IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON ARTIFICIAL INTELLIGENCE IN CYBERSECURITY

Received August 2, 2019, accepted September 11, 2019, date of publication September 23, 2019, date of current version October 9, 2019.

*Digital Object Identifier 10.1109/ACCESS.2019.2943249*

# An Empirical Evaluation of Deep Learning for Network Anomaly Detection

**RITESH K. MALAIYA[1], DONGHWOON KWON[2], SANG C. SUH[1], HYUNJOO KIM[3], IKKYUN KIM[3], AND JINOH KIM[1], (Senior Member, IEEE)**

[1]Computer Science Department, Texas A&M University-Commerce, Commerce, TX 75428, USA
[2]Department of Math, CSCI, and Physics, Rockford University, Rockford, IL 61108, USA
[3]Information Security Research Division, ETRI, Daejeon 34129, South Korea

Corresponding author: Jinoh Kim (jinoh.kim@tamuc.edu)

This work was supported in part by the Institute for Information and Communications Technology Promotion (IITP) Grant funded by the Korean Government (MSIP) under Grant 2016-0-00078 (Project: Cloud-Based Security Intelligence Technology Development for the Customized Security Service Provisioning).

**ABSTRACT** Deep learning has been widely studied in many technical domains such as image analysis and speech recognition, with its benefits that effectively deal with complex and high-dimensional data. Our preliminary experiments show a high degree of non-linearity from the network connection data, which explains why it is hard to improve the performance of identifying network anomalies by using conventional learning methods (e.g., Adaboosting, SVM, and Random Forest). In this study, we design and examine deep learning models constructed based on Fully Connected Networks (FCNs), Variational AutoEncoder (VAE), and Sequence-to-Sequence (Seq2Seq) structures. For the extensive evaluation, we employ a broad range of the public datasets with unique characteristics. Our experimental results confirm the feasibility of deep learning-based network anomaly detection, with the improved performance compared to the conventional learning techniques. In particular, the detection model based on Seq2Seq with LSTM is highly promising, consistently yielding over 99% of accuracy to identify network anomalies from the entire datasets employed in the evaluation.

**INDEX TERMS** Network anomaly detection, traffic analysis, deep learning, neural networks, sequence-to-sequence, performance evaluation.

## I. INTRODUCTION

Keeping the network secure from cyber-attacks has long been an important concern in computer networks. However, we see a growing number of malware and cyber-attacks year by year. In addition, new emerging attacks are much more critical than ever showing greater impacts. For instance, a ransomware attack (''WannaCry'') encrypted files in the victim machines across 10,000 organizations causing a significant financial loss in 2017 [1]. Another critical incident reported in 2016 was by DDoS attacks which hit large datacenters, and Twitter and Spotify temporarily closed down the sites due to the attack [2]. Cyber-attacks could be more severe with the increasing use of mobile and Internet of Things (IoT) devices. An incident reported that hundreds of thousands of compromised IoT devices were employed to make a DDoS attack [3].

The associate editor coordinating the review of this manuscript and approving it for publication was Chi-Yuan Chen.

Anomaly detection is one of the crucial functions in network security, which identifies malicious and unexpected activities from the network traffic. With the remarkable advances in the past several years, machine learning has been widely studied to identify anomalies in the network [4]–[9]. However, our preliminary experiments showed that conventional machine learning is largely limited with sometimes unacceptable accuracy in detection when analyzing network data which are often complex and high-dimensional. For example, we observed no better than 83% accuracy with several machine learning techniques such as Adaboosting, Support Vector Machine (SVM), and Random Forest (RF), against the NSL-KDD dataset [10], which is widely used for evaluating network anomaly detection functions [5], [6], [11]–[16]. From the experiment against another public dataset of UNSW-NB15 [17], the result shows less than 91% of detection accuracy with the above learning methods.

The root reason of the limitation of conventional learning is that such techniques are not effective to deal with the high

complex data. In fact, we observed a significant degree of non-linearity from public network data from our preliminary experiments, which motivates us to explore deep learning structures for network anomaly detection. Deep learning is known to be powerful to deal with complex data with the property of non-linearity [18], [19]. While deep learning has been applied for a variety of applications such as image processing and natural language processing [20], [21], it has not been thoroughly examined for the application of network anomaly detection. Our past survey work [22] also shows that using deep learning for network data analysis is still in the initial stage and the evaluation studies were limited with only a few datasets such as NSL-KDD.

In this paper, we design and evaluate a set of deep learning models established based on Fully Connected Network (FCN) [23], Variational AutoEncoder (VAE) [24], and Sequence-to-Sequence (Seq2Seq) [25]–[27] structures. For thorough evaluation, we employ a variety of datasets with different characteristics: NSL-KDD [10], Kyoto-Honeypot [28], UNSW-NB15 [17], IDS2017 [29], and MAWILab traces [30]. Our experimental results with the aforementioned traces confirm the potential of the evaluated deep learning models for network anomaly detection, with the significantly improved performance compared to conventional shallow learning techniques. In particular, the detection model based on the Seq2Seq structure with LSTM cells shows the highly promising performance, yielding over 99% of detection accuracy against the datasets employed.

The main contributions of this paper are summarized as follows:

- We demonstrate the complexity of the network traffic datasets, explaining why conventional shallow learning would not be adequate for identifying network anomalies.
- We present the deep learning models designed for network anomaly detection based on FCN, VAE, and Seq2Seq structures, with the basic concepts of the learning structures.
- We conduct extensive experiments for the thorough evaluation of the deep learning models with the various datasets (NSL-KDD, Kyoto-Honeypot, UNSW-NB15, IDS 2017, and MAWILab), and report our observations with the implications of the experimental results.

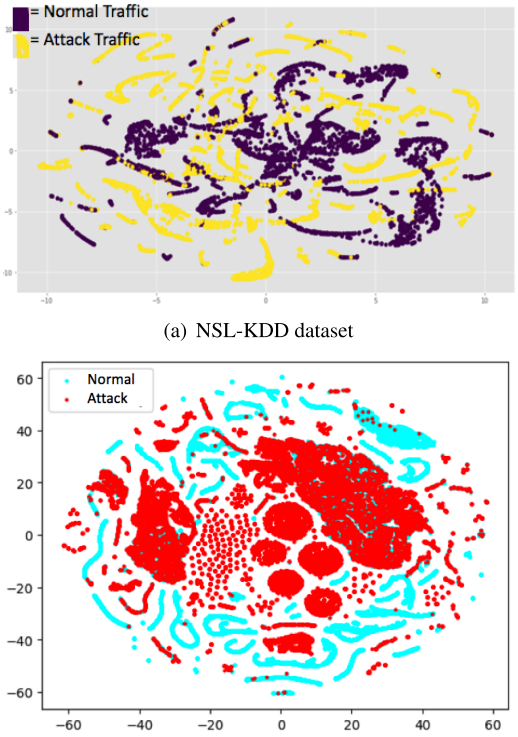
The organization of this paper is as follows. We first introduce a summary of the past studies and the description of the public datasets, along with the discussion of the motivation of this work in Section II. In Section III, we present the design of deep learning models based on the FCN, VAE, and Seq2Seq structures. We next evaluate the deep learning models with the extensive set of public network data in Section IV. Finally, we conclude our presentation in Section V.

## II. BACKGROUND

In this section, we discuss the complexity of network traffic data that may eliminate shallow learning from the consideration for the effective detection of anomalies. We then summarize the closely related studies and describe the datasets employed in the evaluation.

