2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering

STDeepGraph: Spatial-Temporal Deep Learning on Communication Graphs for Long-Term Network Attack Detection

Yepeng Yao*†, Liya Su*†, Zhigang Lu*†‡, Baoxu Liu*†
*Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
†School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
{yaoyepeng, suliya, luzhigang, liubaoxu}@iie.ac.cn

***Abstract*—Network communication data are high-dimensional and spatiotemporal, and their information content is often degraded by common traffic analysis methods. For long-term network attack detection based on network flows, it is important to extract a discriminative, high-dimensional intrinsic representation of such flows. This work focuses on a hybrid deep neural network design using a combination of a convolutional neural network (CNN) and long short-term memory (LSTM) with graph similarity measures to learn high-dimensional representations from the network traffic. In particular, examining a set of network flows, we commence by constructing a temporal communication graph and then computing graph kernel matrices. Having obtained the kernel matrices, for each graph, we use the kernel value between graphs and calculate graph characterization vectors by graph signal processing. This vector can be regarded as a kernel-based similarity embedding vector of the graph that integrates structural similarity information and leverages efficient graph kernel using the graph Laplacian matrix. Our approach exploits graph structures as the additional prior information, the graph Laplacian matrix for feature extraction and hybrid deep learning models for long-term information learning on communication graphs.**

Experiments on two real-world network attack datasets show that our approach can extract more discriminative representations, leading to an improved accuracy in a supervised classification task. The experimental results show that our method increases the overall accuracy by approximately 10%-15%.

***Index Terms*—Long-term network attack detection, spatiotemporal deep learning, graph kernel, graph signal processing, dimensionality reduction**

I. Introduction

Modern enterprises are facing the challenge of sophisticated attacks, such as advanced persistent threats (APTs). These long-term network attacks on enterprise networks are one type of fundamental threat that evolves and constitutes the latest attack vectors. Network traffic analysis-based attack detection discovers aberrant behavior caused by attacks and unusual communication patterns. This approach models both spatial and temporal information of network behaviors to describe the essential factors characterizing such behaviors. The approach does not involve time lags between emerging threats and deployed security devices, as is the case for standard defenses that are built upon retrospective detection of suspected attacks. The advantage of network behavior detection is its independence from attack signatures/patterns. This property potentially enables proactive defenses against threats with new and unknown patterns.

With the continuous growth in the number and diversity of Internet hosts and applications, it is becoming more difficult to understand communication patterns of end hosts and network applications for efficient network management and security monitoring. At the same time, network attack detection research uses aggregate fundamental variables of network flows, such as speed, volume, and density, as indicators to measure the current flow of traffic and detect the latent threat [1]. With a large number of features, it has been a major challenge to understand communication patterns precisely.

Several methods have been used to perform dimensionality analysis of network data, e.g., principal component analysis (PCA) and its numerous variants. By modeling network flow data as signals residing on communication graphs, several methods have been proposed to apply the recent graph signal processing for dimensionality reduction [2] and for statistical property representation [3] for network attack flows. In addition, it has been recognized that there are patterns of communication links, statistical dependencies and causal interactions between nodes within a sensor network [4].

In this paper, inspired by the recent advances in graph signal processing and deep learning, we propose a long-term network attack detection framework using high-dimensional representations of the temporal communication graph. Our approach overcomes two major challenges in constructing a topology- and context-sensitive model for network attack detection.

Challenge 1: Large-scale network volume and complex topology. The large scale of the network in the high-dimensional detection space makes it difficult to identify useful features and a good hyperplane for traditional classifiers. Our approach extracts graph kernels from the temporal communication graph and reduces the high-dimensional kernel space to a constant-size low-dimensional detection space by graph signal decomposition.

Challenge 2: Long-term detection of stealthy attacks. Various attacks in real-world networks lead to diverse communication behaviors. Stealthy attacks can often obfuscate their

‡Corresponding author: luzhigang@iie.ac.cn

2324-9013/19/$31.00 ©2019 IEEE
DOI 10.1109/TrustCom/BigDataSE.2019.00025

attack behaviors, exploit the imprecision of normal boundaries generated by traditional detectors and subvert detection. For example, if the statistical feature of the destination port number suddenly increases, it is assumed that some hosts are performing port scanning. However, the activities that may not increase the number of packets are not visible in a large network. Our STDeepGraph method extracts spatial and temporal features from network flows and then performs precise characterization of network behaviors.

We present STDeepGraph – a **S**patial-**T**emporal combined **Deep** learning on communication **Graph** design – to recognize long-term network attack behaviors and perform precise characterization of the benign and attack network traffic flows. We approach the long-term network attack detection problem from the machine learning perspective and use graph similarity feature learning solutions for communication graph analysis. Our contributions are threefold:

- First, we model the network communication structure with suitable temporal communication graphs.
- Second, we propose a new long-term network attack detection framework that integrates deep learning methods with graph kernels; the system is trained end-to-end in a supervised way.
- Third, we perform extensive experiments to demonstrate that our model can extract more discriminative representations of network flow data.

