Network Intrusion Detection with Improved Feature Representation

Geonsu Lee*, Hochang Rhee*, Jae Hoon Shim*, Hyung Il Koo† and Nam Ik Cho*
* Department of ECE, INMC, Seoul National University, Seoul, Republic of Korea
† Ajou University, Gyeonggi-do, Republic of Korea
E-mail: drt4917@ispl.snu.ac.kr, hochang@ispl.snu.ac.kr, joinjhs@ispl.snu.ac.kr, hikoo@ajou.ac.kr, nicho@snu.ac.kr

***Abstract*—This paper presents new feature representation methods for the network intrusion detection. Like conventional work, our method is based on neural networks. However, rather than focusing on the network architecture search, we improve the performance by developing effective feature representation methods. First, we apply the embedding method to categorical features in the network data. Although embedding has been commonly used in natural language processing and recommendation systems, categorical features in network problems were often ignored or simply used in a one-hot-encoding vector form. By applying the embedding method to categorical features, we can effectively exploit them. Second, we apply a robust scaler to numerical features. Numerical features are concentrated in a few clusters with a small portion of outliers, and we can effectively remove outliers with a robust scaler. Finally, we augment feature vectors with categorical representations of numerical values. These categorical representations (e.g., high, medium, and low) help to discover simple logical rules and facilitate the intrusion detection. We have applied our method to two scenarios: a normal/attack classification and an unsupervised learning of anomaly detection. Experimental results have shown that the proposed method outperforms conventional methods on public benchmark datasets.**

***Index Terms*—Network intrusion detection, Feature representation, Feature embedding**

I. Introduction

With the advancement of cloud computing and the Internet of Things (IoT), the volume and usage of communication networks have been greatly increased in recent decades. Also, securing network from malicious network intrusions have become ever more important. In order to address the security issues, numerous intrusion detection methods were proposed based on classical machine learning techniques, such as support vector machine (SVM) [1] and random forest (RF) [2]. However, like many other research fields, these classical methods have been replaced with deep neural networks: Numerous methods were proposed based on fully connected networks (FCNs), convolutional neural networks (CNNs), and recurrent neural networks (RNNs). Although they outperformed classical methods [3], we still believe there is room for performance improvement, since most researchers focused on the design of neural network architectures.

In this paper, we focus on feature representation methods and propose new feature pre-processing and embedding methods for network intrusion detection. Network data consist of categorical (e.g., transaction protocol, type of service), numerical (e.g., duration, transmitted data size), and nominal (e.g., IP address, port number) features and these features are used to detect suspicious activities [4]. In conventional methods, numerical features are directly fed into neural networks and the categorical features are either used in the form of a one-hot-encoding or ignored (probably due to its high dimensionality in one-hot representation). Nominal features are seldom used: IP addresses are randomly replaced due to privacy issues, and port numbers are also likely to be chosen randomly.

Although we agree that nominal features do not carry information for suspicious activity detection, we believe that the usage of numerical and categorical features can be improved. First, in order to effectively exploit the information of categorical features, we apply feature embedding to categorical features. For numerical features, we have found that most of these values are concentrated in compact intervals with a small portion of outliers. Therefore, rather than using raw values, we apply the robust scalers to numerical features. Finally, we also represent numerical values into categorical features (i.e., high, medium, and low) and concatenate them to feature vectors. We have applied the proposed method to the normal/attack classification and the unsupervised learning of anomaly detection, showing notable improvements on Kyoto Honeypot [5], UNSW-NB15 [6], and CICIDS-2018 [7] datasets.

The rest of this paper is organized as follows. Section II briefly reviews the related works about various deep learning methods in network intrusion detection and explains network intrusion datasets. In Section III, we present the details of our feature representation methods. In Section IV, we evaluate our methods on three public datasets and discuss the results. Finally, we conclude this paper in Section V.

II. Related Work

Although numerous methods have been proposed to address the network intrusion detection problem, most of them focused on network architectures and training methods. In this section, we first review deep learning based methods and present the characteristics of public datasets.

A. Deep Learning Methods in Network Intrusion Detection

The network intrusion detection task is a classification problem that determines whether a given session corresponds to normal (negative samples) or attack (anomaly or positive samples) behavior. As deep learning methods have shown the state of the art performance in many tasks such as image classification, speech recognition, machine translation, many researchers also attempted to apply deep learning to the network intrusion detection problem. Specifically, deep learning structures like FCN [8], [9], CNN [10], [11], RNN [12], and long short-term memory (LSTM) [13], [14] were adopted for the network intrusion detection.