**TABLE 1. Accuracy of conventional ML techniques (dataset: NSL-KDD).**

| Training | Testing | Adaboosting | SVM | RF |
|---|---|---|---|---|
| Train- | Test+ | 82.5% | 79.6% | 78.3% |
| Train- | Test- | 65.5% | 56.5% | 53.4% |
| Train+ | Test+ | 80.5% | 79.1% | 76.1% |
| Train+ | Test- | 58.7% | 56.4% | 50.3% |




(a) NSL-KDD dataset

(b) UNSW-NB15 dataset

**FIGURE 1. Visualization using t-SNE for NSL-KDD and UNSW-NB15: The distributions show the mixture of normal and attack data points sharing the same feature space.**

### A. WHY DEEP LEARNING FOR NETWORK ANOMALY DETECTION?

In the preliminary experiment, we examined conventional learning techniques including Adaboosting, SVM, and RF to see how well they are working to identify network anomalies. Table 1 shows the detection accuracy using the conventional learning methods against the NSL-KDD dataset. For this experiment, the default setting was used without fine-tuning. The result shows less than 83% of accuracy, which may not be acceptable in practice. Note that the description of the datasets can be found in Section II-C.

To see why, we analyzed the distribution of the data points using *t*-Distributed Stochastic Neighbor Embedding (*t*-SNE), a tool to effectively visualize high-dimensional data. Figure 1 demonstrates the distributions of normal and attack data points for different datasets of NSL-KDD and

UNSW-NB15. The *t*-SNE plots in the figure show that the two different classes of data (normal and attack) share the same feature space. This indicates that discriminating attacks from the normal data points would be significantly challenging.

Deep learning is powerful enough to deal with the data with non-linearity [19], which motivates us to examine deep learning methods for network anomaly detection. In this work, we will present the construction of deep learning models to improve the detection performance with an extensive evaluation with a diverse set of network traffic data.

### B. RELATED WORK

Our past survey work [22] investigated the use of deep learning for network anomaly detection, and the conclusion was that using deep learning in network traffic analysis is still in the initial stage and the evaluation studies were limited with only a few datasets such as NSL-KDD. In this section, we briefly introduce the closely related studies employed deep learning techniques for network anomaly detection.

The work in [11] utilized a Deep Neural Network (DNN) for flow-based anomaly detection in a Software Defined Networking (SDN) environment. Note that the main concept of SDN is to decompose the architecture into the network control and forwarding functions. By doing so, the network control and underlying infrastructure can be programmable and abstracted, respectively for applications and network services. In the proposed SDN security architecture, the intrusion detection module is implemented in the SDN controller to monitor Open-Flow switches. The designed model employs a simple deep neural network consisting of one input layer, three hidden layers, and one output layer. The evaluation was conducted using the NSL-KDD dataset. The authors reported 75% of classification accuracy as the best performance against the NSL-KDD dataset. One reason for the poor performance would be from the use of a small subset of the features (six out of 41 features) for training and testing. Hyper parameters used in the experiments are: number of inputs $=$ 6 features, number of hidden nodes $=$ {3, 6, 12}, number of outputs $=$ 2, batch size $=$ 10, learning rate $=$ {1$e$-01, 1e-02, 1e-03, 1e-04}, and epochs $=$ 100.

The authors in [15] combined spectral clustering and neural networks (DNN) for multi-classification for intrusion detection. In the training phase, the clustering groups the given dataset and the data points for each group are used to train the associated DNN. Similarly, the clustering is applied to the testing dataset and then the data points in each group are fed into the DNN for that group. Finally, the output of the DNNs is aggregated to make the final decision. Using the KDDCup 1999 data with five classes (Normal, DOS, Probe, U2R, and R2L), the authors show that the proposed technique works slightly better than the conventional learning methods including SVM and random forest. However, the reported accuracy is not acceptable showing lower than 80% accuracy for three classes (Probe, U2R, and R2L).

In [14], the authors set up an intrusion detection model based on Convolutional Neural Networks (CNNs). Since CNN models often take two- or three-dimensional data as input (e.g., images), the authors converted the NSL-KDD connection record into a $8 \times 8$ gray-scale pixel image using the one-hot encoding and binning technique. Then the authors simply employed the existing CNN models, ResNet 50 and GoogLeNet, for learning and testing. The observed accuracy against NSL-KDD is not that promising showing less than 82% of accuracy.

In addition, another work in [31] introduced a deep learning approach based on a simple Recurrent Neural Networks (RNN) with forward and back propagations, for network intrusion detection. The authors reported F-measure of 68%–99% as the binary classification accuracy against the NSL-KDD datasets. A recent study [13] set up a deep learning model using Restricted Boltzmann Machines (RBMs) for network anomaly detection. The authors evaluated their deep learning model using NSL-KDD and reported 91% of accuracy. Additionally, the RBM-based model has also been evaluated with the UNSW-NB15 dataset [17] containing 49 features with the class label, and the authors showed their technique works slightly better than shallow learning techniques (proposed $=$ 95.8% and random forest $=$ 94.4% of accuracy).

Additionally, the authors in [32] examined the Restricted Boltzmann Machine (RBM) with both Contrastive Divergence (CD) and Persistent Contrastive Divergence (PCD) algorithms for the purpose of network anomaly detection. This study evaluated the performance with the IDS2012 dataset collected in 2010 [33], which is a predecessor of the IDS2017 dataset employed in our evaluation. The reported accuracy is ranged between 88–89%. The work in [34] presented an ensemble method by adopting multiple learners using a variety of algorithms including a Self-Organization Map (SOM) Artificial Neural Network (ANN). The evaluation was conducted only with the NSL-KDD dataset, and ANN does not show any better performance than the other shallow learning techniques employed.

As summarized, there were several past studies that utilized deep learning techniques for network anomaly detection. However, the past work simply introduced a single deep learning model without extensive comparison. In this work, we set up a set of deep learning models based on Fully Connected Network (FCN) [23], Variational AutoEncoder (VAE) [24], and Sequence-to-Sequence (Seq2Seq) [25]–[27] structures. In addition, the previous studies largely relied on the KDDCup 1999 and its variant (NSL-KDD) datasets only, which would raise a question that their techniques can work well in different environments. For this purpose, we employ five different datasets with unique characteristics to make thorough evaluation.

### C. DESCRIPTION OF DATASETS

This section provides a brief description of the datasets used in this research.

#### 1) KDD CUP 1999 DATASET [35] AND NSL-KDD [10]

The KDD Cup 1999 dataset was collected from a simulated military network environment over nine weeks. It contains various intrusions including R2L (unauthorized access from a remote machine), U2R (unauthorized access to local root privileges), probing (surveillance and other probing), and DOS (denial-of-service). Each data point in the dataset includes the connection information with 41 features plus the associated label, which can be used to distinguish attack connections from the normal traffic.

NSL-KDD is a refined version of the KDD Cup 1999 data that filters out a number of duplicated records to reduce the bias in the classification. NSL-KDD contains two files for training (''Train+'' and ''Train-'') and two other files for testing (''Test+'' and ''Test-''). Note that Train- and Test- are subsets of Train+ and Test+, respectively, to give different degrees of difficulty in classification. The size of data files is as follows: Train+ (125,973 records), Test+ (22,544 records), Train- (25,192 records), and Test- (11,850 records). The distributions of normal and anomalous connections are fairly well balanced in the range of 46% – 82% in the dataset.