The rest of the paper is structured as follows. Section II formulates the threat model. The STDeepGraph method is detailed in Section III. Section IV reports our experimental evaluation. Section V reviews the related studies. Our conclusions and directions of future research are described in Section VI.

II. Threat Model

We formalize the long-term network attack threat as follows. An attack flow $f_j \in F$ is a timestamped capture recorded at timestamp j, where F denotes the set of all unique flows, and $|F|$ denotes the size of F. A long-term attack trace can be converted into a communication graph of flows ordered by observation time, $g_i = f_1^{(i)}, f_2^{(i)}, ..., f_n^{(i)}$. We define the to-be-detected flow as the target flow, denoted as f_{target}. Each target flow f_{target} is associated with a number of already observed security events denoted as l. The problem is to learn a structural similarity function $f : f_1, f_2, ..., f_l \rightarrow f_{target}$ that accepts a long-term sequence $f_1, f_2, ..., f_l$ and detects the target flow f_{target} for a given network trace. Note that our problem definition is a significant departure from previous approaches that accept only a single flow.

We believe that an attack detection system should be capable of understanding data and producing results, given long-term network flow sequences as the context; hence, our threat model is a better formulation in line with real-world long-term attack scenarios.

III. STDeepGraph Methodology

In this section, we introduce STDeepGraph, which is designed based on the proposed deep neural network over the graph-based similarity feature representation, as shown in Fig 1. The main intuition behind our approach is to map communication topologies in time interval t to graphs, with nodes representing communication endpoints and edges representing flows.

A. Preprocessing of Network Flow Data

a) Temporal Communication Graph Construction: Network communication patterns always demonstrate strong spatial dependency. First, the communication graph needs to be constructed based on the flow records collected from the Internet boundary routers in a specific time interval. Unlike traditional statistical flow features, this step is based on IP addresses and does not require information from packet contents and statistical network flow features. Each IP address is considered an entity and processed as an individual node. Each node is strongly influenced by its neighbors. The main focus is on building a graph to represent communication behavior similarity of IP addresses on one side based on their connectivity with the IP addresses on the other side. We give an example of the temporal communication graph in Fig. 2. The objective of the graph construction approach is to represent structural similarity of a single network flow within the specific time interval based on the related communication pattern. The details of the algorithm are presented in Algorithm 1.

Algorithm 1 Temporal communication graph construction

Input:
The network flows F_n for each specific time interval;
Output:
The temporal communication graph $TCG_n = (V, E)$;

for all $f \in F_n$, extract the source IP address $SrcIP$ and destination IP address $DestIP$;
  if $SrcIP_n \in V$ **then**
    $V := SrcIP_{n+1} \cup V$;
    $E :=< SrcIP_n \rightarrow SrcIP_{n+1} > \cup E$;
    $E :=< DestIP_n \rightarrow SrcIP_{n+1} > \cup E$;
  else if $SrcIP_n \notin V$ **then**
    $V := SrcIP_n \cup V$;
  end if
  if $DestIP_n \in V$ **then**
    $V := DestIP_{n+1} \cup V$;
    $E :=< DestIP_n \rightarrow DestIP_{n+1} > \cup E$;
    $E :=< SrcIP_n \rightarrow DestIP_{n+1} > \cup E$;
  else if $DestIP_n \notin V$ **then**
    $V := DestIP_n \cup V$;
  end if
  if $SrcIP_n \notin V$ **and** $DestIP_n \notin V$ **then**
    $E :=< SrcIP_n \rightarrow DestIP_n > \cup E$;
  end if
end for
return TCG_n;

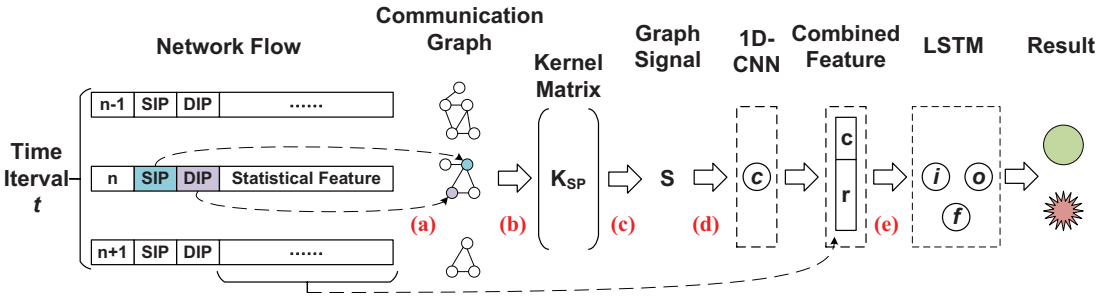


Fig. 1. Schematic representation of the process of a deep neural network on a communication graph. (a) Temporal communication graph construction; (b) Mapping graphs into the graph kernel matrix; (c) Graph signal decomposition; (d) 1D-CNN for learning the probability distribution; (e) LSTM on the combination of flow statistical features and probability distribution sequences for learning temporal dependencies.