Also, autoencoder (AE) and generative adversarial network (GAN) were adopted in [15]–[17], and techniques like transfer learning were explored in [18], [19]. Even though there have been a variety of attempts, the researchers basically focused on the adoption of existing network architectures for the problem, not on the feature representation.

B. Network Intrusion Dataset

The network intrusion detection problem has a long history, and numerous datasets have been proposed for decades. Although old datasets like KDD CUP 99 and NSL-KDD [20] are still available, we believe they have limitations in reflecting the characteristics of modern network traffic and attacks. Therefore, we choose recent datasets as our targets: Kyoto Honeypot [5], UNSW-NB15 [6], and CICIDS-2018 [7].

The Kyoto Honeypot [5] dataset is built with actual network traffic data and has a total of 9.35 million session data, in which 46.49% is attack data. The Kyoto Honeypot dataset does not provide the type of attacks. The number of features for each session is 24, and we have to detect network intrusion from the feature vector. There are 14 numerical features, two categorical features, one label, three label-related features, and 4 IP address and port number features. In our experiments, we use numerical features, categorical features, and a label. Among a huge number of session data, we randomly select 100,000 samples for training and 20,000 samples for testing. Normal and attack ratio of our training and testing set is set to 50:50.

The UNSW-NB15 [6] is a synthetic dataset that includes Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms attacks. It consists of 2.54 million session data, where 12.65% is attack data. The number of features is 49. There are 38 numerical features, five categorical features, two labels, and four IP address/port number features. In our experiments, we use numerical features, categorical features, and labels. Instead of using the entire set, we use the training and testing set provided by [6]: The training set includes 175,341 samples, and the test set includes 82,332 samples. The normal and attack ratio of the training set is 32:68, and the ratio in the test is 45:55.

The CICIDS-2018 [7] is also a synthetic dataset that includes Bruteforce, Dos, Web, Infiltration, Botnet, DDoS, and PortScan attacks. It consists of 9.33 million session data, where 23.17% is attack data. Since CICIDS-2018 provides raw traffic data, we use Argus to extract 21 features. Extracted features are listed in Table I. We randomly select 100,000 samples for training and 20,000 samples for test. The normal and attack ratio of our training and test set is 50:50.

TABLE I: Extracted features from CICIDS-2018 using Argus.

Feature name	Type	# of categories
proto	Categorical	10
dur	Numerical	-
spkts	Numerical	-
dpkts	Numerical	-
sbytes	Numerical	-
dbytes	Numerical	-
sttl	Numerical	-
dttl	Numerical	-
sload	Numerical	-
dload	Numerical	-
sloss	Numerical	-
dloss	Numerical	-
swin	Numerical	-
stcpb	Numerical	-
dtcpb	Numerical	-
dwin	Numerical	-
tcprtt	Numerical	-
synack	Numerical	-
ackdat	Numerical	-
smean	Numerical	-
dmean	Numerical	-

III. Proposed Method

In this section, we present our feature representation methods for the network intrusion detection. Our methods are independent of neural network structures, and we apply our method to three popular network architectures in the experimental section.

