Marrying Graph Kernel with Deep Neural Network: A Case Study for Network Anomaly Detection

Yepeng Yao[1,2*], Liya Su[1,2*], Chen Zhang[1], Zhigang Lu[1,2(✉)], and Baoxu Liu[1,2]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{yaoyepeng, suliya, zchen, luzhigang, liubaoxu}@iie.ac.cn
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

Abstract. Network anomaly detection has caused widespread concern among researchers and the industry. Existing work mainly focuses on applying machine learning techniques to detect network anomalies. The ability to exploit the potential relationships of communication patterns in network traffic has been the focus of many existing studies. Graph kernels provide a powerful means for representing complex interactions between entities, while deep neural networks break through new foundations for the reason that data representation in the hidden layer is formed by specific tasks and is thus customized for network anomaly detection. However, deep neural networks cannot learn communication patterns among network traffic directly. At the same time, deep neural networks require a large amount of training data and are computationally expensive, especially when considering the entire network flows. For these reasons, we employ a novel method *AnoNG* to marry graph kernels to deep neural networks, which exploits the relationship expressiveness among network flows and combines ability of neural networks to mine hidden layers and enhances the learning effectiveness when a limited number of training examples are available. We evaluate the proposed method on two real-world datasets which contains low-intensity network attacks and experimental results reveal that our model achieves significant improvements in accuracies over existing network anomaly detection tasks.

Keywords: Network Anomaly Detection · Deep Neural Network · Graph Kernel · Communication Graph Embedding

1 Introduction

The Internet has become an important part of our society in many areas, such as science, economics, government, business, and personal daily lives. Further, an increasing amount of critical infrastructure, such as the industrial control

* The two lead authors have contributed equally to this study.

systems and power grids, is managed and controlled via the Internet. However, it is important to note that today's cyberspace is full of threats, such as distributed denial of service attack, information phishing, vulnerability scanning, and so on. Recent news also points out that attackers can build large organization consisting of different types of devices and bypass existing defenses by changing communication patterns [6].

Despite these difficulties, network anomaly detection is still making progress. Network anomaly detection systems identify network anomalies by monitoring and analyzing network traffic and searching for anomalous flows with malicious behaviors. Despite having high accuracy rates and low false alarm rates, these systems are easily evaded by recent collaborative-based anomalies, even anomalies with the similar packets as legitimate traffic. Moreover, existing detection modules usually consider less about the communication relationship, such as the communication pattern between inner and outer networks, so that it is difficult to describe anomalies from single flow itself in network.

Collective network anomalies are major kinds of network anomaly nowadays [12]. According to the communication volume of the traffic flows, individual steps can be roughly classified into two types, namely high-intensity attacks and low-intensity attacks. High-intensity attacks such as Smurf [9] transmit packets at a high rate causing a sudden surge in traffic volume whereas low-intensity attacks such as Slowloris [19] are high-level attacks that do not rely on intensity and volume to breakdown network service. Other low-intensity attacks do not attempt to completely disrupt the service but rather degrade the quality of service over a long period of time to achieve economic damage or reduce the performance of servers, indicates that all the packets involved in the attack are not detected. In complex attacks, an attacker floods the target system with low-intensity, highly targeted, and application-specific traffic.

Recent research has shown that the following challenges that are not tackled in previous researches. We briefly describe two of the modern challenges for network anomaly detection that we will address in this work.

Challenge 1: Low-intensity volume of network anomaly traffic might seem innocuous. Most traditional methods analyze statistical changes of traffic volume. The statistical features of the network flows contain information such as the number of flows, the number of packets of the flow, the average length of packets of the flow, the network protocol flags, and the duration of the flow. If a statistical mode mismatch in those values is detected, the event is reported as an anomaly. For example, if the statistical features of the destination port numbers suddenly increase, it is assumed that some hosts conduct port scanning. However, the anomalous activities that may not increase the number of packets are not visible in a large network. Those volume-based anomaly detection techniques, however, cannot detect low-intensity attacks.

Challenge 2: More advanced evasion techniques can conceal real attacks. Even though some flow features have the ability to capture anomalous traffic that has statistical variations of traffic volume, they cannot capture correlations in communications between hosts because they focus on the statistics

of traffic sent by individual hosts. For example, command and control communications, the major source of advanced cyber anomalies, consist of attackers that originate attack commands. The attackers make up a cluster where they communicate with each other. With traffic statistics-based detection methods, we can detect partial anomalies but cannot detect or capture the overall anomalies, especially for the concealed network attacks.