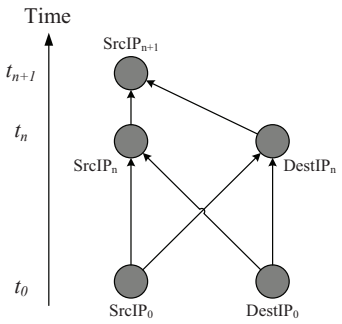


Fig. 2. Toy example of the temporal communication graph.

b) Communication Graph Distance Kernel: After combining all network flows in a time interval into a graph structure, it is natural and reasonable to formulate networks as graphs mathematically. However, even with convolutions on graphs, this approach can only capture the spatial structure approximately due to compromises of data modeling.

A matrix of pairwise flow-to-flow similarities is created from the communication graphs. We select the shortest path similarity measures that capture diverse aspects of the network communication topology and are efficiently computable. The shortest path similarity compares the sorted endpoints and the lengths of the shortest paths that are common between two graphs. Each metric is a function of nodes corresponding to the edge, while $N(n)$ indicates the nodes adjacent to n. While determining all paths is an NP-hard problem, finding special subsets of paths is not necessarily so. Determining the longest paths in a graph is again an NP-hard problem, as it would allow deciding whether a graph contains a Hamilton path or not.

The shortest path $d(i, j)$ represents the distance of the shortest path from i to j, reflecting the expectation that nodes with low degrees of separation are likely to interact. In a communication graph context, the shortest path can also encode the communication of two nodes toward or away from each other over time. The calculation of the shortest path can potentially be demanding, but given the small world property of networks, we can limit the distance searched by the algorithm before stopping and presuming that no path exists.

Using nonlinear feature mapping φ, a graph in a high-dimensional space can be mapped into a low-dimensional feature space so that the graphs become linearly separable in this new vector space [5]. Let a graph set $\mathcal{G}$ contain N graphs; an $N \times N$ kernel matrix $\boldsymbol{K}(\mathcal{G}_i, \mathcal{G}_j)_{N \times N}$ can be calculated using the formula $\boldsymbol{K}(\mathcal{G}_i, \mathcal{G}_j) = \langle \varphi(\mathcal{G}_i), \varphi(\mathcal{G}_j) \rangle$.

Numerous graph kernel functions have been defined to measure the similarity between two graphs in previous research. The shortest-path kernel decomposes a graph into paths [5] and counts the co-occurrence of paths in two graphs. Let $\mathcal{P}_{\mathcal{G}}$ represent the set of all shortest paths in graph $\mathcal{G}$, and let $p \in \mathcal{P}_{\mathcal{G}}$ denote a triplet (s, e, l), where l is the length of the path and s and e are the starting and ending vertices, respectively.

In this paper, we choose the shortest-path kernel to map graphs into a high-dimensional space. The shortest-path kernel matrix between graphs $\mathcal{G}_i$ and $\mathcal{G}_j$ can be defined as:

$$\boldsymbol{K}_{sp}(\mathcal{G}_i, \mathcal{G}_j) = \langle \varphi_{sp}(\mathcal{G}_i), \varphi_{sp}(\mathcal{G}_j) \rangle \tag{1}$$

where the n-th component of $\varphi_{sp}(\mathcal{G}_i)$ contains the frequency of the n-th triplet occurring in graph $\mathcal{G}_i$. Vector $\varphi_{sp}(\mathcal{G}_j)$ is defined analogously for $\mathcal{G}_j$. With the help of the shortest-path kernel, each $\mathcal{G}_i \in \mathcal{G}$ can be converted to vector $\varphi(\mathcal{G}_i)$, and an $N \times N$ kernel matrix $\boldsymbol{K}$ that describes the similarity among the graphs is available.

c) Similarity Matrix Decomposition: Within the graph set, the process of structural similarity feature learning is to convert the graph data into the vector space with the graph kernel function and then reduce its dimension for further analysis. The emerging field of graph signal processing offers a method for applying signal processing approaches to large datasets by representing the signals on graphs [6].

Notation. Before describing the graph structural similarity feature learning method, we introduce some notation. Throughout this part, the following notation is used: the capital non-bold letter X denotes a scalar variable, the lowercase bold letter $\boldsymbol{x}$ represents a vector, and the capital bold letter $\boldsymbol{M}$

denotes a matrix. In this paper, we use the graph kernel as the similarity matrix and adopt the signal graph to convert structured data to vector-type data. Informally, a kernel is a function of two objects that measures their similarity. Mathematically, it corresponds to an inner product in a reproducing kernel Hilbert space.