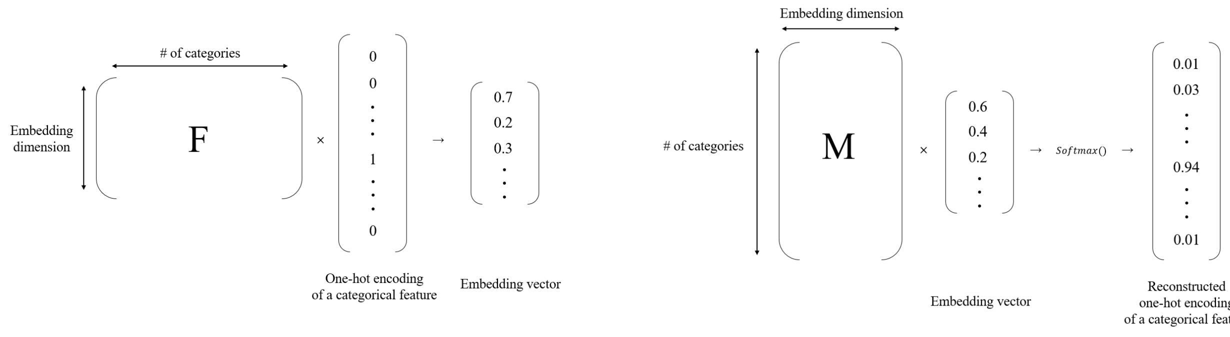
A. Categorical Feature Embedding

When categorical features are used in the form of one-hot vectors, the dimension of the neural network inputs gets high, and it is likely to introduce difficulties in training. Actually, researchers usually focused on numerical features and discarded categorical features in the literature.

In order to exploit categorical features, we represent categorical features in an effective form. To be precise, similar to [21], we insert an embedding layer (embedding matrix) to convert a one-hot vector to the input of neural networks as shown in Fig. 1-(a). For each categorical feature, the dimension of the embedding vector (e_{dim}) is determined by

$$e_{dim} = \text{round}\left(\#(category)^{\alpha}\right) \tag{1}$$

where $\#(category)$ denotes the number of categories for each categorical feature, α denotes a parameter, and round$(\cdot)$ denotes a rounding function. We set α to 0.25 in our experiments according to [22]. These embedding tables are updated during network training. Unlike network parameters that reflect the relationship of multiple features, the embedding tables are affected only by a single feature. In autoencoder training [15], the inverse process of embedding is required, and it is illustrated in Fig. 1-(b). As shown, through matrix multiplication and $softmax(\cdot)$, the embedding vector is restored to categorical representation.



(a) One-hot to numerical conversion (b) Numerical to one-hot conversion

Fig. 1: Categorical feature embedding structure.

B. Robust Scaler

Since diverse dynamic ranges can result in imbalanced gradients, a proper normalization scheme is essential for effective training. We have found that most numerical features fall in compact regions with a small number of outliers. Therefore, the commonly used minmax scaler, which linearly maps the max value to 1 and the min value to 0, is not suitable. Rather, we use a robust scaler, that maps the top $l\%$ value to 1 and bottom $l\%$ value to 0, reducing the effects of outliers:

$$v_{norm} = \frac{v}{v_{top\_l\%} - v_{bottom\_l\%}} \tag{2}$$

We set l to 0.5 in this paper.

C. Categorical Features from Numerical Features

Deep learning models achieve generalization power by learning features from data, which have been a very difficult task with ad-hoc manners. On the other hand, deep learning models are known for having difficulties in discovering and exploiting simple logical relations, e.g., AND. To alleviate this problem, researchers add wide features to memorize low-level feature interactions along with deep features [23].

We analyze network data and find out that network data features also have simple relations between them and corresponding labels: Normal and attack samples are clustered as shown in Fig. 2, and their groups can be described with simple logical relations (e.g., small "ct_dst_ltm" and small "ct_src_dport_ltm" $\rightarrow$ normal). In order to exploit these properties, we apply the categorization to numerical features: After applying the robust scaler described in Section III.B, we partition each normalized feature into k bins according to its value and treat them as categorical features. Partition boundaries can be determined in a variety of ways, but we set boundaries linearly for simplicity. We set k as 3 for our experiments. These newly generated categorical features are concatenated to original categorical features and embedded just like original categorical features in Section III.A.

IV. Experimental Results

For the evaluation of the proposed method, we apply our method to two tasks. One is the supervised learning task that determines whether a given session data is a normal class or an attack class. The other is the anomaly detection task, which learns anomaly detection rules only from normal data.

A. Network Structures

For the supervised learning task, we first evaluate our method on FCN. The FCN consists of 3 hidden layers with 256, 128, 64 nodes and uses ReLU as an activation function. We use the cross-entropy as a loss function. For the baseline, we only use numerical features. Then, the proposed methods are incrementally applied. Categorical features (additional categorical features as well as original categorical features) are incorporated through embedding, and this increases the input dimensionality. However, due to the embedding, the increase of dimensionality is moderate.

Inspired by the wide and deep network in the recommendation system [23], we also apply the proposed method to this new architecture. Wide-deep networks use one-hot categorical features and cross-product-transformed categorical features, where these transformed categorical features are generated by applying AND operations to categorical features [23]. As discussed in Section III-C, network data show simple relations between features and labels. Therefore, we can benefit from both memorization and generalization with the wide and deep structure. Our baseline contains categorical feature embedding and two steps of the proposed method are applied, i.e., robust scaler and additional categorical features.