By exploiting the above two challenges, we adopt communication graph embedding method to represent the structural similarities of communication patterns by calculating the similarities between graphs generated by network flows. For the reason that network anomalies may not occur as one single action on a single host, there may be many small seemingly innocuous multi-hosts behaviors. Attackers might do several small probe actions that are temporally spaced or they might initiate a massive denial of service attack. Each change of anomalies has a specific pattern of a particular feature set.

In this paper, we investigate the benefit and efficiency of applying the graph kernel embedding approach to improve the deep neural networks. Our proposed approach aims to improve the accuracy of network anomaly detection process and analyze through an attributed communication graph by using communication graph features, which stores the representation of the status of the participating networks in a specific time interval. Our contribution in this paper is threefold:

- To understand the new network threat, we propose a novel network flow analytics architecture for low-intensity network anomaly that can evade other detection approach easily;
- To detect anomaly flows and improve the accuracy of network anomaly detection process, we develop the first work to systematically combine graph kernels with deep neural networks for network anomaly detection;
- To evaluate the proposed method, we build a working system to hold a sliding time window view of the entire network based on the structural similarity analysis of network flows. In this way, it is optimal for the application of machine learning techniques. We thoroughly evaluate our system with two real-world datasets and demonstrate that the proposed method effectively outperforms the traditional machine learning solutions and the baseline deep neural network detection solutions in terms of accuracy.

The rest of the paper is organized as follows. Backgrounds and previous work related to this paper are discussed in Sect. 2. Sect. 3 described the threat model and major approaches. The proposed anomaly detection framework is discussed in Sect. 4 followed by the empirical evaluation in Sect. 5. Last, Sect. 6 concluded this paper.

2 Background and Related Work

We first lay out the background information of network evasion attacks against anomaly detection models. Then we review related prior research.

2.1 Network Evasion Attack against Anomaly Detection Models

Since attackers have become more organized, skilled and professional, research in network anomaly detection must always take into account the presence of an evasion attack. While some detection papers are providing discussion on evasion resistance, there is rarely any practical method to increase evasion resilient capabilities of the anomaly detection models. In fact, the purpose of evasion attacks is to move an anomalous flow to a benign one, or to attempt to mimic a specific benign point. Therefore, the attacker attempts to modify the features of the anomalous flow so that the anomaly detection model marks the flow as benign.

The limitation of evasion attacks is performed in two ways. First, there could be features that are not modifiable to the extent required to perform the attack. There is also a limitation on how difference the anomalous activities can be changed, because the activities still need to follow their original, anomalous purposes. For example, the target IP addresses of network flows cannot be modified arbitrarily, so the communication patterns of anomalous activities are relatively stable. The evasion attack also needs understandings of what is considered benign in the production detection model, which is usually not available.

2.2 Deep Neural Network based Network Anomaly Detection

Deep neural networks have been shown tremendous performance on a variety of application areas, such as image, speech and video analysis tasks, while they are high-dimensional inputs and have high computational requirements. For network anomaly detection, deep neural networks increase the detection rate of known anomalies and reduce the false positive rate of unknown anomalies.

Alom et al. [1] train deep belief network models for identifying any kind of unknown anomalies in dataset and evaluated the performance on intrusion detection datasets. Their proposed system not only detects anomalies but also classifies them and achieves 97.5% accuracy for only fifty iterations. Alrawashdeh et al. [2] explore the anomaly detection capabilities of restricted boltzmann machine and deep belief network. They outperform the former works in both detection speed and accuracy and achieve a detection rate of 97.9% presenting machine learning approaches for predicting anomalies with reasonable understood. As for deep belief neural network [11], it achieves accuracy as high as 99% combining neural networks with semi-supervised learning in a smaller percentage of labeled samples.

In the case of Convolutional Neural Network(CNN)-based and Long Short-Term Memory(LSTM)-based anomaly detection, Zhang et al. [20] propose a specially designed CNN to detect network anomalies. They find only some preprocessing is needed whereas the tedious feature extraction is done by the CNN itself and the experimental results show that the designed CNN model has a good performance. Chawla et al. [4] employ recurrent neural network(RNN) as their model for host-based intrusion detection systems which determine legitimate behavior based on sequences of system calls. Kim G et al. [7] propose

a system-call language-modeling method on host-based intrusion detection systems. Their method learns the semantic meaning and interactions of each system call which has never been considered before and the validity and effectiveness are shown on public benchmark datasets.

