An Encoding Technique for CNN-based Network Anomaly Detection

Taejoon Kim* Sang C. Suh* Hyunjoo Kim† Jonghyun Kim† Jinoh Kim*
Texas A&M University, Commerce, TX 75428, USA *
ETRI, Yuseong-gu, Daejeon, 34129, Korea †
Email: tkim3@leomail.tamuc.edu, sang.suh@tamuc.edu, hjookim@etri.re.kr, jhk@etri.re.kr, jinoh.kim@tamuc.edu

***Abstract*—An important challenge in the cyber-space is the effective identification of network anomalies, often caused by malicious activities. With the remarkable advances, machine learning algorithms have widely been studied for network intrusion and anomaly detection. In particular, deep learning based on neural network structures has recently been given a greater attention to deal with the growing complexity of data with higher dimensions and non-linearity. Convolutional Neural Networks (CNNs) is one of the widely employed deep learning methods. In this work, we introduce a new encoding technique that enhances the performance for the identification of anomalous events using a CNN structure. To evaluate, we utilize three different datasets for the extensive analysis. The experimental results show that our method consistently outperforms the gray-scale encoding technique previously proposed over the datasets employed in the evaluation.**

***Index Terms*—Network anomaly detection, Convolutional Neural Networks, Data encoding, Data transformation**

I. Introduction

With the increasing reliance on networking, we have a growing concern about cyber-security. For instance, distributed denial of service (DDoS) attacks are prevalent in the Internet, and a recent analysis shows that DDoS attack volumes increased by 50% in Q2 2018.[1] Moreover, a new type of such attacks began to utilize hundreds of thousands of IoT devices[2], which considerably outnumbers the traditional computers. In addition, the impacts of cyber-attacks are much more significant. Several datacenters were targeted by DDoS attacks in 2016, which caused Twitter, Spotify, and other major sites to close down.[3] In 2017, a ransomware attack (known as "WannaCry") hit the global world affecting over 10,000 organizations in over 150 countries.[4] Data breach is also a growing concern with the financial and privacy problems. One example is the Equifax data breach incident taken place from May to July in 2017, in which the hackers stole credit card numbers for over 200,000 people.[5]

An important challenge in the cyber-space is the effective identification of network anomalies, often caused by malicious activities. With the remarkable advances, machine learning algorithms have widely been studied for network intrusion and anomaly detection. In particular, deep learning based on neural network structures has recently been given a greater attention to deal with the growing complexity of data with higher dimensions and non-linearity [1]. Convolutional Neural Networks (CNNs) are one of the widely employed deep learning methods, particularly for image processing and classification [2]. In addition to the traditional image classification, the CNN method has been studied for diverse applications, such as biomedical text analysis [3], malware classification [4], and anomalous crowded scenes detection [5], to name a few. In our previous work [1], we examined a set of deep learning models, including a fully connected neural network, variational autoencoder, and sequence-to-sequence structures, and showed the feasibility of deep learning for network anomaly detection. In this work, we evaluate the feasibility of CNN for the identification of malicious activities in the network.

To employ CNN for network anomaly detection, data encoding is an essential function to convert one dimensional connection records to an image-like, two-dimensional matrices. A past work in [6] introduced an encoding technique called "gray-scale encoding" for network intrusion detection. Our preliminary experiment in [7] shows that using a vector-type connection data (with no transformation of data) as input to CNN does not work well, yielding worse performance than the use of gray-scale encoding. While interesting, this past work was still limited with respect to the performance to detect anomalies and the reported classification accuracy is no better than the use of conventional shallow learning. In addition, the evaluation was limited with only a single dataset of NSL-KDD [8] (a modified version of KDD Cup 1999 data [9]) without comprehensive analysis. The NSL-KDD dataset would still be useful in considering the shortage of public datasets in the network security research community, but relying only on this old dataset would limit the quality of the evaluation.