For the anomaly detection task, we evaluate our methods on a memory-guided autoencoder network from [24]. This structure attaches memory modules to the autoencoder structure in order to capture various normal features by clustering them into multiple points. Since [24] deals with image and video data, we change the network structure for the network anomaly detection. Unlike images or videos that usually show abnormalities in sub-regions, network data features should be considered as a whole. Therefore, the number of queries in our modified network structure is fixed to one. Encoder and decoder are consist of 4 hidden layers with 128, 64, 32, 16 nodes and use ReLU as an activation function. We

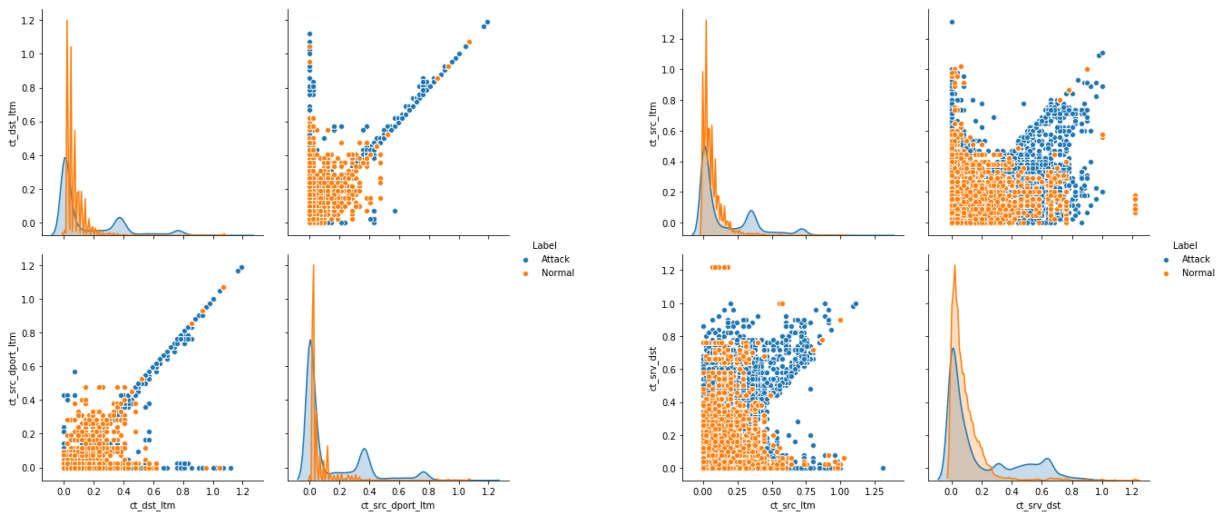


Fig. 2: Histograms and scatter plots of selected numerical features in UNSW-NB15 dataset. We observe that normal and attack samples are clustered.

use reconstruction loss, feature compactness loss, and feature separateness loss in [24].

B. Evaluation Metrics

In network intrusion detection, commonly used metrics are accuracy, precision, recall, F1-score, and false alarm rate (FAR). These metrics are defined as

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{3}$$

$$Precision = \frac{TP}{TP + FP}, \tag{4}$$

$$Recall = \frac{TP}{TP + FN}, \tag{5}$$

$$F1\text{-}score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}, \tag{6}$$

and

$$FAR = \frac{FP}{FP + TN}, \tag{7}$$

where TP, TN, FP, and FN denote true positive, true negative, false positive, and false negative, respectively. There is a trade-off between recall and FAR, depending on the threshold determining normal or attack. Also, unlike the experimental setting where the normal and attack samples are almost 50:50, the ratio of normal samples is often dominant in the real-world situation. In this case, the values of the above metrics vary according to the ratio of normal and attack samples. Therefore we choose the area under the receiver operating characteristic (AUROC), which is not affected by thresholds or normal and attack samples ratio, as the main metric to evaluate the performance of target networks. The receiver operating characteristic (ROC) curve, which has FAR in the horizontal axis and recall in the vertical axis, is obtained by changing the threshold, and AUROC can be calculated by measuring the area under the ROC curve.