2.3 Graph based Anomaly Detection

Several graph-based and machine learning techniques have been investigated over the last two decades for detecting network anomaly detection. Our previous work [18] proposed an effective method for deep graph feature extraction to distinguish network anomaly in network flows.

The graph kernel approach is a rapid feature extraction scheme and its runtime scales only linearly in the number of edges of the graphs and the length of the subgraph sequence, by converting a high-dimensional graph into a feature vector space. The scalar product of the two graphs in space measures their similarity. Generally, graph kernels are utilized to define the similarity function of two graphs. One effective way is the Weisfeiler-Lehman(WL) kernel [16]. However, WL kernel only support discrete features and costs linear memory of training examples. Other methods include depth graph kernel [17] and graph invariant kernel [13] comparing based on the existence or counting of the small substructures, such as the shortest path [3], graphlets, subtrees, and other graph invariants [13]. We adopt general shortest path kernel as one possible labeling procedure to compute receptive fields.

3 Threat Model and Our Approach

To better understand the efficiency of the method we proposed in this paper, we describe the threat model and the defense model. The threat that we study as a use case is a kind of the denial of service attack that exhausts the resources using low-intensity traffic flows described in [8], seen in Fig. 1. We consider an outside attacker that can send arbitrary network flows to a server or link that is vulnerable to network anomalies. The attacker can exploit the vulnerability by sending carefully crafted flows that will consume a significant amount of the resources. The attackers' goal is to occupy available resources by sending multiple flows in parallel at a low intensity.

In low-intensity attacks, the attacker can use controller servers to attack the targets. To achieve this goal, the attack assigns botnets controlled by his controller servers to send legitimate, low-intensity network flows towards certain target servers. Since anomalous flows are well-formed, they cannot be easily distinguished from legitimate flows through statistical features, such as the average packet size. The attacker can also send legitimate flows to hide attack trace among legitimate network flows. The attacker does not send numerous attack flows within a very short time interval, because volumetric attacks with a high intensity can be easily detected by network-based anomaly detection, and attacks with low-intensity are already sufficient to affect the network.

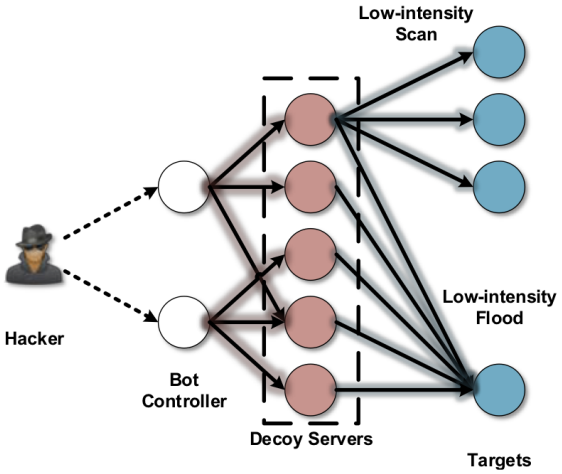


Fig. 1. A use case for threat model: a low-intensity version of the network anomalies.

3.1 Problem Definition

Given a set of network flows, we can detect several kinds of anomalous network flows, but some kinds of anomalies can be evaded by modifying several statistical flow features. In fact, similar communication patterns represents on communication graphs, for example, when the network flows come from the similar sources and go towards the similar destinations.

Communication graphs are used as abstraction of network communication patterns. Hosts are represented by vertices, while communication between them is indicated by edges, weighted by the statistical features of exchanged data. Given a directed multigraph $G < V, E >$, the purpose of the anomaly detection is to relate a set of vertices to a set of edges. An anomaly at time interval $[t, t+1]$ attacks a set of host $E_{A(t)} \in E$, affecting the connectivity of a set of vertices $V_{A(t)} \in V$. As described in [8], the goal of the anomaly detection is to find the malicious host set and detect the malicious communication flows until $t + 1$.