In this work, we introduce a new encoding method that improves the performance for the identification of network anomalies using a CNN structure. The main difference is that our method allocates the equal weight to individual features, while the existing method gives a greater space to numeric features. For evaluating, we employ an extensive set of datasets to minimize the bias, including UNSW-NB15 [10], IDS2017 [11], as well as NSL-KDD [8], which will be

[1]https://www.link11.com/en/blog/latest-link11-ddos-report-shows-that-ddos-attack-volumes-increased-by-50-in-q2-2018/

[2]http://www.eweek.com/security/ddos-attack-snarls-friday-morning-internet-traffic.html

[3]http://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf

[4]http://www.gartner.com/newsroom/id/3715918

[5]https://www.consumer.ftc.gov/blog/2017/09/equifax-data-breach-what-do

978-1-5386-5035-6/18/$31.00 ©2018 IEEE

described in detail in the following section. Our experimental results with these datasets show that our encoding consistently outperforms the existing gray-scale encoding.

The organization of this paper is as follows. We provide a summary of the related studies and network traffic datasets in Section II. In Section III, we introduce our new encoding method with the comparison to the gray-scale encoding proposed in [6]. We report the experimental results conducted with the aforementioned datasets in Section IV, and conclude our presentation with the summary and future tasks in Section V.

II. Background

In this section, we summarize the past studies closely related to our work and provide a description of the datasets employed in this paper.

A. Related work

With its significant progress, machine learning has been widely employed for diverse applications including network anomaly detection. The work of [1] claimed why conventional shallow learning may not work for identifying anomalies from the network traffic datasets. This past work designed a set of deep learning models using fully connected neural networks, variational autoencoder, and sequence-to-sequence structures, and showed the feasibility of deep learning with greater accuracy in detection. The experimental results also showed that the sequence-to-sequence model outperforms the others consistently. This work however does not design and evaluate any model based on convolutional neural networks.

The CNN model was inspired by the animal visual cortex organization, which is in charge of light detection in overlapping and small partitions of the visual field [12]. Typically, a CNN consists of two types of layers called convolution layers extracting features and sampling layers for feature mapping. As it is from visual cortex, the input is often assumed as a two-dimensional image with a set of pixels. For this reason, the killer application of CNN has been basically image classification. With its potential, however, CNN was considered for other applications in diverse domains including network security. The authors of [4] employed the concept of CNN for identifying malware traffic from raw network traces. In the work of [13], the authors utilized CNN for denial of service (DoS) attacks. The studies in [14] and [6] applied CNN for network anomaly detection. In particular, the authors in [6] presented an encoding technique to convert the NSL-KDD connection record to a gray-scale pixel image. In this work, we examine the feasibility of CNN for network anomaly detection with a new encoding technique to improve the performance.

GoogLeNet [15] is a CNN model developed for computer vision and is the winner of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) competition in 2014. The architecture consists of a 22 layer deep CNN with additional pooling layers internally. An interesting part in GoogLeNet is the local inception module that figures out the optimal local construction. In the competition, it showed an error rate of 6.67% that is almost the same as the human level performance. With its benefits, GoogLeNet has been employed for various applications for internal learning and classification, including character recognition [16], [17], malware detection [18], and intrusion detection [6]. In this work, we employ the GoogLeNet architecture for internal learning and classification for network anomaly detection. Our primary contribution is the development of new encoding technique for network connection data to utilize the CNN model.

Despite its limitation, the KDD Cup 1999 dataset [9] and its modified version NSL-KDD that curtails redundant records to reduce the bias [8], has long been widely employed for evaluating network anomaly detection due to the shortage of public network datasets including the annotation information. For example, the studies of [1], [19]–[21] evaluated their methods developed based on deep learning techniques using this class of datasets. This is the same in the aforementioned studies employed CNN [6], [14]. However, the KDD Cup 1999 class datasets may not represent the network activities for today's networks, and solely relying on this class dataset for evaluating may mislead the community. For this reason, we employ a set of different datasets for the comprehensive evaluation, including UNSW-NB15 [10], IDS2017 [11], MAWILab trace [22], as well as NSL-KDD [8], summarized next.