#### 2) KYOTO UNIVERSITY HONEYPOT DATASET (KYOTO-HONEYPOT) [28]

Kyoto Honeypot is a daily-based evaluation dataset since 2009 for network anomaly detection. This dataset includes a number of data points collected and analyzed through honeypots. Due to this reason, it is highly unbalanced and the vast majority of the data is for attacks (97% of the entire data). The number of features is 24 in total (14 basic and 10 extended features). For this dataset, we excluded 6 minor features related to the host and port information in our experiments.

#### 3) UNSW-NB15 DATASET [17]

Similar to KDD Cup 1999, the UNSW-NB15 dataset was collected from a simulated environment in 2015. A set of servers generate traffic in the network, and a set of routers create pcap files as the capture of packets. Then Bro-IDS[1] classifies the captured data to construct the label information. The number of features is 49 for a single data point, with the associated label information. The total number of records is over two million in four CSV files, and the dataset also offers a training set and testing set for the evaluation purpose.

#### 4) IDS2017 DATASET [29]

This IDS2017 dataset also provides the labeled connection data for the network intrusion detection research. The data was captured from Monday, July 3, 2017 to Friday July 7, 2017. from a testbed system with two networks, one for attacks and the other for victims. The dataset contains a set of attacks, including DoS, Web attack, infiltration attack, botnet attack, port scan, and so forth, in addition to the normal traffic. The first day data contains the normal traffic only, while the data for the other days have attacks. Each data point contains 85 features including the label, and the number of data points is over 2.8 million.

[1]https://www.bro.org/

#### 5) MAWILAB DATASET [30]

MAWI is a collection of network traffic traces captured from a link between Japan and the USA since 2001 [36] [37]. The MAWILab logs contain the attack information collected from multiple intrusion detections systems against the captured packets. In our previous work [38], we combined the captured packet traces with the IDS logs to generate the labeled data for network anomaly detection. A single day trace contains the packets captured in the 15-minute interval, and the number of data points for a day is tens of millions. For the day of August 27, 2017, for example, a five-second data contains roughly 400,000 data points on average.

## III. DEEP LEARNING MODELS

In this section, we present a set of deep learning models established based on the FCN, VAE, and Seq2Seq structures. FCN is a simple form of neural networks. We also consider VAE to see if the dimension reduction by the auto-encoding function would be beneficial to deal with the network-related data. In addition, we include Seq2Seq with LSTM cells with the power to convert input features to context vectors as will be described in detail next. Table 2 summarizes the symbols and notations commonly used for the mathematical formulas for machine learning models.

**TABLE 2. Symbols and notations.**

| Symbol | Description |
|---|---|
| $H(x)$ | Hypothesis of machine learning |
| $x$ | Input data |
| $m$ | Size of $x$ |
| $W$ | Weights |
| $b$ | Bias |
| $x^{(i)}$ | $i$-th input data |
| $y^{(i)}$ | $i$-th output data |
| $cost(W, b)$ | Cost function for a learning model |

### A. FCN MODEL

The fundamental concept of machine learning is based on the following mathematical formula [39]:

$$H(x) = Wx + b \tag{1}$$

where $H(x)$ is a hypothesis, $x$ indicates input data, $W$ and $b$ denote weights and bias, respectively. The main goal of learning a neural network using the above formula as a linear regression model is to find a set of the optimized network parameters which minimize a cost function. The most popular algorithm used to minimize the cost function for the linear regression model is gradient descent, and this can be denoted as:

$$cost(W, b) = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)}) - y^{(i)})^2 \tag{2}$$

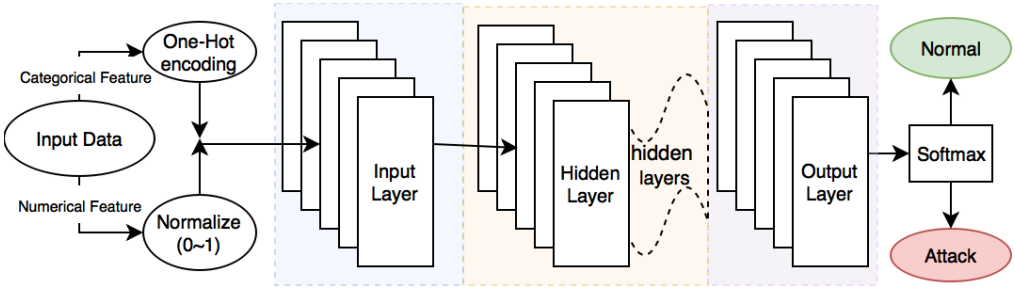



**FIGURE 2.** Overview of the FCN model. The number of hidden units and layers are configurable, and the loss function defined in this model is cross entropy.

where $m$ is the total number of $x$ input data, and $x^{(i)}$ and $y^{(i)}$ denote the $i$-th input data and $i$-th output (or prediction). Note that two formulas above are the fundamental concept of a multi-variable linear regression and multinomial logistic regression. Yet, machine learning based on the concept above results in one main issue which cannot linearly separate the exclusive-OR (XOR) gate. To address this problem, back-propagation was proposed as a solution, which is the chain rule-based algorithm [23], [40]. Back-propagation is the main idea of FCN. The basic concept of FCN is to learn non-linear combinations of given data based on matrix multiplications, and every neuron in fully connected layers is connected to each other.

For network anomaly detection, we design a deep learning model based on the FCN structure. Figure 2 shows the overview of our FCN-based anomaly detection model. The first step in this model is data preprocessing for normalization and transformation. Numerical features are normalized using MinMax scaling, while categorical features are encoded as a set of dummy numerical values using one-hot encoding. The model then passes the pre-processed data to the fully connected network for training. To overcome the vanishing gradient problem, Rectified Linear Unit (ReLU) is used as an activation function, and the Softmax layer with a cross entropy cost function is added to produce two dimensional outputs, i.e., either normal or attack.

### B. VAE MODELS (WITH/WITHOUT A BACKEND FCN)

Autoencoder is a reconstruction-based neural network [24] which is composed of encoder and decoder. If one hidden layer is given, Eq. 3 indicates that the encoder maps the original input vector $x$ to the latent representation $z$:

$$z = \sigma(Wx + b) \tag{3}$$

where $\sigma$ is an activation function such as sigmoid and ReLU, and $W$ and $b$ are the weight and bias, respectively.

The decoder maps $z$ back to $x'$ for the purpose of reconstruction:

$$x' = \sigma'(W'z + b') \tag{4}$$

where $\sigma'$, $W'$, and $b'$ are an activation function, weight, and bias for the decoder.

The main goal here is to minimize a reconstruction error by finding the minimal difference between $x$ and $x'$; that is, the reconstruction error is computed by:

$$|| x - x' || \tag{5}$$

VAE is a variation of the basic auto-encoder structure. VAE additionally refers to a generative, multi-layered, and directed probabilistic graphical model. The key concept is to infer $q(z|x;\ \phi)$ and $p(x'|z;\ \theta)$, where $q(z|x;\ \phi)$ is the likelihood of the latent variable $z$ given the data $x$, and $p(x'|z;\ \theta)$ is the likelihood of the data $x'$ given the latent variable $z$. Note that $z$ is the starting point of the generative process, and $\phi$ and $\theta$ are the parameters updated in the training phase.