Assume that $p(e)$ expresses the probability of e acting as a malicious host at time interval t, $p(v)$ is the probability of v being an target at time interval $[t, t + 1]$, and $p(e, v)$ is the probability of e attacking v at this time interval. The goal is to seek to minimize the total possibility of $p(e)$ over all time intervals. In the neural networks model, system needs to extract the information about $p(v)$ and seek to minimize $p(v)$. However, attackers are able to do little changes to evade the single time interval detection without much effort. While it is difficult to evade from the whole communication graph pattern which graph kernels have the ability to describe, that is, graph kernels can capture the relationship between $p(v)$ and $p(e)$, that is $p(e, v)$.

For estimating the similarity between communication patterns on a pair of communication graphs, we adopt graph kernel functions. It relies on graph comparison. It is clear that there is no link between these network flows because

they happened at different time. We construct the communication graph for each network flow using a specific time interval.

3.2 Communication Graph Embedding

Most existing machine learning or neural network algorithms are generally applicable to vector-based data. In order to enhance interpretability of deep neural networks and learn accurate graph feature embedding, it is necessary to extract the features of graph similarity data in advance. Algorithm 1 presents the details of communication graph feature representation.

First, we extract the statistical flow features. These features are extracted from each network flow, which directs source IP address to destination IP address and contains statistical features that are commonly used for anomaly detection.

Then we construct the communication graph for each network flow by a specific time interval. Given a set of graphs, many algorithms have been designed and widely used for similarity analysis on graphs. While determining all paths is NP-hard, finding special subsets of paths is not necessarily. Determining longest paths in a graph is also NP-hard. To overcome this problem, the shortest path similarity measures [3] are selected in this paper, which capture diverse aspects of the network communication pattern and can be efficiently computed.

According to the sorted hosts and the length of shortest paths, the shortest path compares the similarity of two communication graphs. Using the network flows of each pair of hosts noted as i and j constructs the similarity metrics. The shortest path $d(i, j)$ represents the shortest distance from host i to j. With low resolution the vertices are more likely to interact which is what our model wants to detect recorded as $p(h, v)$ (seen in Section 3.1).

At the same time, the shortest path also describes the communication situation of pair of hosts as toward or away from each other in the network flow graph. As discussed in [5], given the small world property of networks, we can limit the distance searched before stopping and supposing no other paths.

4 Graph Kernels Combine Deep Neural Networks Framework

We propose AnoNG framework by combining deep neural networks and graph kernels from two different angles. We adopt the LSTM and CNN as our baseline deep neural network models because they are effective for network anomaly detection. There are two ways to extend the kernel vectors for DNNs. We can append the kernel vectors either to input vectors, or to the softmax classifier. To determine which strategy works best, we perform a pilot study. We found that the first method causes the dimension of features turn to be too large, so we use dimensional reduction method before inputting the DNN. While for the second method, the softmax classifier tends to heavily rely on the kernel vectors, and the result is similar to the one only given kernel features, so we design a weight variable to calculate the final results. The architecture of AnoNG framework can be seen in Fig. 2.

Algorithm 1 Communication Graph Embedding

Input:

A network communication graph set $\{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_N\}$ with N communication graphs;

Output:

A $|N| \times |N|$ kernel matrix K_{SP} as similarity features with K_{SP_N} assigning to $\mathcal{G}_N$ as its embedding feature for similarity measurement;

1: Computing the shortest path graphs $S < V, E >$ of each network communication graph: Where the weight of each edge in E is equal to the length of the shortest path between the corresponding vertices in $\mathcal{G} < V, E >$

2: Computing the graph kernel matrices K_{SP} for each pair of graphs: For each pair of graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ with $S_1 < V, E >$ and $S_2 < V', E' >$

$$K_{SP}(\mathcal{G}_1, \mathcal{G}_2) = \sum_{e \in E} \sum_{e' \in E'} \kappa(e, e')$$

where κ is shortest path kernel defined on the edges of S_1 and S_2, which measures the similarity between local sub-structures centered at e and e'

3: Computing the kernel matrix for the N graphs: Compute the kernel matrices K_{SP}.

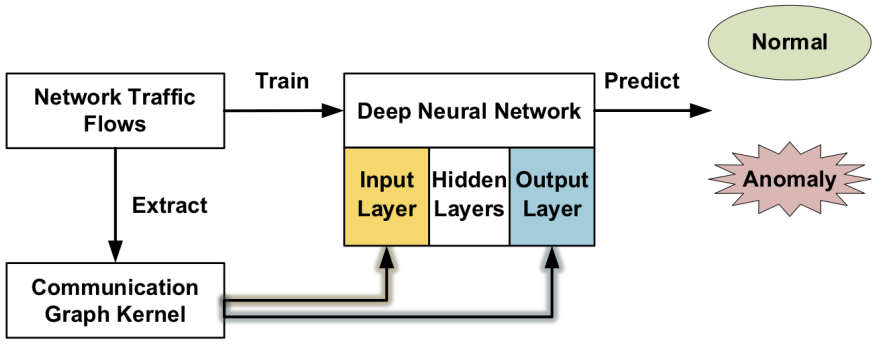


Fig. 2. Overview of the network anomaly detection framework using deep neural networks and graph kernels (AnoNG).

