International Journal of Advanced Smart Convergence Vol.11 No.4 10-19 (2022)
http://dx.doi.org/10.7236/IJASC.2022.11.4.10

IJASC 22-4-2

Evaluating Unsupervised Deep Learning Models for Network Intrusion Detection Using Real Security Event Data

Jiho Jang[1*], Dongjun Lim[1*], Changmin Seong[2*], JongHun Lee[3], Jong-Geun Park[3] and Yun-Gyung Cheong[4*]

[1] *B.A., Department of Computer Software, Sungkyunkwan University, Korea*
[2] *M.A., Department of Computer Software, Sungkyunkwan University, Korea*
[3] *Senior Researcher, Electronics and Telecommunications Research Institute, Korea*
[4] *Professor, Department of Artificial Intelligence, Sungkyunkwan University, Korea*
zyoa@skku.edu, flamecracker1220@gmail.com, tjdckdals@skku.edu, mine@etri.re.kr, queue@etri.re.kr, aimecca@skku.edu

Abstract

AI-based Network Intrusion Detection Systems (AI-NIDS) detect network attacks using machine learning and deep learning models. Recently, unsupervised AI-NIDS methods are getting more attention since there is no need for labeling, which is crucial for building practical NIDS systems. This paper aims to test the impact of designing autoencoder models that can be applied to unsupervised an AI-NIDS in real network systems. We collected security events of legacy network security system and carried out an experiment. We report the results and discuss the findings.

Keywords: *NIDS, Deep Learning, Autoencoder*

1. Introduction

As network technology develops, the importance of network security technology has also increased. Traditional network security authentication using public key certificates can verify network traffic originated from trusted users. However, it has limitation in that it cannot detect and block abnormal network traffic generated from the hosts compromised by attackers. To overcome this issue, a Network Intrusion Detection System (NIDS) that can find out network intrusion by monitoring network traffic is used [1]. NIDS can be implemented in two ways: signature-based approach and anomaly detection based approach [6]. While signature-based NIDS employ rules to check whether a given input is an attack or not [7], anomaly detection based NIDS determines attacks relying on how much the input deviates from the majority of the data [8]. Therefore, signature-based NIDS is suitable for capturing well-known attacks, while anomaly based NIDS is suitable for capturing unknown attacks [6].

In the past, classical machine learning models such as K-Nearest Neighbors (KNN), Support Vector Machine (SVM), DBSCAN were widely used for anomaly detection [2, 10–12]. For instance, KNN can detect

Manuscript Received: September. 15, 2022 / Revised: September. 17, 2022 / Accepted: September. 19, 2022
Corresponding Author: aimecca@skku.edu
Tel: +82-31-299-4393
Professor, Department of AI, Sungkyunkwan University, Korea

outliers based on their distance to the k nearest neighbors, and One Class SVM (OC-SVM) can be used for anomaly detection [24]. Recently, deep learning models such as Long Short-Term Memory, Autoencoder, and Convolutional Neural Network have been much studied as they can learn features on their own, enhancing the performance of NIDS [9, 13, 5, 14, 24].

However, these supervised AI-NIDS have drawbacks. First, they require labeled data, which takes time and cost to label the data for training supervised machine learning models. Moreover, new attacks are being developed every day, and as a result, intrusion patterns are changing rapidly [3]. Supervised learning models can show poor performance in detecting such new attacks [4]. To handle this problem, unsupervised machine learning models are employed. Song et al. [5] reports that NIDS using autoencoder can achieve good performance on network benchmark datasets such as NSL-KDD and IoT datasets. The study suggests experimenting different latent sizes for finding the optimal autoencoder model. Nonetheless, it is not confirmed yet if this finding can be applied to various situations.

Therefore, we carried out experiments using three different autoencoder structures and security events of legacy network security system. This paper reports the results and discuss how the model structure impacts the NIDS performance.

2. Unsupervised Learning Models

There are different unsupervised deep learning models. GANs learn to classify fake data generated by the generator and existing real data using a discriminator [15]. Variations of GANs are developed to deal with image generation problems. StyleGANs redesign the architecture of the generator to be style-based, producing better quality images from the latent code [16]. CycleGANs perform unpaired Image-to-Image translation from source to target by training the correspondence between image pair, using the adversarial loss and cycle consistency loss [17]. GANs can be used for anomaly detection as well. AnoGAN applies unsupervised learning approach to obtain large amounts of labeled data in the medical field, filtering out normal data and unseen data based on the anomaly score [18].

Autoencoder is one of the most commonly used unsupervised learning model. Autoencoder is a neural network that compresses the input to a smaller vector and reconstruct it to its original form. The difference occurring between the input and output is computed as the reconstruction error. An autoencoder model is trained to minimize the reconstruction error. Variational autoencoders (VAE) presented the model with a generative approach instead of manifold approach [19]. Denoising autoencoder improved the performance of the autoencoder model by adding noise to the input data [20]. Even if data with good representation is slightly damaged, core features can be stably extracted from a part of the input data. Autoencoder can also be used for anomaly detection. Aygun et al. proposed two deep learning based anomaly detection models combining autoencoders and denoising autoencoders, and achieved higher performance than using single models [21]. Mirsky et al. apply an ensemble scheme to autoencoder to detect attack data [22]. Since GAN is difficult to train, we select autoencoder as an experimental model.