B. Description of datasets

a) **KDD Cup 1999 dataset [9]:** This dataset contains various intrusions simulated in a military network environment collected for 9 weeks. The dataset basically includes the connection information with 41 features plus the associated label, which is classified into five categories of normal, DOS (denial-of-service), R2L (unauthorized access from a remote machine), U2R (unauthorized access to local root privileges), and probing (surveillance and other probing). The dataset contains a considerable number of repeated data, which often resulted in overestimated performance. The NSL-KDD dataset is a refined version that reduces a number of duplicated records [8]. In this dataset, there exist four files: Train+, Train- (a subset of 20% of the training data), Test+, and Test- (a subset of the testing data with a greater difficulty to classify). The fractions of anomalies in the files are well balanced from 47% to 82%.

b) **UNSW-NB15 dataset [10]:** This dataset is collected in 2015 from a simulated environment. The simulated network consists of servers for traffic generations and routers capturing the packets to create pcap files. Three servers were used for the traffic generation: two servers for normal spread of the traffic and the other one for malicious traffic. The Bro-IDS tool[6] is employed to obtain the label information from the collected traffic files. In the dataset, the number of features is 49, along with the label information. The dataset provides a pair of training and testing files, which are smaller than the raw labeled data stored in four CSV files.

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

c) **IDS2017 dataset [11]:** This dataset provides the labeled data for network intrusion detection research. The testbed to collect data includes two networks: one for attacks and the other for victims. The captured dataset contains several different types of attacks including DoS, Web attack, infiltration attack, botnet attack, port scan, as well as background normal traffic that is generated by using the abstract behavior of 25 users based on the HTTP/HTTPS, FTP, SSH, and email protocols. The captured data is five days long, from Monday, July 3, 2017 to Friday July 7, 2017. The first day contains the normal traffic only, while the other days have malicious activities. The number of records is over 2.8 millions with 85 features including the label information.

III. Proposed Encoding Technique

To employ CNN for identifying network anomalies, transforming the one-dimensional connection record into a two-dimensional, image-like data would be essential. Then, the converted 2D data is fed into the CNN for training and testing. In this section, we introduce our new encoding method that converts the 1D connection record to a 2D matrix. Before moving to our method, we first describe normalization and one-hot encoding performed in the data pre-processing stage, and then provide a summary of the gray-scale encoding proposed in [6].

A. Data pre-processing

When using machine learning, there are two types of features, *numerical* and *categorical*. For example, "duration" (indicating connection duration time) is a numerical feature, while "protocol" (indicating application protocols such as HTTP, FTP, etc) would be a categorical feature. The numerical features are normalized to give the identical weight to the independent features. The Min-Max normalization is one of the widely applied technique for normalization, as follows:

$$x_{norm} = \frac{x_{orig} - x_{min}}{x_{max} - x_{min}}$$

Here, x_{orig} is the value given, x_{min} and x_{max} are the minimum and maximum values for that feature across the dataset, and x_{norm} is the normalized value.

To utilize machine learning techniques, a categorical feature is transformed to a set of numerical features and one-hot encoding is one of the broadly used techniques for this purpose. In this paper, we use a term of "sub-categorical features" to refer to the numerical features populated from a categorical feature (using one-hot encoding). In detail, suppose the value space (P) for a categorical feature f_i, $P(f_i) = \{p_1, p_2, ..., p_n\}$. Using one-hot encoding, this single feature is transformed to n sub-categorical features ($f_{i,j}$, and $j = 1..n$), each of which is mapped with a single value in $P(f_j)$. Then, the original value indicates which of the new feature is set. For example, if the value of f_i for a certain connection is p_k, then the new feature $f_{i,k}$ is set to 1 and all the other features $f_{i,j}(j \neq k)$ should be set to a zero. The value of the sub-categorical feature is thus either 0 or 1, while a numeric feature has a real number between 0 and 1 after pre-processing.

Here is an example. For the feature of *proto*, the value space for this feature is {tcp, udp}. The one-hot encoding results in two sub-categorical features of *proto_tcp* and *proto_udp*. If a connection record had 'tcp' for the *proto* feature, it will have *proto_tcp*=1 and *proto_udp*=0 as a result of pre-processing.