4.1 Integrating Graph Kernels at Input Level

At the input level, we use the evaluation outcomes of graph kernels as features fed to DNN models as Fig. 3(a). The number of graph features for network anomaly detection can be different in detailed situation. We extract network flow features according to effectiveness, and graph kernels' number can change depending on computational efficiency.

Because of the conflicting situations illustrated above, we cannot directly add or multiply the kernel vectors to flow features. To solve this issue, we extend the flow features with kernel features to form aggregated features as the input of DNNs.

4.2 Integrating Graph Kernels at Output Level

At the output level, graph kernels are used to improve the output of DNNs in Fig. 3(b). Since the graph kernels used in the input level only extend the statistical network flow features, we can also dig further expression by DNNs. So we combine the softmax classifier results of DNNs as the final results.

For instance, the softmax classifier of graph kernels can output the probability distribution $K_p = (K_{p1}, K_{p2})$ of two classes. DNN outputs the probability distribution $N_p = (N_{p1}, N_{p2})$ for two classes. We define the final probability of the label value for a network flow is:

$$p = a_1 K_p + a_2 N_p \tag{1}$$

where $a_1 + a_2 = 1$ and K_p is the probability distribution produced by the graph kernels and N_p is the probability distribution produced by the baseline deep neural network model. And a_1 and a_2 are trainable weights indicating the overall confidence for anomalous or benign labels. Here we assign 0.1 and 0.9 for original trainable weights and use 0.1 as interval. We modify 0.1 as interval when training a_1 and a_2 instead of more precise interval because using 0.1 is precise enough for the difference of two classes. Actually, using more precise interval such as 0.01 has the same effect as interval 0.1.

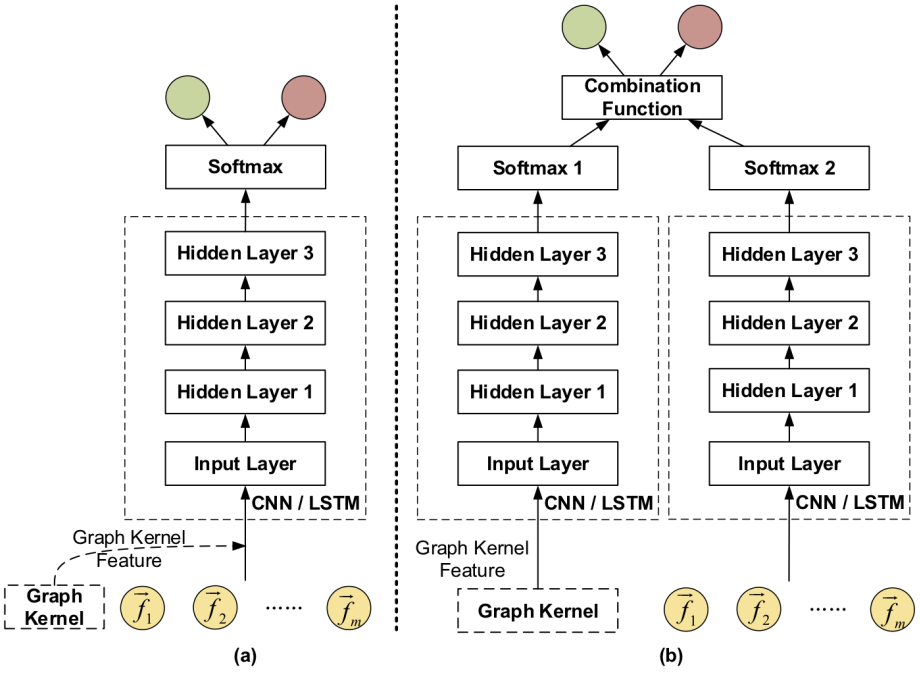


Fig. 3. Combination of graph kernels and deep neural networks.