Dimensionality Reduction. To extract the useful features from the graph similarity matrix, we propose graph Laplacian matrix learning based on graph signal processing for feature dimensionality reduction. Graph signal processing seeks methods to analyze and process graph data. The most effective function in graph signal processing is the graph Laplacian.

Given a graph kernel matrix $\boldsymbol{K}$, the corresponding real-valued and symmetric graph Laplacian matrix is defined as:

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{K} \tag{2}$$

where $\boldsymbol{D}$ is the diagonal matrix with diagonal elements that contain the degree values, defined as:

$$\boldsymbol{D}_{ij} = \begin{cases} \sum_{j=1}^{N} \boldsymbol{K}_{ij}, & i = j, \\ 0, & i \neq j \end{cases} \tag{3}$$

This equation indicates that an input function defined on the vertex domain of a graph can be converted into the corresponding graph in the spectral domain by using the concept of Fourier analysis on graphs.

$$\boldsymbol{L} = \sum_{i=1}^{N} \lambda_i \boldsymbol{u}_i \boldsymbol{u}_i^T \tag{4}$$

where $\lambda_i \in \boldsymbol{\Lambda}$ denotes the set of eigenvalues and $\boldsymbol{u}_i \in \boldsymbol{U}$ denotes the set of orthogonal eigenvectors associated with the Laplacian matrix. Set $\boldsymbol{U}$ constitutes the basis functions for the underlying signal defined on the graph, while $\boldsymbol{\Lambda}$ represents the corresponding frequencies. The graph signal $\boldsymbol{s}_t$ at time t is decomposed into the graph of Fourier domain components denoted by $\hat{\boldsymbol{s}}_t$ at each time interval using the eigenvectors $\boldsymbol{u}_{it}$ of the corresponding Laplacian matrix at $\boldsymbol{L}_t$, as given by [7]:

$$\hat{\boldsymbol{s}}_t = \boldsymbol{U}_t^T \boldsymbol{s}_t \tag{5}$$

where $\boldsymbol{U}_t$ is a matrix composed of the eigenvectors of $\boldsymbol{L}_t$. A graph signal $\boldsymbol{s}_t$ is regarded as smooth with respect to the graph kernel matrix if most of its energy is concentrated in the low frequencies [8]. In other words, a smooth signal $\boldsymbol{s}_t$ gives rise to a smaller value of graph smoothness regularizer $\boldsymbol{s}_t^T \boldsymbol{L}_t \boldsymbol{s}_t$ defined by:

$$\boldsymbol{s}_t^T \boldsymbol{L}_t \boldsymbol{s}_t = \sum_{i=1}^{N} \lambda_i (\boldsymbol{u}_{it}^T \boldsymbol{s}_t)^2 \tag{6}$$

The graph Laplacian matrix plays an important role in describing the underlying structure of the graph signal. The graph signals are modeled by a graph kernel matrix that encodes the similarity between communication graphs. The graph kernel matrix measures N communication graphs, where N is the dimensionality of a graph signal. Individual low-dimensional representations of the graph signals are subject to classification independently. Low-dimensional representations could reduce the deep learning complexity in space and time. Within the graph set, the objective of similarity feature extraction is to convert the graph data into the $N \times N$ vector space with the graph kernel function and then reduce its dimensions for further detection. The emerging field of graph signal processing offers a method for applying signal processing approaches to large datasets by representing the signals on graphs [6].

B. Long-Term Network Attack Detection

a) CNN for Spatial Feature Extraction: Accordingly, to extract spatial patterns and features in the graph structure similarity domain, the graph signal sequence $\boldsymbol{s}_t$ converted from the graph similarity matrix is used directly in the CNN model. A convolution operation involves a filter $\boldsymbol{w}$ that is applied to a window of graph signals to produce a new feature map $\boldsymbol{c}_t$, defined as:

$$\boldsymbol{c}_t = \phi(\boldsymbol{w} \cdot \boldsymbol{s}_t + \boldsymbol{b}) \tag{7}$$

where ϕ is the nonlinear rectifier function, and $\boldsymbol{b}$ is the bias. After convolution, pooling is applied to the sequence. Then, a maximum-over-time pooling operation is applied over the feature map, and the maximum value $\hat{\boldsymbol{c}} = max\{\boldsymbol{c}\}$ is taken as the next feature, which can be mapped to a one-hot vector to obtain the probabilities of benign communication and attacks.

1D Convolution Layer. To process a given input graph signal vector, many filters are convolved with the normalized representations of the patches contained in the graph.

Pooling Layer. We then apply a max-pooling operation over the feature map, thus retaining only the maximum value of $\boldsymbol{c}$, $max(\boldsymbol{c}_1, \boldsymbol{c}_2, ..., \boldsymbol{c}_{Pmax})$ as the signal associated with w.