B. Gray-scale encoding

The gray-scale encoding (proposed by the past work [6]) maps the features into a binary vector and then transforms the vector into a matrix. The encoding performs as follows. For a numerical feature, the gray-scale encoding converts the (normalized) numeric value to a 10-digit binary number using quantization. The value of the feature determines which digit in the binary number is set to 1 (and all the other digits are set to 0). Recall the value space for a normalized feature, which is from 0.0 to 1.0. The first digit is set to 1 if the feature value is within the range between 0.0 and 0.1, the second digit is set to 1 if the feature value is within 0.1 to 0.2, and so forth. For example, if a feature has a value of 0.213, then it is encoded to '0010000000'. As discussed, a sub-categorical feature created from a categorical feature contains a binary value

The next step is the serialization of the encoded feature values. Since any encoded value should be either 0 or 1, a connection record looks like a binary vector if we concatenate the entire feature values. For example, a connection record in NSL-KDD has 41 features (plus the label). One-hot encoding populates new features by transforming the categorical features to sub-categorical ones, which results in 122 features in total. With the quantization of the numeric features, the number of binary digits for a single record becomes 464, which is equal to the length of the binary vector. Then the binary vector is regarded as a set of gray-scale pixels, each of which is 8 bits long. Hence, there exist $464/8 = 58$ pixels for a single record in NSL-KDD. The authors reorganize the pixels into an $N \times N$ gray-scale image with padding, and $N = 8$ for NSL-KDD (with 6 pixels padded).

C. Our encoding method

In the gray-scale encoding, a numeric feature is encoded to a 10-digit binary number, while a sub-categorical feature (populated from a categorical feature) is encoded to a single digit. This makes that a numeric feature occupies 1.25 pixels but a sub-categorical feature has 0.125 pixel only. With this encoding, hence, the value of a single pixel can be from a single feature or multiple features. In addition, the rough quantization using 10 bins for a numerical feature would be a lossy encoding. For example, 0.500 and 0.599 will be encoded to the identical binary number of '0000010000'.

The basic idea of our method is to allocate the equal number of pixels to individual features (i.e., a single pixel for each feature). To do this, we consider a RGB-like encoding with 24 bits for each pixel, which is assigned for a single feature. The encoding rules are as follows. For a sub-categorical feature, the value should be either 0 or 1, and we encode it to 0x000000

(for '0') or 0xffffff (for '1'). For a numeric feature, the value v should be in a range between 0.0 and 1.0. We calculate a new value $v' = v \times (2^{24} - 1)$ for that feature. The encoded value will be ranged between 0 and $2^{24} - 1$, which is then accommodated within a single 24-bit pixel.

With our encoding scheme, the transformation to an RGB image is straightforward since each feature is mapped to an independent pixel. For NSL-KDD, there exist 122 features to encode, which can be transformed to a (12 $\times$ 12) with some padding pixels at the end. As described, the size of the resulted matrix using the gray-scale encoding is (8 $\times$ 8) with padding.

The number of pixels after encoding is deterministic for both encoding techniques. Assuming N is the number of numeric features and M is the number of sub-categorical features, the gray-scale encoding results in $(10 \times N + M)/8$ pixels and the number of pixels for our encoding method is $N + M$ pixels.

IV. Evaluation

In this section, we evaluate our encoding technique by comparing the performance with the gray-scale encoding. We implemented the code on top of TensorFlow, and employed the latest version of GooLeNet (Inception V3 model), with the default parameter setting (i.e., epoch=4000, learning rate=0.01, and batch size=100). Additionally we used a gradient descent optimizer as an optimizer and cross entropy for the cost function.

We mainly utilize accuracy and F1-score (or F-measure) to analyze the performance of the encoding techniques, which are widely employed to measure the quality of classification methods. By convention, accuracy is the fraction of the data points (or records) in testing that were correctly classified and defined $accuracy = \frac{TP+TN}{TP+TN+FP+FN}$, where TP stands for true positive (the number of anomalies correctly classified), TN for true negative (the number of normal correctly classified), FP for false positive (the number of normal wrongly classified into anomalies), and FN for false negative (the number of anomalies incorrectly classified to normal). F1-score is the weighted harmonic mean of precision and recall of the test, and defined $F1 - score = 2 \cdot \frac{precision \cdot recall}{precision + recall}$, where $precision = \frac{TP}{TP+FP}$ and $recall = \frac{TP}{TP+FN}$.

We first conduct experiments with the NSL-KDD dataset used in the past study. Then, we compare the performance of the encoding techniques with the other datasets.