5 Evaluations

In this section, we evaluate the proposed AnoNG framework. To evaluate the anomaly detection performance of our proposed AnoNG detection module, we adopt the UNSW-NB15 [10] dataset and the CIC-IDS-2017 [15] dataset, respectively. Both the datasets labeled the network anomaly flows, which means we have ground truth of the traffic. Then, we divide each dataset into training and test data sets using a ratio of 70% and 30%.

5.1 Dataset Description

UNSW-NB15 Dataset: This dataset is created by establishing a synthetic environment at the UNSW cybersecurity lab. The data presents a mixture of the real modern normal and the contemporary synthesized attack activities of the network traffic. It had 47 features including new low-intensity attack scenarios and 9 significant series of the cyberattack. We select only 8 statistical network flow features from this dataset, namely *dur, sbytes, dbytes, Sload, Dload, Spkts, Dpkts, Sintpkt, Dintpkt.*

CIC-IDS-2017 Dataset: This dataset is a newer dataset that contains a more recent form of anomalies such as high-intensity high level layer attacks generated using botnets and low-intensity attacks generated using Slowloris tool. We also select 8 statistical network flow features from this dataset, namely *Flow Duration, Total Fwd Packets, Total Backward Packets, Total Length of Fwd Packets, Total Length of Bwd Packets, Flow Bytes/s, Flow Packets/s, Average Packet Size.*

5.2 Experimental Environment

We implemented a proof-of-concept version of our AnoNG framework (see Section 3 and Section 4). Python3, Scikit-learn and Keras with TensorFlow backend are used as the software frameworks, which are run on the Ubuntu 16.04 64-bit OS. The server is a DELL R720 with 16 CPU cores, 128 GB of memory and NVIDIA Quadro GPU. To training the deep learning methods, we set the batch size to 128, the learning rate is set to 0.001, the epochs of training as 10, and the dropout rate is set to 0.5.

For evaluation, we report precision, recall, and F1-score, which depend on the four terms true positive (TP), true negative (TN), false positive (FP) and false negative (FN) for the anomaly detection results achieved by all methods.

We conduct three experiments: baseline DNNs, integrating graph kernels at input level of DNNs, and integrating graph kernels at output level of DNNs on two real-world datasets. We select t as 60 seconds as the time window to construct communication graphs as it was shown as a suitable classification point. All experiments were repeated 10 times and we report the means of evaluation matrices for all algorithms. The best results are bolded. For the compared methods, both LSTM and CNN are considered as baselines. To provide a better

overview of the performance of the baseline approach on the statistical network flow features, the overall precision, recall and F1-score are presented in Table 1.

We set dimensions of the CNN layers to input-128-256-128-output for both datasets. Dimensions of the LSTM layers are set to input-128-256-128-output, where the output is the statistical feature dimension, which varies between datasets. All layers are densely (fully) connected. To training the deep learning methods, we set the batch size to 128, and set the learning rate to 0.001. The epochs of training is set to 10, and the dropout rate is set to 0.5. The ADAM optimizer is adopted for variant of gradient descent.

Both the datasets are divided into two sets: training set and test set, which are the mixture of both anomalies and benign ones. Then, a deep neural network model is trained to predict on the training set and the prediction errors on the training set are fit to a multivariate Gaussian using maximum likelihood estimation. The loss function that we adopt is the sigmoid cross entropy. The threshold for discriminating between anomalous and benign values is then determined via by maximizing the accuracy value with respect to the threshold. At last, the trained model is then used to predict on the test set and the results are recorded and compared.

5.3 Performance on DNN with Graph Kernels at Input Level (GKIL)

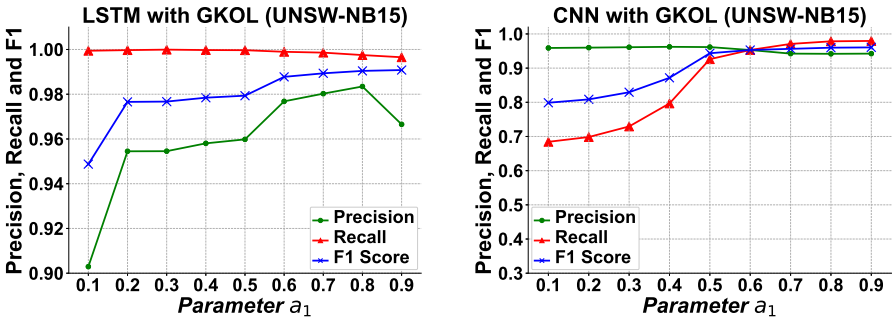
Table 1 details the results of the experiments. According to our experimental results, we believe that the AnoNG framework is successful because it consistently generates highly accurate results in both real-world datasets used to identify anomaly and benign flows. From the results, we can see that AnoNG significantly outperforms baseline methods. We also compare AnoNG's accuracy on the network anomaly detection task against the random forests classifier proposed in [15] and [14].