One of the important functions in VAE is to calculate the marginal likelihood of data, which is the sum over the marginal likelihood of individual data points, denoted as:

$$log\ p(x^{(i)};\ \theta) = \sum_{i=1}^{n} log\ p(x^{(i)};\ \theta) \tag{6}$$

Alternatively, Eq. 6 can be denoted as Eq. 7:

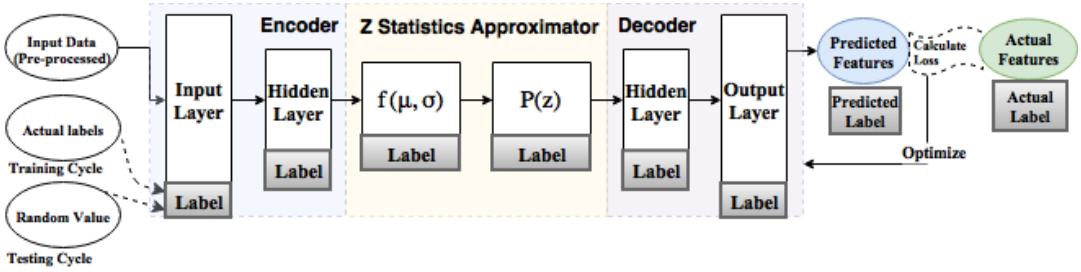
$$log\ p(x^{(i)};\ \theta) = KL(q(z|x;\ \phi)\ ||\ p(z;\ \theta)) + L(\theta, \phi; x^{(i)}) \tag{7}$$

where $q(z|x;\ \phi)$ and $p(z;\ \theta)$ are the approximate posterior and prior distribution of $z$, respectively, $KL$ is the Kullback-Leibler (KL) divergence, and $L(\theta,\ \phi,\ x^{(i)})$ is the variational lower bound on the marginal likelihood of the $i$-th data point. Since the KL divergence is always higher than a zero [24] [41] [42], it can be rewritten as follows:

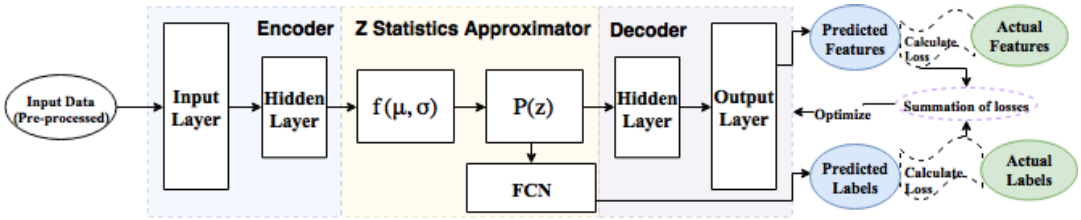
$$L(\theta, \phi; x^{(i)}) = E_{q(z|x^{(i)};\phi)}[log\ p(x'|z;\ \theta)] - KL(q(z|x^{(i)};\ \phi)\ ||\ p(z;\ \theta)) \tag{8}$$

Here, $E_{q(z|x^{(i)};\phi)}$ describes the reconstruction error, and $KL(q(z|x^{(i)};\ \phi)\ ||\ p(z;\ \theta))$ indicates the similarity between the approximate posterior and prior distribution of $z$. One important precondition here is that both $q(z|x^{(i)};\ \phi)$ and $p(z;\ \theta)$ have a form of the normal distribution. Furthermore, if binary-based data is given, Bernoulli distribution needs to be used for the distribution of the likelihood $p(x|z;\ \theta)$.

With the VAE concept, we implement two models for network anomaly detection: VAE-Pure is a simple VAE model (Figure 3(a)), whereas VAE-FCN connects a VAE to a fully




(a) VAE with generated labels (VAE-Pure)




(b) VAE with FCN (VAE-FCN)

**FIGURE 3.** Overview of two VAE models. VAE-Pure is a model based on label inclusion and the labels are treated as an independent feature. VAE-FCN does not utilize the label information, and runs in an unsupervised manner. The loss is calculated by comparing the original input data ($x$) to the output data ($x'$).

connected neural network at the end (Figure 3(b)). For VAE-Pure, the labels are treated as an independent feature in our design. In detail, actual labels are included in the training phase and the value of the label is randomly chosen to have a form of the normal distribution ($\mu = 0$ and $\sigma = 1$) in the testing phase. This VAE model performs learning to regenerate the labels out of the provided random values. The motivation why the VAE-Pure model includes the labels in the training phase is to see if it would be possible to predict the appropriate label in the reconstruction phase as an independent feature by the auto-encoder function.

VAE-FCN in Figure 3(b) is almost identical to the previous one (Figure 3(a)) except that this VAE model performs learning in an unsupervised manner by regenerating the dataset and learning through the loss function. A separate FCN with 1 hidden layer (ReLU activation) and 1 Softmax layer is attached to this VAE model. This FCN receives inputs from the probability distribution of $z$ and produces label probabilities as the output. Both of the VAE and the attached FCN are trained together in this model.

### C. SEQ2SEQ MODEL

Another model we establish for network anomaly detection utilizes the Sequence-to-Sequence (Seq2Seq) structure which is composed of two RNNs corresponding to the encoder and decoder. Both encoder and decoder can have a single or multiple cells, which could be Long Short-Term Memory (LSTM) or Gate Recurrent Unit (GRU). We simply chose LSTM for the memory cells in our Seq2Seq model since LSTM is powerful to overcome the vanishing gradient problem [43]. Figure 4 shows the Seq2Seq model designed in this work.

In the Seq2Seq architecture, all the inputs are encoded into a fixed-size vector, and the encoded vector is the only one

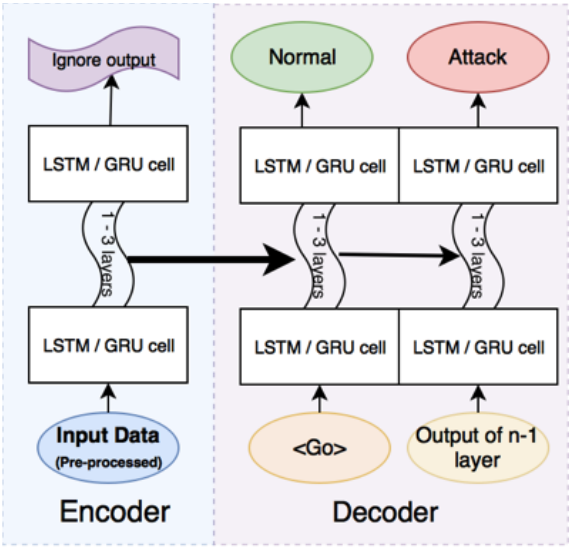



**FIGURE 4.** Overview of the Seq2Seq model composed of the encoder and decoder. There can be one or more cells for each LSTM stack and we set it to three by default. The loss function used in this model is Mean Squared Error.

which is passed to the decoder. Hence, the goal of Seq2Seq is to yield a target sequence ($y_{T'}$) and conditional probability ($p(y_{T'}|x_T)$) through the architecture [25], [26]. The definition of the notations is as follows:

- $x$ is an input sequence, $x = \{x_1, x_2, ...x_T\}$ depending on a time step ($t$)
- $c$ is a fixed-length vector, $c = q(\{h_1, h_2, ...h_{T_x}\})$, obtained by the last hidden state of the LSTM. Here $q(\cdot)$ is a non-linear activation function.
- $y$ is a target (output) sequence, $y = \{y_1, y_2, ...y_{T'}\}$ which is corresponding to $x = \{x_1, x_2, ...x_T\}$.