A. Experimental results against NSL-KDD data

The NSL-KDD dataset defines 41 features in three feature groups plus the associate label. As discussed in Section III, the number of features is 122 in total by one-hot encoding. By encoding, the gray-scale encoding converts the vector with 122 elements to a (8 $\times$ 8) matrix, while our encoding transforms it to a (12 $\times$ 12) matrix.

Figure 1 compares the encoding techniques with respect to train and validation accuracy. In NSL-KDD, there are two training files of "Train+" and "Train-" (and two testing files of "Test+" and "Test'", as described in Section II-B). From the figure, for both of the training files, we can see that our method slightly works better than the gray-scale encoding technique.

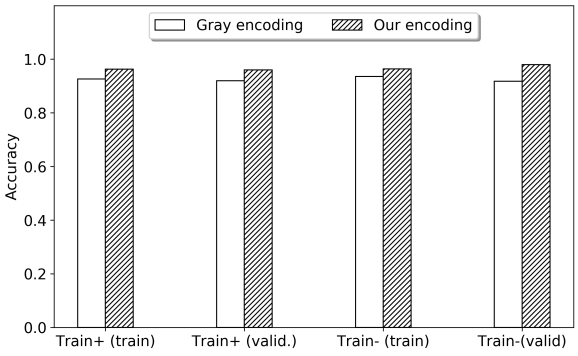

Fig. 1. Training and validation accuracy for NSL-KDD: Our encoding method yields slightly better performance than the gray-scale encoding for the accuracy of training and validation.

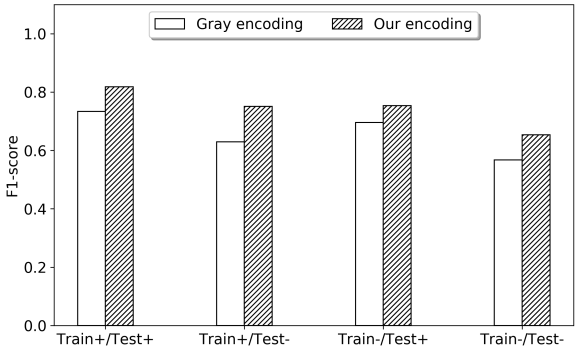

Fig. 2. Classification performance against the NSL-KDD dataset: The proposed method consistently outperforms the gray-scale encoding, with the 4%–12% improvement in F1-score.

Figure 2 shows the actual classification performance using the training and testing datasets. With two files for training and testing respectively, there exist four combinations for training and testing as can be seen from the figure. The figure shows that our proposed method consistently outperforms, with the 4%–12% improvement in F1-score.

B. Experimental results against UNSW-NB15 data

The UNSW-NB15 dataset includes 49 features in five groups (flow, basic, content, time, and additional feature groups), and the last two features are for the attack category and label. The IP addresses and port numbers are not actually included in the data files. Two features are timestamps for recording the starting and ending times and we excluded them from learning. Finally, we utilized 41 features for encoding, excluding the IP addresses and port numbers, timestamps, and the label and attack category.

With three categorical features, and the number of features extended becomes 185 in total after pre-processing. For learning and testing, the gray-scale encoding makes a set of two dimensional arrays with the size of (8 $\times$ 8) as input to CNN.

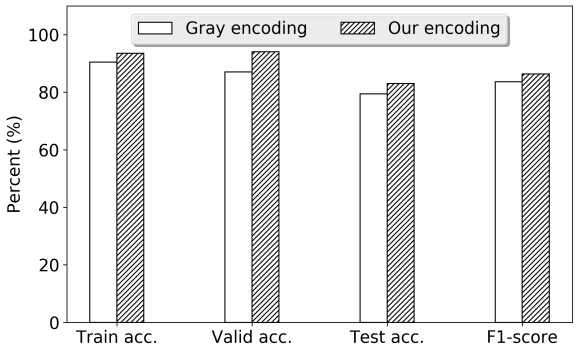

Fig. 3. Classification performance against the UNSW-NB15 dataset: Our encoding technique yields slightly better results than the gray-scale encoding for testing accuracy as well as training and validation accuracy (with up to 7% improvement).