It is noted that the communication graph features improve the recall increasing from 0.90105 to 0.99757 and the F1-score increasing from 0.93292 to 0.96990 of CNN model in UNSW-NB15 dataset, as well as the precision increasing from 0.95420 to 0.98847 and the F1-score increasing from 0.95719 to 0.98776 of LSTM model in CIC-IDS-2017 dataset.

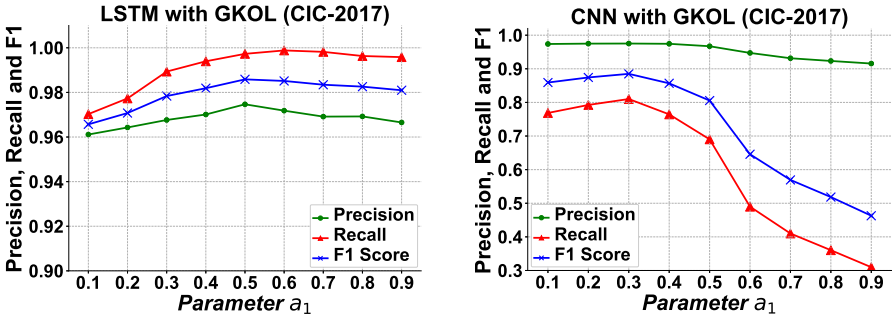
5.4 Performance on DNN with Graph Kernels at Output Level (GKOL)

To understand the necessity of graph kernels, we further measured the performance of the GKOL algorithm. The results for the GKOL algorithm of our validation are equally promising.

After training and obtaining representations of statistical network flow features and graph kernels, for each DNN model, we select trainable weights of a_1 from 0.1 to 0.9. The results of precision, recall and F1-score are shown in Fig. 4, respectively, from which we can observe that GKOL is consistently above baseline methods, and achieves statistically significant improvements on all metrics.



(a) UNSW-NB15 dataset.



(b) CIC-IDS-2017 dataset.

Fig. 4. Precision, recall and F1-score with respect to parameter a_1

Take $a_1 = 0.6$ as an example, LSTM with GKOL outperforms best performance in CIC-IDS-2017 dataset. Therefore, we can draw the conclusion that GKOL maintains a more decent performance in anomaly detection tasks compared with other methods.

5.5 Discussions

AnoNG is designed based on learning the features of network communication graphs. For one thing, the evaluation for this approach verifies its capability to effectively detect existing low-intensity attacks in large-scale networks as its profile involves communication patterns which can be extracted via constructing the communication graphs. For another, graph kernel embedding aims at find small variations on communication patterns between benign and anomalous flows. If the variances between these features are not high, the performance of this technique will decrease, the feature evaluation methods can be further adopted to select the most highly varied features by testing these features and their principal components.

Table 1. Precision, recall and F1-score over the datasets of different methods. The best performance is shown in **BOLD**.

Evaluation Methods	UNSW-NB15			CIC-IDS-2017		
	Precision	***Recall***	***F1-score***	***Precision***	***Recall***	***F1-score***
LSTM Baseline	0.97999	0.99635	0.98810	0.95420	0.96020	0.95719
LSTM+GKIL	0.97576	0.99085	0.98325	**0.98847**	0.98705	**0.98776**
LSTM+GKOL	0.97682	**0.99896**	0.98776	0.97182	**0.99882**	0.98514
CNN Baseline	0.96713	0.90105	0.93292	0.86467	0.96081	0.91021
CNN+GKIL	0.94372	**0.99757**	**0.96990**	**0.97657**	0.95929	**0.96786**
CNN+GKOL	0.95294	0.95305	0.95300	0.94720	0.48977	0.64568
Random Forests [15] [14]	0.999	0.910	0.953	0.98	0.97	0.97

6 Conclusion

In recent years, graph-based or DNN-based network anomaly detection has attracted more and more research attention. In this paper, we propose a novel framework called AnoNG to solve new network anomaly detection tasks by integrating graph kernels with deep neural networks, then apply it to the network anomaly detection problem and investigate different methods of combining deep neural networks with graph kernels. Experiments show that AnoNG outperforms the existing deep neural network models on two real-world network anomaly detection datasets.