We use three 1D convolution layers and two pooling layers to extract spatial features. These features form the penultimate layer and are passed to a fully connected softmax layer, the output of which is the probability distribution of the graph signal sequence. It should be noted that both the convolution filter and pooling layers are one-dimensional operations, which is the key characteristic of 1D-CNN.

b) LSTM for Temporal Features Extraction: We adopt a modified LSTM recurrent neural network that learns the dynamic temporal dependencies present in network flow. In this model, the gate structure in the traditional LSTM and the hidden state are unchanged, but the input is replaced by the combined statistical network flow features and spatial features produced by the 1D-CNN in Section III-B, which are reshaped into a vector. The network still consists of five subunits: input gate $\boldsymbol{i}$, forget gate $\boldsymbol{f}$, output gate $\boldsymbol{o}$, the candidate hidden representation $\boldsymbol{g}$ and the internal memory of the LSTM cell $\boldsymbol{c}$. They are computed by:

$$\begin{aligned} \boldsymbol{i}_t &= \sigma(\boldsymbol{w}_i \cdot \langle \boldsymbol{r}_t, \boldsymbol{c}_t \rangle + \boldsymbol{u}_i \cdot \boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \\ \boldsymbol{f}_t &= \sigma(\boldsymbol{w}_f \cdot \langle \boldsymbol{r}_t, \boldsymbol{c}_t \rangle + \boldsymbol{u}_f \cdot \boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \\ \boldsymbol{o}_t &= \sigma(\boldsymbol{w}_o \cdot \langle \boldsymbol{r}_t, \boldsymbol{c}_t \rangle + \boldsymbol{u}_o \cdot \boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \\ \boldsymbol{g}_t &= tanh(\boldsymbol{w}_g \cdot \langle \boldsymbol{r}_t, \boldsymbol{c}_t \rangle + \boldsymbol{u}_g \cdot \boldsymbol{h}_{t-1} + \boldsymbol{b}_g) \\ \boldsymbol{c}_t &= \boldsymbol{c}_{t-1} \odot \boldsymbol{f}_t + \boldsymbol{g}_t \odot \boldsymbol{i}_t \end{aligned} \tag{8}$$

where $\boldsymbol{w}$ and $\boldsymbol{u}$ are learnable weight matrices and $\boldsymbol{b}$ is a learnable bias vector.

A final softmax layer is added to complete the architecture. The LSTM model is repeatedly trained using the combined statistical network flow features and spatial features, and then, they are classified into benign flow and attack flow. Our proposal of this new deep neural network architecture is motivated by better performance of graph signal processing for graph similarity feature extraction, as shown in the evaluation in Section IV.

IV. Experimental Evaluation

In this section, we conduct a set of experiments on two real-world network attack datasets UNSW-NB15 [9] and CIC-IDS-2017 [10], described in Table I.

A. Dataset Description

UNSW-NB15 is a public dataset created by establishing a synthetic environment at the UNSW cybersecurity lab. The data represent a hybrid of contemporary real-world normal and synthesized attack network traffic. The data have 47 features, including new long-term attack scenarios and 9 significant families of network attacks.

CIC-IDS-2017 is a newer public dataset that contains a more recent form of stealthy attacks including low-frequency and long-term network attacks, such as DDoS, DoSGoldenEye, DoSHulk, and DoSSlowhttptest, generated by botnets and tools. This dataset is provided by the Canadian Institute for Cybersecurity (CIC).

Both datasets are newer attack detection datasets; we divide them into training and test data sets using the ratio of 60% to 40%, respectively. It is important to note that training datasets and test datasets originate from different collections so that the general prediction capability of STDeepGraph can be tested on data that are not part of the training data.

TABLE I
Description of the Two Datasets

Name	# Dimensions	# Instances	Attack Ratio (ρ)
UNSW-NB15	47	2,540,044	ρ = 0.145
CIC-IDS-2017	82	2,830,743	ρ = 0.167

B. Experimental Setup

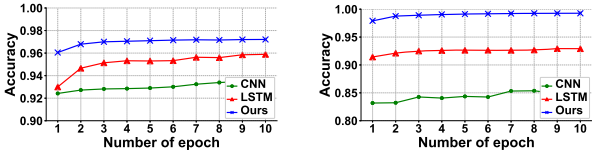
We implemented a proof-of-concept of our proposed framework using the software frameworks Keras, NetworkX and Scikit-Learn, which were run on Ubuntu 16.04 64-bit OS.

For convenience, we set CNN network dimensions to $s/128/256/128/c$ for both datasets, where s is the graph similarity dimension, which depends on the preset time interval t. The LSTM network dimensions are set to $(r + c)/128/256/128/2$, where c is taken from the output of the CNN, and r is the statistical feature dimension that varies between datasets. All layers are densely (fully) connected. To train the deep learning methods, we set the batch size to 128 and the learning rate to 0.001. The number of epochs of training is 10, and the dropout rate is set to 0.5. The 'Adam' optimizer is used for a variant of gradient descent. The time interval t is $60s$.