We use the anomaly score to evaluate the performance of anomaly detection, which includes reconstruction loss, feature compactness loss, and feature separateness loss. Unlike supervised intrusion classification problems, we need to determine the threshold suitable for the situation. Therefore, accuracy, precision, recall, f1-score, and FAR are not evaluated; they vary depending on the threshold. Instead, we use AUROC for the anomaly detection task.

C. Evaluation

We compare the performance of the proposed method with the baselines. Experimental results for FCN, wide and deep network, and memory-guided network are summarized in Tables II, III, and IV, respectively. As shown, in most cases, using all of our proposed methods yields the best performance. Note that the wide and deep network uses categorical feature embedding for the deep network part [23].

CICIDS-2018 dataset is relatively easier than the other two sets, and thus all metrics are almost saturated in the case of FCN. The wide-and-deep network performs worse than the FCN due to its complex structure, and the improvement from the baseline is relatively small. In the case of the memory-guided network, using additional categorical features from numerical features gives a slightly lower performance on UNSW-NB15 and CICIDS-2018 datasets. We believe that the memory-guided network uses the reconstruction loss, which needs to reconstruct original inputs, and additional features lead to a performance drop. Nevertheless, the performance drop is small, and our method still outperforms the original networks. Finally, we would like to emphasize that the proposed method outperforms all baselines on Kyoto Honeypot [5], which consists of real data.

TABLE II: FCN results on Kyoto Honeypot, UNSW-NB15, and CICIDS-2018 dataset.

		Accuracy	Precision	Recall	F1-score	FAR	AUROC
Kyoto Honeypot	FCN	0.8205	0.8064	0.8433	0.8245	0.2024	0.8819
	+ Robust scaler	0.8687	0.8376	0.9147	0.8745	0.1774	0.9437
	+ Categorical feature embedding	0.8758	0.8478	**0.9159**	0.8805	0.1644	0.9536
	+ Additional categorical features from numerical features	**0.8870**	**0.8694**	0.9108	**0.8896**	**0.1369**	**0.9544**
UNSW-NB15	FCN	0.8507	0.7999	0.9720	0.8776	0.2979	0.9596
	+ Robust scaler	0.8596	0.8158	0.9622	0.8830	0.2662	0.9650
	+ Categorical feature embedding	0.8733	0.8306	0.9670	0.8936	0.2416	0.9758
	+ Additional categorical features from numerical features	**0.8896**	**0.8492**	**0.9721**	**0.9065**	**0.2115**	**0.9776**
CICIDS-2018	FCN	0.9955	0.9910	**1.0000**	0.9955	0.0091	0.9999
	+ Robust scaler	0.9977	0.9959	0.9994	0.9977	0.0041	**1.0000**
	+ Categorical feature embedding	0.9982	0.9963	**1.0000**	0.9982	0.0037	**1.0000**
	+ Additional categorical features from numerical features	**0.9992**	**0.9986**	0.9997	**0.9992**	**0.0014**	**1.0000**

TABLE III: Wide and deep results on Kyoto Honeypot, UNSW-NB15, and CICIDS-2018 dataset.

		Accuracy	Precision	Recall	F1-score	FAR	AUROC
Kyoto Honeypot	Wide and deep (including categorical feature embedding)	0.8569	0.8604	0.8521	0.8562	0.1382	0.9176
	+ Robust scaler	0.8853	0.8686	**0.9080**	0.8878	0.1374	0.9544
	+ Additional categorical features from numerical features	**0.8910**	**0.8818**	0.9030	**0.8923**	**0.1210**	**0.9556**
UNSW-NB15	Wide and deep (including categorical feature embedding)	0.8649	0.8188	0.9691	0.8876	0.2627	0.9775
	+ Robust scaler	0.8783	0.8336	0.9731	0.8980	0.2379	0.9796
	+ Additional categorical features from numerical features	**0.8822**	**0.8356**	**0.9786**	**0.9015**	**0.2358**	**0.9799**
CICIDS-2018	Wide and deep (including categorical feature embedding)	0.9438	0.9761	0.9107	0.9423	0.0226	0.9676
	+ Robust scaler	**0.9457**	**0.9786**	0.9121	**0.9442**	**0.0202**	**0.9716**
	+ Additional categorical features from numerical features	0.9453	0.9773	**0.9125**	0.9438	0.0215	0.9688

TABLE IV: Memory guided network results on Kyoto Honeypot, UNSW-NB15, and CICIDS-2018 dataset.