The main objective of the encoder is to read $x = \{x_1, x_2, ...x_T\}$ into $c$ sequentially, and the first two conditions refer to the encoder. Eq. 9 is applied to the encoder to update the hidden state, and the LSTM as a non-linear function $f$ is used:

$$h_t = f_w(h_{t-1}, x_t) \tag{9}$$

where $h_t$ is a new state, $f_w(\cdot)$ is a function with the parameters of $h_{t-1}$ and $x_t$. Here $h_{t-1}$ is an old state and $x_t$ is an input vector at a time step.

The decoder associated with the last condition trains a model to predict $y_{T'}$ with given $c$ and $\{y_1, y_2, ...y_{t-1}\}$, and a conditional probability $P(y|x)$ is computed by the following equation:

$$P(y|x) = \prod_{t=1}^{T'} p(y_t|c, y_1, y_2, ...y_{t-1}) \tag{10}$$

where $p(y_t|c, y_1, y_2, ...y_{T-1})$ is a distribution represented with the Softmax function.

The LSTM cell shown in Figure 4 is basically composed of three gates: input, forget, and output. The details about how each gate is updated can be found from [44]. As mentioned, GRU could be configured instead of LSTM, but we simply chose LSTM for our Seq2Seq model.

Here are some details of the implementation of our Seq2Seq model for network anomaly detection. As mentioned, we chose LSTM for the memory cells to construct the Seq2Seq model. We considered one and three cells for the LSTM stack in our experiments and observed that configuring three cells works much better with little increase of learning complexity. It is possible to add more cells in the stack, but our observation tells that three cells would be enough with the excellent performance (over 99% of accuracy in detection). For this reason, we set it to three by default. The output of the previous LSTM layer (denoted as $t-1$) is supplied to the subsequent LSTM layer in the decoder at time $t$. In the figure, the "Go" signal indicates the input in the decoder as a tensor in TensorFlow. Once the LSTM stack in the decoder is executed, it produces the probability for normal and anomalous, and the model picks the greater one as the classification decision. The loss function used is Mean Squared Error.

## IV. EVALUATION

We evaluate the deep learning models presented in the previous section. In this section, we report the experimental results performed with the public datasets: NSL-KDD, Kyoto-Honeypot, UNSW-NB15, IDS2017, and MAWILab datasets.

### A. EVALUATION METRICS

We employ a set of metrics that are widely accepted in practice for evaluating classification performance. The confusion matrix is a table, in which the elements are referenced to measure the performance of a classification model. The elements in the matrix are as follows:

- True Negative ($TN$): the number of normal instances correctly identified
- False Negative ($FN$): the number of anomalies classified into normal
- True Positive ($TP$): the number of anomalies correctly classified
- False Positive ($FP$): the number of normal classified into anomalies

Based on the confusion matrix, the following metrics are defined:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$
$$\text{Precision} = \frac{TP}{TP + FP}$$
$$\text{Recall} = \frac{TP}{TP + FN}$$
$$\text{F-measure} = 2 \cdot \frac{precision \times recall}{precision + recall}$$

F-measure is a harmonic mean of the precision and recall of the test. While widely employed for evaluating classification performance, F-measure might lead to a biased result if the dataset is critically skewed. Alternatively, the measure of Matthew Correlation Coefficient (MCC) is a useful tool to estimate the quality of binary classification even for a biased dataset [45], the value of which is ranged from $-1.0$ (*poor*) to 1.0 (*good*). MCC is defined as follows:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

We report our experimental results with the metrics defined above. In particular, we use MCC to report the experimental results conducted against the Kyoto-Honeypot dataset since it is significantly skewed with only a small number of normal instances ($\sim$3%).

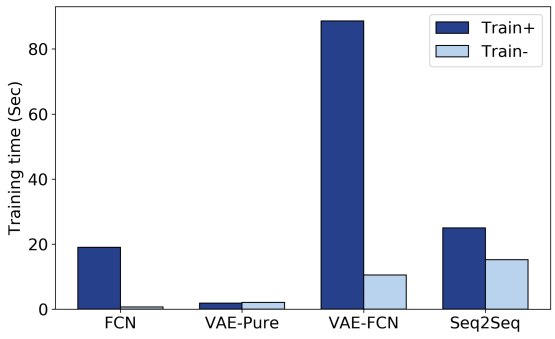
### B. EXPERIMENTAL RESULTS AGAINST THE NSL-KDD DATASET

In this first experiment, we report *training time* to measure the learning complexity and *F-measure* for the performance of classification. F-measure (also known as *F1-score*) combines precision and recall and is widely used for measuring classification performance.

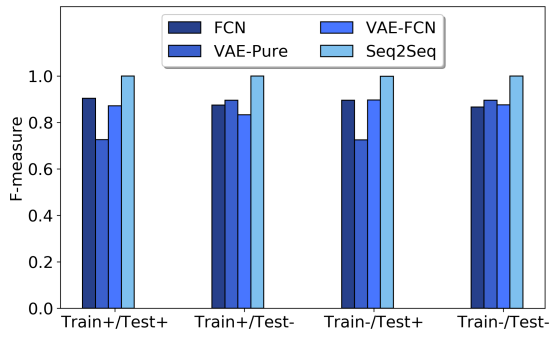
We experimented with a set of hyper-parameter configurations as follows and report the best result for each model:

- Number of hidden units = {25% of the entire features, 50% of the entire features, 75% of the entire features, and 100% of the entire features}
- Number of hidden layers = {1, 3}
- Epochs = {10, 20, 40, 80}
- Learning rate = {1e-02, 1e-03, 1e-04, 1e-05}

Figure 5 shows the measured training time and classification performance in F-measure. To measure the training time,




(a) Training time




(b) Performance (F-measure)

**FIGURE 5.** Experimental results against NSL-KDD: (a) training time for the two training files (Train+ and Train-), and (b) F-measure for the combinations of the training and testing files.

the experiment was conducted on a dedicated machine in the Google cloud. Figure 5(a) shows the time taken (in seconds) to construct the learning model for each of the training files. The figure shows VAE-Pure < FCN < Seq2Seq, with respect to the cost of learning. The Seq2Seq model shows a little greater overhead than FCN and VAE-Pure, but the training cost is quite manageable (i.e., less than 25 seconds for over 125K data points in Train+). Although not shown in the figure, the Seq2Seq model with a single cell in the LSTM stack shows a slightly smaller learning time compared to one with three cells in the stack but the gap is not significant (22% at max). Finally, VAE-FCN shows a high degree of training overhead for Train+, whereas the overhead is less than Seq2Seq for Train- (consisting of 25K data points). This implies that VAE-FCN would be limited with respect to scalability.

Figure 5(b) shows the performance for binary classification. From the figure, the Seq2Seq model shows a very promising performance (>99.9% of F-measure), regardless of the combinations of the datasets. In fact, the results are much better than what we observed from the experiment with conventional techniques shown in Table 1. FCN shows somewhat consistent results over 86% of F-measure, outperforming the conventional learning techniques. The VAE-class models do not show better performance than FCN, yielding 72%–90% of F-measure.