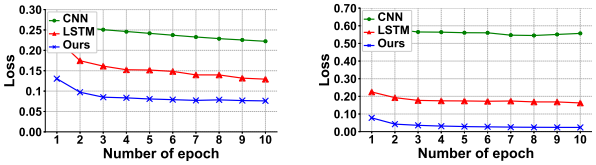
The learning performance of the model will be measured using the evaluation matrix of the classifier. For evaluation, we report the *accuracy (ACC)*, *precision (PR)*, *detection rate (DR, namely, recall)* and *the false alarm rate (FAR)*.

C. How Does the CNN-LSTM Combination Model Help?

To quantitatively evaluate our approach, we report the comparison to baseline methods for detecting performance characteristics. Table II lists the results of experiments. The results show that our method significantly outperforms baseline methods. The proposed method achieves the best performance on the UNSW-NB15 and CIC-IDS-2017 datasets, with 98.6% and 99.4% accuracy on the test set, respectively. It is also noted that our method increases the accuracy by approximately 10%-15% and decreases the FAR below 10%; additionally, the precision increases by approximately 2%-13% for both datasets.



(a) Test accuracy curves.



(b) Test loss curves.

Fig. 3. Deep neural network testing process for the two datasets. Results for UNSW-NB15 are on the left, and those for CIC-IDS-2017 are on the right.

Fig. 3 shows the accuracy and loss curves versus the number of epochs during testing. Note that we show the accuracy curves of the original CNN, the original LSTM and our proposed method. Our proposed method clearly has a better learning curve.

Based on the results and analysis above, we can conclude that the proposed model can achieve an outstanding performance.

D. How Effective is STDeepGraph Compared with Traditional Machine Learning Techniques?

We evaluate the effectiveness of STDeepGraph on both datasets and compare our results on the CIC-IDS-2017 dataset with those of several state-of-the-art machine learning detection models described in [10]. In terms of precision and

TABLE II
EVALUATION OF THE OVERALL PERFORMANCE. THE BEST PERFORMANCE FOR EACH DATASET IS SHOWN IN **BOLD**.

Dataset Name		Only CNN				Only LSTM				Our Method (STDeepGraph)			
		ACC	*PR*	*DR*	*FAR*	*ACC*	*PR*	*DR*	*FAR*	*ACC*	*PR*	*DR*	*FAR*
UNSW-NB15	Benign	0.887	0.535	0.788	0.212	0.976	0.979	0.994	0.149	**0.986**	**0.988**	**0.996**	**0.083**
	Attack		0.535	0.788	0.099		0.956	0.851	0.006		**0.967**	**0.917**	**0.004**
CIC-IDS-2017	Benign	0.842	0.865	0.961	0.750	0.928	0.954	0.960	0.230	**0.994**	**0.995**	**0.998**	**0.025**
	Attack		0.561	0.250	0.039		0.795	0.770	0.039		**0.991**	**0.975**	**0.002**

detection rate, we observe that our proposed STDeepGraph method can outperform the alternative detection models.

To illustrate the performance advantage of our model, we compare it with our reproduction of the random forests model illustrated in [10] using statistical features for detection on the two datasets. For the random forests method and the STDeepGraph method, we train models on the respective sets of hyperparameters and choose the model that has the best performance.

Compared to the random forest method, our STDeepGraph method shows an excellent capability of dealing with long-term attacks. The ACC, PR, DR and FAR metrics of both models on the test dataset are shown in Table II. Our proposed STDeepGraph method achieves higher average accuracy, higher average detection rates and much lower average false alarm rates on the two datasets than does the random forests model. For the UNSW-NB15 and CIC-IDS-2017 datasets, the respective accuracy values are approximately 0.986 and 0.994; the detection rates are approximately 0.956 and 0.986; and the false alarm rates are approximately 0.044 and 0.013.

E. Discussion

Based on our evaluation results, we believe that the proposed approach is successful because it produced highly accurate results consistently for both datasets used in terms of detecting attack and benign network flows and correctly detecting long-term attack scenarios such as DosSSlowhttptest, DoSslowloris, and DoSHulk.

On the one hand, STDeepGraph is designed based on identifying the flow-to-flow similarity of network communication graphs. The evaluation of this approach verifies its capability to effectively detect existing and stealthy attacks in large-scale networks, as its profile involves communication patterns that can be extracted by constructing the communication graph. This approach also does not require any prior information about attack observations, which shows the efficiency of its application in online systems without much effort required in the training phrase.

