on both datasets, as you see in Table II and Table III. It can be concluded that the botnets of network traffic flow data should be detected utilizing sequential and periodic patterns.

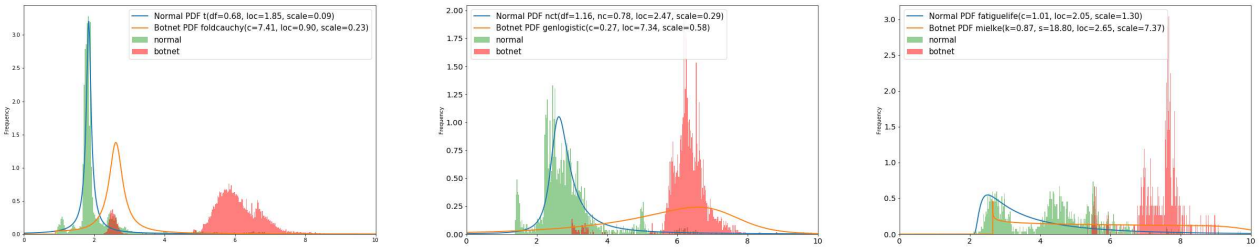
TABLE IV
RESULTS COMPARISON : DIFFERENT WINDOW DURATION

Window duration(s)	Recall	Precision	F1	AUPRC	AUROC
5	**1.000**	0.865	0.928	0.791	0.881
60	0.969	**0.892**	**0.929**	**0.972**	**0.975**
300	0.998	0.537	0.699	0.905	0.972

Probability density function of reconstruction errors As shown in Fig. 2, the distribution of the reconstruction errors of botnet samples can be distinguished from the distribution of the normal sample reconstruction errors. As we only use *non-malicious* samples for training, we expect that the reconstruction errors of *malicious* samples are larger than that of the *non-malicious* samples. Comparing medians of those two distributions, we can intuitively notice that the median of the distribution of *non-malicious* reconstruction errors is larger than the median of the distribution of botnet reconstruction errors.

Especially, you can find a group of botnet samples which have the smaller reconstruction errors compared to the other botnet samples in Fig. 2b. We focus on the samples whose reconstruction errors are smaller than 4. We find that 66% of the samples of the scenario 6 labeled as botnet show the reconstruction errors less than 4, while only 0%$\sim$4% of samples in the other scenario show reconstruction errors less than 4. The scenario 6 utilizes proprietary command control channels unlike other scenarios most of which use IRC, HTTP and P2P communication methods [26]. The samples of the group having small reconstruction errors show low values for DNS, smtp, ssl, the number of IP addresses, the number of ports, and the number of different IP addresses in window. These characteristics mainly represent *non-malicious* other than the botnet. We conclude that the general nature which can be found in the scenario 6 makes dozens of samples belonging to the scenario obtain the smaller reconstruction errors.

Duration of window In order to propose the right duration of window, our experiments have been done with changing the duration of window to 5 seconds, 60 seconds and 300 seconds in Table IV. In general, the performance of 60-second duration of window are higher than those of other duration lengths. We infer that as we use a long duration, the number of source IP addresses which belongs to the same time window increases, which aggravates the vanishing gradients problem in a long-term sequence. On the other hand, too short duration cannot provide efficient length to represent the patterns of the time windows with statistically aggregated values. From our experiments, 60-second duration is the most suitable, as you can find in Table IV quantitatively and Fig. 2 qualitatively.



(a) duration of window : 5s (b) duration of window : 60s (c) duration of window : 300s

Fig. 2. Distribution of reconstruction errors of *normal* and *botnet* samples with different duration of window

VI. CONCLUSION

In this paper, we validate RVAE anomaly detection method taking into account for the sequential and periodic nature for the network traffic flow data. The study is of significance to providing the applicable solution for the botnet detection system, which is able to be used in an on-line manner. Moreover, as the proposed method is validated on various scenarios of botnet operation, including the botnets which are not used for training, it can be concluded that the proposed method is robust in detecting previously unseen botnets.

For future studies, fuzzy logic can be adapted to improve the anomaly detector utilizing PDF. It can provide more logical and systematic way of using PDFs for anomaly detection than comparing likelihoods from two distributions. In addition, it is potential to improve performance of the anomaly detector if the method copes with some cases of botnets having small reconstruction errors in the normal cases. Moreover, setting online evaluation environment can help to prove its anomaly detection performance in the streaming mode.

VII. ACKNOWLEDGEMENT

This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231, and also used resources of the National Energy Research Scientific Computing Center (NERSC).