**TABLE 3.** Training and testing datasets selected from Kyoto-Honeypot.

| Dataset | Dates | # records/day |
|---|---|---|
| Training | January 1–7, 2014 | 268K |
| Testing | December 1–31, 2015 | 236K |

**TABLE 4.** Experimental result against daily data (December 1st, 2016) of Kyoto-Honeypot.

| Model | Precision | Recall | F-measure | MCC |
|---|---|---|---|---|
| FCN | 99.7% | 87.4% | 93.1% | 0.37 |
| VAE-Pure | 97.5% | 75.3% | 85.0% | 0.05 |
| VAE-FCN | 98.1% | 90.1% | 93.9% | 0.19 |
| Seq2Seq | 100.0% | 100.0% | 100.0% | 1.0 |

### C. EXPERIMENTAL RESULTS AGAINST THE KYOTO-HONEYPOT DATASET

As summarized in Section II-A, Kyoto-Honeypot contains a large amount of daily connection data gathered through honeypots since 2009. In our experiments, we selected a subset of data without any preference other than the 2-year gap for the training and testing datasets, not to choose too close data temporally for the learning and actual classification. Table 3 shows the information of the data selected.

As discussed earlier, the data is significantly skewed and 97% of the records are the data points for attacks, as the collection took place in honeypots. Due to this severe imbalance with respect to the fractions of normal and attack data points, we do not rely only on F-measure to measure the performance because it could lead to a critical bias otherwise. For example, if a model in evaluation simply classifies all the data points into attack, the resulted F-measure would still be around 97%. For this reason, we employ MCC to report the experimental results conducted against this highly skewed dataset.

The hyper-parameter configurations used in the experiments are as follows:

- Number of hidden units = {1 unit, 10% of the entire features, 20% of the entire features, 40% of the entire features, and 100% entire features}
- Number of hidden layers = {1, 3}
- Epochs = 10
- Learning rate = 1e-02

We now report the experimental results against the Kyoto-Honeypot data. We observed as follows for training a one-day data (including 268K records): FCN (3.93 sec), VAE-Pure (3.87 sec), VAE-FCN (6.54 sec), and Seq2Seq (9.20 sec). We can see that the training cost is not much expensive even with a single dedicated server. As shown in Figure 5(a), Seq2Seq sometimes requires a greater amount of time for training than VAE-FCN, but it does not show a high degree of variance depending on input data unlike VAE-FCN.

We next report the performance of the deep learning models. Table 4 shows the measured performance against a daily

**TABLE 5. Quality score (MCC) between December 1–31, 2015 in Kyoto-Honeypot.**

| Model | Best | Worst | Average |
|---|---|---|---|
| FCN | 0.54 | 0.02 | 0.19 |
| VAE-Pure | 0.06 | 0.00 | 0.01 |
| VAE-FCN | 0.38 | 0.00 | 0.09 |
| Seq2Seq | 1.00 | 1.00 | 1.00 |

data (December 1st, 2016) with a set of measures. As can be seen from the table, Seq2Seq consistently outperforms showing almost perfect performance. The F-measure shows that the models seem to be working well, but we can see that the MCC values are far from 1.0 other than the Seq2Seq model. In particular, the VAE models show less than 0.2 for MCC, which is unacceptable. VAE-FCN works better than VAE-Pure, but it still shows a significantly high FP rate (55.2%, not shown in the table due to space reasons). FCN outperforms the VAE models. The F-measure of FCN is quite high (0.93), but it shows a low score of MCC = 0.37.

We also conducted a set of experiments against the entire data collected in December 2015 on a daily basis, and Table 5 reports the measured MCC values over the month. Here, "Average" is the aggregated MCC over the 31 days, while "Best" and "Worst" are the best/worst result out of 31 days. The Seq2Seq model works consistently, and Worst is still 1.0 (perfect). Overall, FCN works better than the VAE models as observed in the previous section.

### D. EVALUATION WITH UNSW-NB15 AND IDS2017 DATASETS

We next report the experimental results conducted with the UNSW-NB15 [17] and IDS2017 [29] datasets. UNSW-NB15 provides a pair of training and testing data files, the size of which are training = 82K and testing = 175K, in terms of the number of data points. The IDS2017 dataset consists of 5-day traces from Monday to Friday, and daily trace records include no or only a subset of attacks selectively. Therefore, for example, if we use the first day trace for training, we cannot build a model for any attack because it does not contain any attack record at all. The second day trace contains FTP and SSH attacks but nothing about DoS/DDoS, Web, and infiltration attacks that are found in the rest of the days. For this reason, we sampled ten sets of 100K data points at random, out of which five files are used for training and the other five files are for testing. The data files were then pre-processed for the normalization for the numeric features and for the one-hot encoding for the categorical features.

Table 6 shows a summary of the results. The table compares the performance of the deep learning models (FCN and Seq2Seq) and the shallow learning methods (SVM and RF). Due to the poor performance, we discarded the VAE models for the subsequent experiments. For UNSW-NB15, we can see that the Seq2Seq classifier consistently outperforms the other learning methods. However, FCN shows a worse performance than SVM and RF in F1-score; it shows a higher accuracy but also shows a high degree of false positive rate, degrading the overall performance. RF works better than SVM and FCN with this dataset.

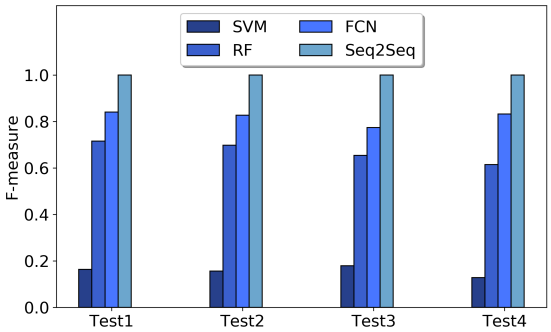



**FIGURE 6. Performance comparison with the MAWILab data: the deep learning models work better than the shallow learning models, and Seq2Seq significantly outperforms the others.**

The result for IDS2017 is averaged out from the five independent experiments, as mentioned above. Similar with the result collected using UNSW-NB15, Seq2Seq outperforms the other methods for IDS2017. FCN works pretty well with the higher performance in F1-score than RF and SVM; it shows a higher rate for Recall than Precision like the result with UNSW-NB15, which indicates that this classifier tends to strictly classify the samples in testing into anomalies. For RF and SVM, we observed a very high degree of variations from the individual experiments, and the standard deviation is 16.9% (SVM) and 4.7% (RF), while it is less than 0.11% for both FCN and Seq2Seq. This implies that the deep learning models work more in a consistent manner.

### E. EVALUATION WITH A MAWILAB TRACE

Finally, we report our experimental results against the MAWILab data. We chose the trace collected on August 27th, 2017 without any preference. A daily trace contains 15-minute packet data collected on that day. We split the trace chosen into a set of small files in a disjoint manner, each of which contains the packets for five seconds, and we used the first five sub-data for our experiments. As shown in Table 7, the first file is used for training and the subsequent four files were utilized for testing. The table also provides the information about the fraction of normal and anomaly data points.

For learning and testing, we considered the following five features: protocol, TCP flags, number of packets, number of bytes, and duration. The total number of features after one-hot encoding is 23. We assumed epochs = 20 and learning rate = 1e-02 as the default setting in the experiments.