Acknowledgement. This research is supported by the National Natural Science Foundation of China (No. 61702508, No. 61802394, No. 61802404) and strategic priority research program of CAS (No. XDC02040100, No. XDC02030200, No. XDC02020200). This research is also supported by Key Laboratory of Network Assessment Technology, Chinese Academy of Sciences and Beijing Key Laboratory of Network Security and Protection Technology. We thank the anonymous reviewers for their insightful comments on the paper.

References

1. Alom, M.Z., Bontupalli, V., Taha, T.M.: Intrusion detection using deep belief networks. In: 2015 National Aerospace and Electronics Conference (NAECON). pp. 339–344. IEEE (2015)
2. Alrawashdeh, K., Purdy, C.: Toward an online anomaly intrusion detection system based on deep learning. In: 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA). pp. 195–200. IEEE (2016)
3. Borgwardt, K.M., Kriegel, H.P.: Shortest-path kernels on graphs. In: Fifth IEEE international conference on data mining (ICDM'05). pp. 8–pp. IEEE (2005)
4. Chawla, A., Lee, B., Fallon, S., Jacob, P.: Host based intrusion detection system with combined cnn/rnn model. In: Proceedings of Second International Workshop on AI in Security (2018)

5. Chiu, C., Zhan, J.: Deep learning for link prediction in dynamic networks using weak estimators. IEEE Access **6**, 35937–35945 (2018)
6. Ionut, A.: New ddos attack method obfuscates source port data. `https://www.securityweek.com/new-ddos-attack-method-obfuscates-source-port-data`, accessed March 10, 2019
7. Kim, G., Yi, H., Lee, J., Paek, Y., Yoon, S.: Lstm-based system-call language modeling and robust ensemble method for designing host-based intrusion detection systems. arXiv preprint arXiv:1611.01726 (2016)
8. Liaskos, C., Kotronis, V., Dimitropoulos, X.: A novel framework for modeling and mitigating distributed link flooding attacks. In: IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications. pp. 1–9. IEEE (2016)
9. Mirkovic, J., Reiher, P.: A taxonomy of ddos attack and ddos defense mechanisms. ACM SIGCOMM Computer Communication Review **34**(2), 39–53 (2004)
10. Moustafa, N., Slay, J.: Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In: 2015 military communications and information systems conference (MilCIS). pp. 1–6. IEEE (2015)
11. Nadeem, M., Marshall, O., Singh, S., Fang, X., Yuan, X.: Semi-supervised deep neural network for network intrusion detection (2016)
12. Navarro, J., Deruyver, A., Parrend, P.: A systematic survey on multi-step attack detection. Computers & Security **76**, 214–249 (2018)
13. Orsini, F., Frasconi, P., De Raedt, L.: Graph invariant kernels. In: Twenty-Fourth International Joint Conference on Artificial Intelligence (2015)
14. Rahul, V.K., Vinayakumar, R., Soman, K., Poornachandran, P.: Evaluating shallow and deep neural networks for network intrusion detection systems in cyber security. In: 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT). pp. 1–6. IEEE (2018)
15. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: ICISSP. pp. 108–116 (2018)
16. Shervashidze, N., Schweitzer, P., Leeuwen, E.J.v., Mehlhorn, K., Borgwardt, K.M.: Weisfeiler-lehman graph kernels. Journal of Machine Learning Research **12**(Sep), 2539–2561 (2011)
17. Yanardag, P., Vishwanathan, S.: Deep graph kernels. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1365–1374. ACM (2015)
18. Yao, Y., Su, L., Lu, Z.: Deepgfl: Deep feature learning via graph for attack detection on flow-based network traffic. In: MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM). pp. 579–584. IEEE (2018)
19. Zargar, S.T., Joshi, J., Tipper, D.: A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks. IEEE communications surveys & tutorials **15**(4), 2046–2069 (2013)
20. Zhang, M., Xu, B., Bai, S., Lu, S., Lin, Z.: A deep learning method to detect web attacks using a specially designed cnn. In: International Conference on Neural Information Processing. pp. 828–836. Springer (2017)