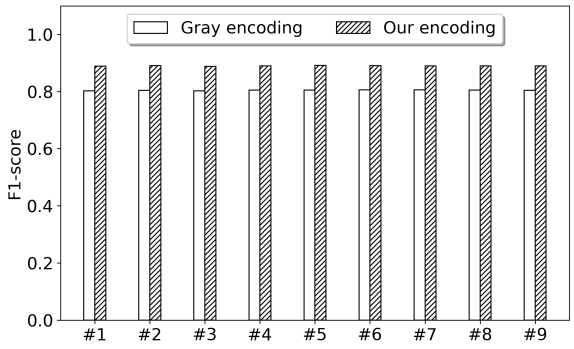

Fig. 4. Classification performance against the IDS2017 dataset: The gray-scale encoding technique yields around 82% accuracy, while our encoding technique shows 88%–89% accuracy in F1-score over the testing files.

Our method encodes the features for a single data point in a (14 $\times$ 14) matrix. Train = 82K, testing=175K.

Figure 3 shows the train, validation, and testing accuracy. For the accuracy of training and validation, our technique slightly works better than the gray-scale encoding technique. Our encoding further improves the performance up to 7%, as can be seen from the figure.

C. Experimental results against IDS2017 data

This dataset contains a 5-day log (from Monday to Friday), and the number of data points is approximately 3.3 millions. While the first day log contains normal activity only, the other days contain the data points for various attacks including DoS, Web attacks, port scan, etc. In our experiment, we created 10 files, each of which contains 100K data points randomly chosen from the entire dataset. Out of the ten files, one file was used for training and the other nine files were used for testing to measure the classification performance.

The number of features is 77 in total, with the label information. As the entire features are the numerical type, the gray-scale encoding constructs (10 $\times$ 10) matrices as input to CNN. Our technique produces a set of (9 $\times$ 9) matrices after encoding.

Figure 4 shows the testing result. While the gray-scale encoding yields around 82% accuracy, our encoding technique shows 88%–89% accuracy in F1-score over the testing files (#1–#9). We also measured the accuracy for training and validation, which are 81.3% and 84.0%, respectively for gray-scale and 89.3% and 82.0% for our proposed technique.

In sum, the experimental results show that our method consistently outperforms the gray-scale encoding over the entire datasets. We also analyze the datasets using random forest as the baseline, which is one of the shallow (non-deep) machine learning techniques based on an ensemble learning method. Random forest is widely used since it produces relatively accurate results compared to other shallow learning techniques. Note that random forest does not rely on the CNN structure, and hence, it is not needed to transform the vector data to a matrix format. Our observation is that the GoogLeNet model with our encoding works better for NSL-KDD, but random forest produces better results for the other two datasets. Figure 5 shows the classification performance when using random forest, along with the CNN with the two encoding techniques. In the figure, IDS2017 shows the classification performance on average against the IDS2017 dataset.

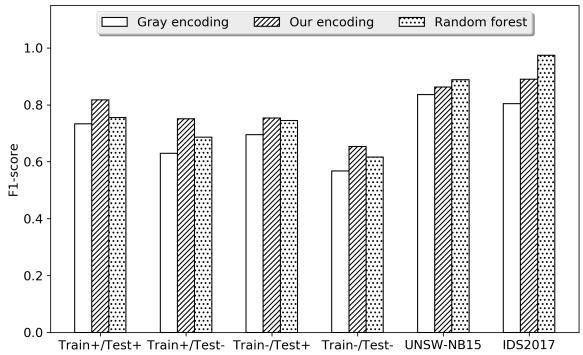

Fig. 5. Comparison of classification performance with random forest: CNN with our encoding works better for NSL-KDD but random forest outperforms for UNSW-NB15 and IDS2017. Note that IDS2017 shows the classification performance on average against the IDS2017 dataset.

V. Conclusions

In this paper, we introduced a new encoding technique that enhances the performance for the identification of anomalous events using a CNN structure. To evaluate, we employed three different datasets of NSL-KDD, UNSW-NB15, and IDS2017. From the evaluation, we observed that our encoding technique consistently outperforms the gray-scale encoding proposed in the past study over the entire datasets. From the comparison study with random forest, we observed that the CNN-based anomaly detection with our encoding sometimes works better but sometimes not, which signals that a degree of optimizations need to be investigated in the future.