In this experiment, we focused on the performance of the Seq2Seq model to see if it consistently yields a high degree of detection accuracy. Figure 6 compares the performance of the shallow learning models (SVM and RF) and the deep learning

**TABLE 6.** Experimental results with UNSW-NB15 and IDS2017 datasets.

| Dataset | Classifier | Accuracy | Precision | Recall | F1-score |
|---|---|---|---|---|---|
| UNSW-NB15 | SVM | 0.819 | 0.973 | 0.756 | 0.821 |
| | RF | 0.853 | 0.975 | 0.805 | 0.882 |
| | FCN | 0.857 | 0.701 | 0.938 | 0.807 |
| | Seq2Seq | 1.000 | 1.000 | 1.000 | 1.000 |
| IDS2017 | SVM | 0.742 | 0.567 | 0.782 | 0.608 |
| | RF | 0.954 | 0.952 | 0.763 | 0.846 |
| | FCN | 0.834 | 0.834 | 1.000 | 0.910 |
| | Seq2Seq | 1.000 | 1.000 | 1.000 | 1.000 |

**TABLE 7.** MAWILab dataset from August 27, 2017.

| File | # data points | # normal | # anomaly | % anomaly |
|---|---|---|---|---|
| Train | 383,615 | 266,593 | 117,022 | 30.5% |
| Test1 | 407,807 | 291,488 | 116,319 | 28.5% |
| Test2 | 472,654 | 327,413 | 145,241 | 30.7% |
| Test3 | 423,984 | 261,426 | 162,558 | 38.3% |
| Test4 | 425,994 | 300,759 | 125,235 | 29.4% |

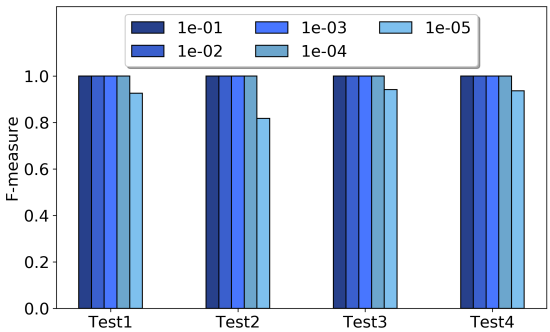



**FIGURE 7.** Impact of learning rate for the Seq2Seq model: the model works consistently over the different testing datasets if the learning rate is not too small.

models (FCN and Seq2Seq) against the MAWILab dataset. As can be seen from the figure, the deep learning models work better than the shallow learning models. While FCN produces 77%–84% of F-measure, Seq2Seq works consistently across the testing sets showing 100% accuracy.

Figure 7 shows the impact of the learning rate for the Seq2Seq model against the MAWILab data. As can be seen from the figure, Seq2Seq works highly consistently if the learning rate is not too small. However, it degrades significantly with the very small learning rate of 1e-05. We also measured the impact of epochs with a set of epochs $= \{10, 20, 40, 80\}$, and observed no significant impact.

In sum, the deep learning model based on the Seq2Seq structure with LSTM works very well for network anomaly detection, consistently yielding over 99% of detection accuracy over the diverse datasets with different characteristics. For cross-validation, the implementation of the deep learning models presented in this paper is available at https://github.com/dcstamuc/NetworkAnomaly.

## V. CONCLUSION

While machine learning has widely been applied to various applications in different domains, few studies exist that examined deep learning for the study of network anomaly detection in depth. In this work, we claimed why deep learning would be a good choice for effective network anomaly detection by showing the non-linearity of network traffic data. To evaluate the potential of deep learning, we established a set of deep learning models based on FCN, VAE, Seq2Seq structures, and examined the constructed models with a diverse set of public traffic datasets including NSL-KDD, Kyoto-Honeypot, UNSW-NB15, IDS2017, and MAWILab. Our experimental results are interesting and the model based on the Seq2Seq structure shows the highly promising performance yielding over 99% of accuracy to identify network anomalies across the entire datasets.

In this study, we reported the best performance for each model from the results obtained with a set of hyper-parameter configurations. How to tune the parameters is an important task and we plan to investigate to address this problem as one of the future tasks. This study also assumed a relatively small number of hidden layers (1 or 3) for the deep learning structures, and extensive evaluations with a greater number of layers will be one of the interesting future tasks. In addition, this evaluation study does not include the Convolutional Neural Network (CNN) structure, and examining the feasibility of CNNs for network anomaly detection would also be interesting for future investigation.

## REFERENCES

[1] *Gartner Provides Three Immediate Actions to Take as WannaCry Ransomware Spreads*. Accessed: Jul. 29, 2018. [Online]. Available: http://www.gartner.com/newsroom/id/3715918

[2] *Large DDoS Attacks Cause Outages at Twitter, Spotify, and Other Sites*. Accessed: Jul. 29, 2018. [Online]. Available: http://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf

[3] *DDoS Attack Snarls Friday Morning Internet Traffic*. Accessed: Jul. 29, 2018. [Online]. Available: http://www.eweek.com/security/ddos-attack-snarls-friday-morning-internet-traffic.html

[4] D. Liu, Y. Zhao, H. Xu, Y. Sun, D. Pei, J. Luo, X. Jing, and M. Feng, "Opprentice: Towards practical and automatic anomaly detection through machine learning," in *Proc. ACM IMC*, 2015, pp. 211–224.

[5] A. K. Shrivas and A. K. Dewangan, "An ensemble model for classification of attacks with feature selection based on KDD99 and NSL-KDD data set," *Int. J. Comput. Appl.*, vol. 99, no. 3, pp. 8–13, 2014.