		AUROC
Kyoto Honeypot	Memory guided network	0.8129
	+ Robust scaler	0.8301
	+ Categorical feature embedding	0.8730
	+ Additional categorical features from numerical features	**0.8790**
UNSW-NB15	Memory guided network	0.8696
	+ Robust scaler	0.8819
	+ Categorical feature embedding	**0.9148**
	+ Additional categorical features from numerical features	0.9125
CICIDS-2018	Memory guided network	0.9698
	+ Robust scaler	0.9723
	+ Categorical feature embedding	**0.9803**
	+ Additional categorical features from numerical features	0.9761

V. Conclusions

In this paper, we have proposed new feature representation methods for network intrusion and anomaly detection using neural networks. By adding a categorical feature embedding layer, representing numerical features categorically, and using a robust scaler, we have improved intrusion detection rates on supervised learning and anomaly detection. Unlike previous work, the proposed method focuses on feature representation and our methods can be applied to a range of deep learning methods.

Acknowledgment

This research was supported by Samsung Electronics Co., Ltd.

References

[1] D. S. Kim, H.-N. Nguyen, and J. S. Park, "Genetic algorithm to improve svm based network intrusion detection system," in *19th International Conference on Advanced Information Networking and Applications (AINA'05) Volume 1 (AINA papers)*, vol. 2, pp. 155–158, IEEE, 2005.

[2] N. Farnaaz and M. Jabbar, "Random forest modeling for network intrusion detection system," *Procedia Computer Science*, vol. 89, pp. 213–217, 2016.

[3] S. Gamage and J. Samarabandu, "Deep learning methods in network intrusion detection: A survey and an objective comparison," *Journal of Network and Computer Applications*, vol. 169, p. 102767, 2020.

[4] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security*, vol. 86, pp. 147–167, 2019.

[5] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, and K. Nakao, "Statistical analysis of honeypot data and building of kyoto 2006+ dataset for nids evaluation," in *Proceedings of the first workshop on building analysis datasets and gathering experience returns for security*, pp. 29–36, 2011.

[6] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 military communications and information systems conference (MilCIS)*, pp. 1–6, IEEE, 2015.

[7] A. Thakkar and R. Lohiya, "A review of the advancement in intrusion detection datasets," *Procedia Computer Science*, vol. 167, pp. 636–645, 2020.

[8] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.

[9] Z. Wang, "Deep learning-based intrusion detection with adversaries," *IEEE Access*, vol. 6, pp. 38367–38384, 2018.

[10] Y. Zeng, H. Gu, W. Wei, and Y. Guo, "$deep-full-range$: A deep learning based network encrypted traffic classification and intrusion detection framework," *IEEE Access*, vol. 7, pp. 45182–45190, 2019.

[11] 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)*, pp. 1595–1598, IEEE, 2018.

[12] 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.

[13] A. Diro and N. Chilamkurti, "Leveraging lstm networks for attack detection in fog-to-things communications," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 124–130, 2018.

[14] B. J. Radford, L. M. Apolonio, A. J. Trias, and J. A. Simpson, "Network traffic anomaly detection using recurrent neural networks," *arXiv preprint arXiv:1803.10769*, 2018.

[15] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE transactions on emerging topics in computational intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

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

[17] H. Zenati, M. Romain, C.-S. Foo, B. Lecouat, and V. Chandrasekhar, "Adversarially learned anomaly detection," in *2018 IEEE International conference on data mining (ICDM)*, pp. 727–736, IEEE, 2018.

[18] C. Kneale and K. Sadeghi, "Semisupervised adversarial neural networks for cyber security transfer learning," *arXiv preprint arXiv:1907.11129*, 2019.

[19] J. Zhao, Z. Wu, Y. Wang, and X. Ma, "Adaptive optimization of qos constraint transmission capacity of vanet," *Vehicular Communications*, vol. 17, pp. 1–9, 2019.

[20] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*, pp. 1–6, IEEE, 2009.

[21] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[22] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[23] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, *et al.*, "Wide & deep learning for recommender systems," in *Proceedings of the 1st workshop on deep learning for recommender systems*, pp. 7–10, 2016.

[24] H. Park, J. Noh, and B. Ham, "Learning memory-guided normality for anomaly detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14372–14381, 2020.