On the other hand, in current networks, some types of attacks such as spyware and stealthy attacks attempt mimicking benign activities. The graph similarity representation aims to find small variations in communication patterns between benign and attack flows; however, the communication patterns of current sophisticated attacks such as SSH-Patator and FTP-Patator are sometimes similar to those of benign flows.

V. RELATED WORK

Much work has been done on analyzing network traffic of networked systems. A popular research direction is to adopt deep learning-based and graph-based approaches. Generally, these approaches aim to model different classes of activity (i.e., benign/attack) based on some algorithm that is trained on real data.

A. Network Flow-Based Attack Detection

Network attack detection systems have been proposed for mitigating attacks. However, due to high speeds and large size of current networks, this methodology still faces the challenge of building a scalable, adaptable, and effective attack detection system.

Network traffic analysis approaches to establishing such a system have been recommended in many studies. For instance, Nasr et al. [11] proposed a compressive traffic analysis approach to scalable traffic analysis and then applied such compressive traffic analysis to two widely studied classes of traffic analysis, namely, flow correlation and website fingerprinting. Experiments showed that the state-of-the-art flow correlation and website fingerprinting schemes outperformed their traditional alternatives. Yao et al. [12] proposed methods of deep graph feature extraction to distinguish network attacks in network flows.

B. Graph Kernel

Several surveys have explored graph kernel studies [13]. In this research, we focus on the utilization of graph kernels in the design of graph comparison.

In general, defining the features of the complicated structure of graphs is a difficult problem for researchers. However, the kernel-based machine learning framework provides an alternative systematic approach to this problem. Kernel-based methods do not rely on feature generation required by traditional feature-based methods. The graph kernel uses a kernel function defining the similarity between two graphs. The kernel function maps the input graph space into the feature space.

A kernel-based method also needs a kernel machine, which is an algorithm that needs only kernel function values to learn patterns of the input graphs in the feature space. The performance of kernel-based methods depends significantly on the selection and design of kernel functions for different applications. [14]

In graph-structured data, graph kernels, including the Laplacian kernel [15], diffusion kernels [16] and the marginalized

TABLE III
EVALUATION OF EFFECTIVENESS IN COMPARISON WITH RANDOM FORESTS. THE BEST PERFORMANCE FOR EACH DATASET IS SHOWN IN **BOLD**.

Dataset Name	Random Forests Method in [10]				Our Method (STDeepGraph)			
	ACC	*PR*	*DR*	*FAR*	*ACC*	*PR*	*DR*	*FAR*
UNSW-NB15	0.935	0.860	0.836	0.164	**0.986**	**0.978**	**0.956**	**0.044**
CIC-IDS-2017	0.982	0.986	0.947	0.053	**0.994**	**0.993**	**0.986**	**0.013**

kernel [17], are designed to capture features of graphs. They have been widely used in behavior learning on communication networks or network flows. Graph kernels have also been applied to graph classification (to classify communication patterns or malicious behavior). In these models, graphs are decomposed into their substructures such as nodes, links, random walk paths [18], limited-sized subgraphs [19], and shortest paths [20]. A recently proposed effective class of graph kernels is Weisfeiler-Lehman (WL) kernels [21]. This approach involves subsequent aggregation of substructure similarities to represent graph similarity.

Several methods have been developed to perform graph embedding for the purpose of dimensionality reduction of high-dimensional data. Yan et al. [22] surveyed a list of such methods, including the principal component analysis, the linear discriminant analysis, multidimensional scaling, locality-preserving properties and kernel eigenmaps.

C. Hybrid Deep Neural Network

Our proposal in Section III was inspired by the hybrid deep neural network method.

Deep learning has achieved breakthroughs in image and video analysis. In particular, convolutional neural networks (CNNs) for image and video recognition and recurrent neural networks (RNNs) for speech and natural language processing (NLP) often deliver unprecedented levels of performance [23]. CNNs and RNNs extract data-driven features from input data (e.g., image, video, and audio data) structured in typically low-dimensional regular grids. Such grid structures are often assumed to have statistical characteristics to facilitate the modeling process.

The combination of CNN and LSTM in a unified framework has already achieved state-of-the-art results in many research areas. To combine and fully make use of spatial features, some researchers use CNNs to capture adjacent relations in the communication network, along with using RNN over the time dimension. However, the application of the normal convolutional operation restricts the model to processing only grid structures (e.g., images and videos) rather than being applicable to general domains. Additionally, recurrent networks for sequence learning require iterative training, which introduces error accumulation as the number of steps increases. Some studies have explored the hybrid approach. Wang et al. combined LSTM and CNN for a hierarchical spatial-temporal features-based attack detection system that first learned the low-level spatial features of network traffic using CNN and then learned high-level temporal features using an LSTM network. Yuan et al. [24] proposed a convolutional LSTM network to identify insiders' anomalous behavior.

VI. CONCLUSIONS

In this paper, we proposed a novel method called STDeepGraph for enhancing long-term network attack detection based on the similarity feature of the communication graph and hybrid deep learning, which involves extraction by graph kernels and decomposition by graph signal processing.