[6] M. Panda, A. Abraham, and M. R. Patra, "Hybrid intelligent systems for detecting network intrusions," *Secur. Commun. Netw.*, vol. 8, no. 16, pp. 2741–2749, 2015.
[7] E. E. Papalexakis, A. Beutel, and P. Steenkiste, "Network anomaly detection using co-clustering," in *Proc. Int. Conf. Adv. Social Netw. Anal. Mining (ASONAM)*, 2012, pp. 403–410.
[8] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, Jul. 2009, Art. no. 15.
[9] Y. Liu, L. Zhang, and Y. Guan, "A distributed data streaming algorithm for network-wide traffic anomaly detection," *SIGMETRICS Perform. Eval. Rev.*, vol. 37, no. 2, pp. 81–82, Oct. 2009.
[10] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. 2nd IEEE Int. Conf. Comput. Intell. Secur. Defense Appl. (CISDA)*, Jul. 2009, pp. 53–58.
[11] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *Proc. IEEE Int. Conf. Wireless Netw. Mobile Commun. (WINCOM)*, Oct. 2016, pp. 258–263.
[12] B. Shah and B. H. Trivedi, "Reducing features of KDD CUP 1999 dataset for anomaly detection using back propagation neural network" in *Proc. IEEE 5th Int. Conf. Adv. Comput. Commun. technol. (ACCT)*, Feb. 2015, pp. 247–251.
[13] K. K. Nguyen, D. T. Hoang, D. Niyato, P. Wang, D. Nguyen, and E. Dutkiewicz, "Cyberattack detection in mobile cloud computing: A deep learning approach," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2018, pp. 1–6.
[14] Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, "Intrusion detection using convolutional neural networks for representation learning," in *Proc. Int. Conf. Neural Inf. Process.* Cham, Switzerland: Springer, 2017, pp. 858–866.
[15] T. Ma, F. Wang, J. Cheng, Y. Yu, and X. Chen, "A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks," *Sensors*, vol. 16, no. 10, p. 1701, 2016.
[16] M.-T. Luong, Q. V. Le, I. Sutskever, O. Vinyals, and L. Kaiser, "Multi-task sequence to sequence learning," 2015, *arXiv:1511.06114*. [Online]. Available: https://arxiv.org/abs/1511.06114
[17] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. IEEE Military Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6.
[18] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.
[19] Q. V. Le, "A tutorial on deep learning part 2: Autoencoders, convolutional neural networks and recurrent neural networks," *Google Brain*, pp. 1–20, Oct. 2015.
[20] L. Deng and D. Yu, "Deep learning: Methods and applications," *Found. Trends Signal Process.*, vol. 7, nos. 3–4, pp. 197–387, Jun. 2014.
[21] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *J. Big Data*, vol. 2, no. 1, p. 1, Feb. 2015.
[22] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Comput.*, vol. 22, pp. 949–961, Jan. 2019.
[23] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2285–2294.
[24] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," Seoul Nat. Univ., Seoul, South Korea, Tech. Rep. SNUDM-TR-2015-03, 2015.
[25] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, *arXiv:1409.0473*. [Online]. Available: https://arxiv.org/abs/1409.0473
[26] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.
[27] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," 2014, *arXiv:1406.1078*. [Online]. Available: https://arxiv.org/abs/1406.1078
[28] *Kyoto*. Accessed: Jul. 29, 2018. [Online]. Available: http://www.takakura.com/Kyoto_data/
[29] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. ICISSP*, 2018, pp. 108–116.
[30] R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda, "MAWILab: Combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking," in *Proc. 6th Int. Conf. (Co-NEXT)*, 2010, Art. no. 8.
[31] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.
[32] T. Aldwairi, D. Perera, and M. A. Novotny, "An evaluation of the performance of restricted Boltzmann machines as a model for anomaly network intrusion detection," *Comput. Netw.*, vol. 144, pp. 111–119, Oct. 2018.
[33] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur.*, vol. 31, no. 3, pp. 357–374, 2012.
[34] P. Illy, G. Kaddoum, C. M. Moreira, K. Kaur, and S. Garg, "Securing fog-to-things environment using intrusion detection system based on ensemble learning," 2019, *arXiv:1901.10933*. [Online]. Available: https://arxiv.org/abs/1901.10933
[35] *KDD Cup 1999 Data*. Accessed: Jul. 29, 2018. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[36] J. Mazel, R. Fontugne, and K. Fukuda, "Visual comparison of network anomaly detectors with chord diagrams," in *Proc. 29th Annu. ACM Symp. Appl. Computing*, pp. 473–480 ACM, 2014.
[37] C. Callegari, S. Giordano, and M. Pagano, "Statistical network anomaly detection: An experimental study," in *Proc. Int. Conf. Future Netw. Syst. Secur.* Cham, Switzerland: Springer, 2016, pp. 12–25.
[38] J. Kim, C. Sim, and J. Choi, "Generating labeled flow data from MAWILab traces for network intrusion detection," *CoRR*, vol. abs/1810.01945, pp. 1–4, Oct. 2018. [Online]. Available: https://arxiv.org/abs/1810.01945
[39] K.-L. Du and M. Swamy, "Fundamentals of machine learning," in *Neural Netw. Statistical Learn.*, pp. 15–65 Springer, 2014.
[40] M. Cilimkovic, "Neural networks and back propagation algorithm," Inst. Technol. Blanchardstown, Blanchardstown Road North Dublin, Dublin, Ireland, 2015.
[41] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*. [Online]. Available: https://arxiv.org/abs/1312.6114
[42] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *J. Amer. Stat. Assoc.*, vol. 112, no. 518, pp. 859–877, 2017.
[43] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *Proceedings*. Louvain-la-Neuve, Belgium: Presses Univ. Louvain, 2015, p. 89.
[44] D. Neil, M. Pfeiffer, and S.-C. Liu, "Phased LSTM: Accelerating recurrent network training for long or event-based sequences," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3882–3890.
[45] P. Baldi, S. Brunak, Y. Chauvin, C. A. F. Andersen, and H. Nielsen, "Assessing the accuracy of prediction algorithms for classification: An overview," *Bioinformatics*, vol. 16, no. 5, pp. 412–424, May 2000.

**RITESH K. MALAIYA** received the bachelor's degree in computer engineering from Bharati Vidyapeeth University, India. He is currently pursuing the M.Sc. degree in applied cognition and neuroscience with the University of Texas at Dallas. He was a Software Engineer with IT industry field for nine years, from 2007 to 2016. He has participated in the Smart Connected Car Research Project, from January 2016 to November 2017, and developed the deep learning-based blind spot detection system. His research interests include computational models of brain and behavior, probabilistic graphical modeling, and adaptive learning.

**DONGHWOON KWON** received the D.Sc. degree in applied information technology from Towson University, Towson, MD, USA. He was an Assistant and an Adjunct Professor with the Department of Computer Science, Texas A&M University-Commerce. He has been an Assistant Professor with the Department of Mathmatics, CSCI, and Physics, Rockford University, Rockford, IL, USA, since August 2018. He has participated in the Smart Connected Car Research Project, from August 2016 to November 2017, and developed the deep learning-based blind spot detection system. His research interests include the Internet of Things (IoT), cloud computing, big data with machine learning, and mobile application, as well as, IT/IS project management in conjunction with software engineering and database management.

**SANG C. SUH** founded the Transdisciplinary Research Center called the Intelligent Cyberspace Engineering Lab (ICEL) to launch and carry out many transdisciplinary systems Research and Development projects. He is currently a Professor and the Head of the Computer Science Department, Texas A&M University-Commerce, Commerce, TX, USA. He has authored and published more than a hundred peer-reviewed scientific articles, several book chapters, and books in the areas of data mining, bioinformatics, cyber-physical systems, knowledge and data engineering, visual analytics, and adaptive search engines. His research theme and major research thrusts of his ICEL spans around many interdisciplinary research topics, including bio-informatics and biological sciences, knowledge discovery and engineering, human–computer interaction, data mining, big data, visual analytics, and adaptive search engines.

**HYUNJOO KIM** received the M.S. and Ph.D. degrees in computer engineering from Sungkyunkwan University, South Korea, in 2002 and 2016, respectively. She joined the Electronics and Telecommunications Research Institute (ETRI), in 2002. She is currently a Senior Member of the Engineering Staff of the Intelligent Security Research Group. Her research interests include the malware analysis, network security, cloud security, and bigdata analytics.

**IKKYUN KIM** received the M.S.C.E. and Ph.D. degrees in computer engineering from the Kyungpook National University, South Korea. He was with Electronics and Telecommunications Research Institute (ETRI), South Korea, from 1996 to 1999. From 2000 to 2001, he was a Research Staff Member with Paxcomm Ltd. Since 2001, he has been with the Cybersecurity Research Division, ETRI, where he is currently the Head of the Division. He was a Visiting Scholar with Purdue University, in 2004. He has developed several network-based intrusion detection systems. He is also involved in new anomaly detection method against zero-day attacks for network security. His research interests include DDoS protection mechanism, high-speed network protection systems, the design of network processor architecture for network security appliance, and big data security analytics for security intelligence.

**JINOH KIM** received the Ph.D. degree in computer science from the University of Minnesota, Minneapolis, MN, USA. He has been a Researcher and a Senior Researcher with ETRI, South Korea, since 1991. He was a Postdoctoral Researcher with the Lawrence Berkeley National Laboratory, from 2010 to 2011, and an Assistant Professor of computer science with the Lock Haven University, Lock Haven, PA, USA, from 2011 to 2012. He is currently an Associate Professor of computer science with Texas A&M University-Commerce, Commerce, TX, USA. He has participated in various projects in the areas of network security and ATM Networks. His research interests include span from distributed systems to networking and cybersecurity, cybersecurity analytics, systems/network telemetry and analytics, and data access performance in clouds.