REFERENCES

[1] K. Alieyan, A. ALmomani, A. Manasrah, and M. M. Kadhum, "A survey of botnet detection based on dns," *Neural Computing and Applications*, vol. 28, no. 7, pp. 1541–1558, 2017.

[2] G. K. Venkatesh and R. A. Nadarajan, "Http botnet detection using adaptive learning rate multilayer feed-forward neural network," in *IFIP International Workshop on Information Security Theory and Practice*, pp. 38–48, Springer, 2012.

[3] K. Singh, S. C. Guntuku, A. Thakur, and C. Hota, "Big data analytics framework for peer-to-peer botnet detection using random forests," *Information Sciences*, vol. 278, pp. 488–497, 2014.

[4] E. B. Beigi, H. H. Jazi, N. Stakhanova, and A. A. Ghorbani, "Towards effective feature selection in machine learning-based botnet detection approaches," in *2014 IEEE Conference on Communications and Network Security*, pp. 247–255, IEEE, 2014.

[5] M. Stevanovic and J. M. Pedersen, "An efficient flow-based botnet detection using supervised machine learning," in *2014 international conference on computing, networking and communications (ICNC)*, pp. 797–801, IEEE, 2014.

[6] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, and D. Garant, "Botnet detection based on traffic behavior analysis and flow intervals," *Computers & Security*, vol. 39, pp. 2–16, 2013.

[7] J. R. Binkley and S. Singh, "An algorithm for anomaly-based botnet detection.," *SRUTI*, vol. 6, pp. 7–7, 2006.

[8] G. Gu, J. Zhang, and W. Lee, "Botsniffer: Detecting botnet command and control channels in network traffic," 2008.

[9] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, *et al.*, "Understanding the mirai botnet," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, pp. 1093–1110, 2017.

[10] H. R. Zeidanloo, M. J. Z. Shooshtari, P. V. Amoli, M. Safari, and M. Zamani, "A taxonomy of botnet detection techniques," in *2010 3rd International Conference on Computer Science and Information Technology*, vol. 2, pp. 158–162, IEEE, 2010.

[11] M. Roesch *et al.*, "Snort: Lightweight intrusion detection for networks.," in *Lisa*, vol. 99, pp. 229–238, 1999.

[12] B. AsSadhan, J. M. Moura, D. Lapsley, C. Jones, and W. T. Strayer, "Detecting botnets using command and control traffic," in *2009 Eighth IEEE International Symposium on Network Computing and Applications*, pp. 156–162, IEEE, 2009.

[13] K. Sinha, A. Viswanathan, and J. Bunn, "Tracking temporal evolution of network activity for botnet detection," *arXiv preprint arXiv:1908.03443*, 2019.

[14] P. Torres, C. Catania, S. Garcia, and C. G. Garino, "An analysis of recurrent neural networks for botnet detection behavior," in *2016 IEEE biennial congress of Argentina (ARGENCON)*, pp. 1–6, IEEE, 2016.

[15] Q. P. Nguyen, K. W. Lim, D. M. Divakaran, K. H. Low, and M. C. Chan, "Gee: A gradient-based explainable variational autoencoder for network anomaly detection," in *2019 IEEE Conference on Communications and Network Security (CNS)*, pp. 91–99, IEEE, 2019.

[16] M. Nicolau, J. McDermott, *et al.*, "Learning neural representations for network anomaly detection," *IEEE transactions on cybernetics*, vol. 49, no. 8, pp. 3074–3087, 2018.

[17] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE*, vol. 2, no. 1, 2015.

[18] A. Kumagai, T. Iwata, and Y. Fujiwara, "Transfer anomaly detection by inferring latent domain representations," in *Advances in Neural Information Processing Systems*, pp. 2467–2477, 2019.

[19] T. Ongun, T. Sakharaov, S. Boboila, A. Oprea, and T. Eliassi-Rad, "On designing machine learning models for malicious network traffic classification," *arXiv preprint arXiv:1907.04846*, 2019.

[20] F. Du, Y. Zhang, X. Bao, and B. Liu, "Fenet: Roles classification of ip addresses using connection patterns," in *2019 IEEE 2nd International Conference on Information and Computer Technologies (ICICT)*, pp. 158–164, IEEE, 2019.

[21] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

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

[23] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," *arXiv preprint arXiv:1511.06349*, 2015.

[24] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," *arXiv preprint arXiv:1803.05428*, 2018.

[25] R. Dargenio, S. Srikant, E. Hemberg, and U.-M. O'Reilly, "Exploring the use of autoencoders for botnets traffic representation," in *2018 IEEE Security and Privacy Workshops (SPW)*, pp. 57–62, IEEE, 2018.

[26] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *computers & security*, vol. 45, pp. 100–123, 2014.