It has been shown that the proposed long-term network attack detection method performs significantly better than the other baseline methods, as evidenced by the higher accuracy and detection rate values. Moreover, our results show that network activity can be described by a temporal communication graph model, and this model can be highly robust and therefore capable of reducing false alarm rates. We show that STDeepGraph provides a good use case for the general strategy of combining the strengths of graph embedding and hybrid deep learning approach, which we believe will become common in many other network analysis tasks.

ACKNOWLEDGMENTS

This work was supported by the National Key R&D Program of China (No. 2018YFB0803600, No. 2017YFC0821804-2, No. 2018YFB0803602, No. 2016QY06X1204) and the Strategic Priority Research Program of the Chinese Academy of Sciences (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 many people for their helpful comments on drafts of this paper and the anonymous reviewers for their feedback on the paper.

REFERENCES

[1] Houping Xiao, Jing Gao, Deepak S. Turaga, Long H. Vu, and Alain Biem, "Temporal multi-view inconsistency detection for network traffic analysis," in *Proceedings of the 24th International Conference on World Wide Web*, New York, NY, USA, 2015, WWW '15 Companion, pp. 455–465, ACM.

[2] P. Chen, S. Choudhury, and A. O. Hero, "Multi-centrality graph spectral decompositions and their application to cyber intrusion detection," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2016, pp. 4553–4557.

[3] H. Sadreazami, A. Mohammadi, A. Asif, and K. N. Plataniotis, "Distributed-graph-based statistical approach for intrusion detection in cyber-physical systems," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 4, no. 1, pp. 137–147, March 2018.

[4] H. E. Egilmez and A. Ortega, "Spectral anomaly detection using graph-based filtering for wireless sensor networks," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 1085–1089.

[5] K. M. Borgwardt and H. P. Kriegel, "Shortest-path kernels on graphs," in *IEEE International Conference on Data Mining*, Nov 2005, pp. 74–81.

[6] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, May 2013.

[7] Fan RK Chung and Fan Chung Graham, *Spectral graph theory*, Number 92. American Mathematical Soc., 1997.

[8] Aliaksei Sandryhaila and Jose MF Moura, "Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 80–90, 2014.

[9] N. Moustafa and J. Slay, "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)*, Nov 2015, pp. 1–6.

[10] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization.," in *ICISSP*, 2018, pp. 108–116.

[11] Milad Nasr, Amir Houmansadr, and Arya Mazumdar, "Compressive traffic analysis: A new paradigm for scalable traffic analysis," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 2053–2069.

[12] Yepeng Yao, Liya Su, and Zhigang Lu, "DeepGFL: Deep feature learning via graph for attack detection on flow-based network traffic," in *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*. IEEE, 2018, pp. 579–584.

[13] Palash Goyal and Emilio Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowledge-Based Systems*, vol. 151, pp. 78–94, 2018.

[14] Heiko Paulheim, "Knowledge graph refinement: A survey of approaches and evaluation methods," *Semantic web*, vol. 8, no. 3, pp. 489–508, 2017.

[15] Yasutoshi Yajima, "One-class support vector machines for recommendation tasks," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2006, pp. 230–239.

[16] Risi Imre Kondor and John Lafferty, "Diffusion kernels on graphs and other discrete structures," in *Proceedings of the 19th international conference on machine learning*, 2002, vol. 2002, pp. 315–322.

[17] Francois Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens, "Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation," *IEEE Transactions on knowledge and data engineering*, vol. 19, no. 3, pp. 355–369, 2007.

[18] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt, "Graph kernels," *Journal of Machine Learning Research*, vol. 11, no. Apr, pp. 1201–1242, 2010.

[19] Tamás Horváth, Thomas Gärtner, and Stefan Wrobel, "Cyclic pattern kernels for predictive graph mining," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2004, pp. 158–167.

[20] Karsten M Borgwardt and Hans-Peter Kriegel, "Shortest-path kernels on graphs," in *Data Mining, Fifth IEEE International Conference on*. IEEE, 2005, pp. 8–pp.

[21] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt, "Weisfeiler-lehman graph kernels," *Journal of Machine Learning Research*, vol. 12, no. Sep, pp. 2539–2561, 2011.

[22] Shuicheng Yan, Dong Xu, Benyu Zhang, Hong-Jiang Zhang, Qiang Yang, and Stephen Lin, "Graph embedding and extensions: A general framework for dimensionality reduction," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 1, pp. 40–51, 2007.

[23] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436, 2015.

[24] Fangfang Yuan, Yanan Cao, Yanmin Shang, Yanbing Liu, Jianlong Tan, and Binxing Fang, "Insider threat detection with deep neural network," in *International Conference on Computational Science*. Springer, 2018, pp. 43–54.