One of the planned future studies is to examine other CNN structures, such as ResNet and VGGNet, with our encoding method to compare the performance with GoogLeNet. In addition, there would be a plenty of room to optimize CNN structures for our application, and we plan to investigate to see whether the customized design of the CNN structure would be beneficial for network anomaly detection with respect to improve the identification performance.

ACKNOWLEDGMENT

This work was supported in part by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No.2016-0-00078, Cloud-based Security Intelligence Technology Development for the Customized Security Service Provisioning).

REFERENCES

[1] R. K. Malaiya, D. Kwon, J. Kim, S. C. Suh, H. Kim, and I. Kim, "An empirical evaluation of deep learning for network anomaly detection," in *2018 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2018, pp. 893–898.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'12, 2012, pp. 1097–1105.

[3] Q. Zhu, X. Li, A. Conesa, and C. Pereira, "Gram-cnn: a deep learning approach with local context for named entity recognition in biomedical text," *Bioinformatics*, vol. 34, no. 9, pp. 1547–1554, 2017.

[4] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *Information Networking (ICOIN), 2017 International Conference on*. IEEE, 2017, pp. 712–717.

[5] S. Zhou, W. Shen, D. Zeng, M. Fang, Y. Wei, and Z. Zhang, "Spatial–temporal convolutional neural networks for anomaly detection and localization in crowded scenes," *Signal Processing: Image Communication*, vol. 47, pp. 358–368, 2016.

[6] Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, "Intrusion detection using convolutional neural networks for representation learning," in *International Conference on Neural Information Processing*. Springer, 2017, pp. 858–866.

[7] D. Kwon, K. Natarajan, S. C. Suh, H. Kim, and J. Kim, "An empirical study on network anomaly detection using convolutional neural networks," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2018, pp. 1595–1598.

[8] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications*, ser. CISDA'09, 2009, pp. 53–58.

[9] "KDD Cup 1999 Data," http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[10] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *Military Communications and Information Systems Conference (MilCIS), 2015*. IEEE, 2015, pp. 1–6.

[11] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." in *ICISSP*, 2018, pp. 108–116.

[12] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017.

[13] S.-N. Nguyen, V.-Q. Nguyen, J. Choi, and K. Kim, "Design and implementation of intrusion detection system using convolutional neural network for dos detection," in *Proceedings of the 2Nd International Conference on Machine Learning and Soft Computing*, ser. ICMLSC '18. ACM, 2018, pp. 34–38.

[14] R. Vinayakumar, K. Soman, and P. Poornachandran, "Applying convolutional neural network for network intrusion detection," in *Advances in Computing, Communications and Informatics (ICACCI), 2017 International Conference on*. IEEE, 2017, pp. 1222–1228.

[15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[16] Z. Zhong, L. Jin, and Z. Xie, "High performance offline handwritten chinese character recognition using googlenet and directional feature maps," in *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*. IEEE, 2015, pp. 846–850.

[17] S.-G. Lee, Y. Sung, Y.-G. Kim, and E.-Y. Cha, "Variations of alexnet and googlenet to improve korean character recognition performance." *Journal of Information Processing Systems*, vol. 14, no. 1, 2018.

[18] R. U. Khan, X. Zhang, and R. Kumar, "Analysis of resnet and googlenet models for malware detection," *Journal of Computer Virology and Hacking Techniques*, Aug 2018. [Online]. Available: https://doi.org/10.1007/s11416-018-0324-z

[19] 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 *Wireless Networks and Mobile Communications (WINCOM), 2016 International Conference on*. IEEE, 2016, pp. 258–263.

[20] S. Zhai, Y. Cheng, W. Lu, and Z. Zhang, "Deep structured energy based models for anomaly detection," in *International Conference on Machine Learning*, 2016, pp. 1100–1109.

[21] S. Potluri and C. Diedrich, "Accelerated deep neural networks for enhanced intrusion detection system," in *Emerging Technologies and Factory Automation (ETFA), 2016 IEEE 21st International Conference on*. IEEE, 2016, pp. 1–8.

[22] R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda, "Mawilab: Combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking," in *Proceedings of the 6th International COnference*, ser. Co-NEXT '10, 2010, pp. 8:1–8:12.