Fig. 1 illustrates a structure of the autoencoder for building an AI-NIDS. A red-colored box denotes an encoder, which compresses N inputs to a smaller vector, and a blue-colored box denotes a decoder, which reconstructs the vector to N outputs. The encoder reduces the dimension of an input and transfers it into a hidden representation. In this process, high-dimensional features of the input are compressed to a hidden representation, which is called a latent vector. Through this, we can obtain small-dimensional implicit information. The decoder reconstructs the latent vector into an output of the same shape as the input. In general, decoder plays a role of assisting the encoder to better extract key features from the input. Autoencoders use only normal data in the training phase, so it is trained to reconstruct input as normal data as much as possible. But in the test phase, attack data is also used. So, if autoencoder tries to reconstruct attack data to normal data, the reconstruction error would be high. Using this method, autoencoder can perform anomaly detection in unsupervised learning manner.

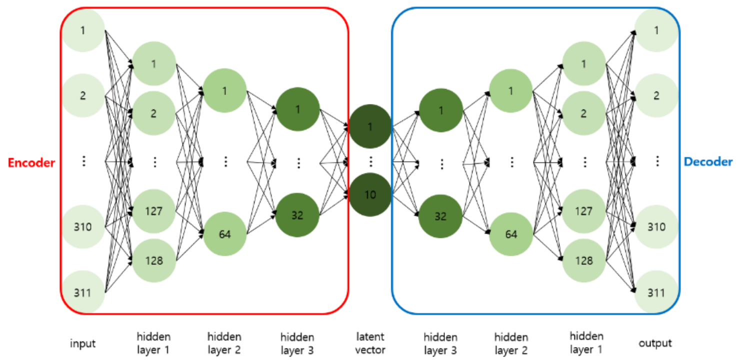


Figure 1. An example of the structure of the autoencoder.

3. Experiment Design

3.1 Security Event Dataset

For evaluation, we collected raw security events from a large enterprise system in real-world environments. The data has been collected over several months, and threat labeling was separately conducted based on an intrusion occurrence report by security operations center (SOC) analysts [23]. In the dataset, there are 798 cyber threats, which occurred evenly over the collection period; and there are also 547 system attacks, 240 scanning, and 11 worm attacks (the categorizing was conducted by the SOC analysts). In total, the data include approximately 4.7 million security event data, of which 0.23 million were identified as cyber threats (i.e., 4,552,316 data were labeled as ‘Normal’, and 230,026 data related to network intrusions were labeled as ‘Threat’). Each raw security event is converted to statistical pattern profiles of concurrent events, following the data preprocessing methodology presented in [23].

In the preprocessing stage, a sliding window that slides at regular intervals according to the event timestamp aggregates security events into an event set. Then, considering each event as a term, TF-IDF is applied to the event set to obtain an event vector with the same size as the number of event types, which can be as many as several thousand. To prevent overfitting due to high dimensionality, event vectors are converted into the pattern profiles to characterize security event patterns. After event vectors including rare security events are selected as basepoint vectors, the cosine similarities between each event vector and the basepoint vectors are collected to form pattern profiles. These pattern profiles consist of 311 features to represent the concurrency of each security event within configured time windows and normalized to be fed into a deep learning model.

Table 1. Statistics of Security Event Dataset

	Train	Validation	Test
Normal	41,029	41,257	41,079
Attack	0	2,536	2,715
Total	41,029	43,793	43,794

Table 1 shows the count of samples for the train, validation, and test sets. The train dataset consists of only normal data for training the autoencoder models. The validation and the test data are highly imbalanced: the size of the normal data is 15 times larger than the attack data.

3.2 Experiment Setup

We use the Area Under the Curve (AUC) score to measure the overall model's performance according to the change in model weight in each training process. As illustrated in Fig. 2, the model with the highest AUC score was chosen as the best model, by comparing the AUC score of the Receiver Operating Characteristic (ROC) during the training process. After the model was selected, a threshold value classifies the outputs of the autoencoder into normal and attack class. The Z-score of standard normal distribution of reconstruction error was used as a criterion to select the threshold value. If the Z-score of the reconstruction error is greater than the set threshold, it is classified as an attack. Otherwise, it is classified as normal.

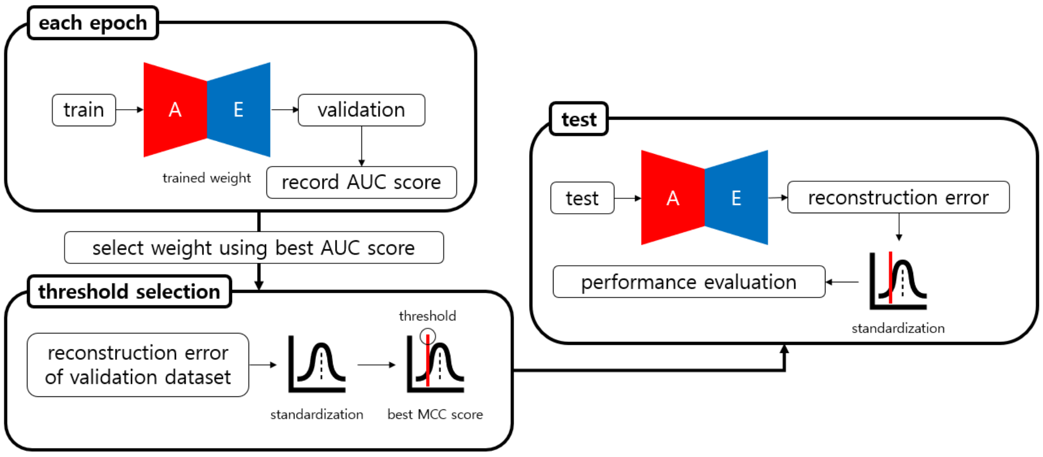


Figure 2. The process of training autoencoder [5]

Autoencoders can take on different structures varying the number of layers and latent size. We configured three representative structures of autoencoder, setting the number of layers for each encoder and decoder and the size of the first layer, following the evaluation methods in [5]. Detailed structures are presented in Table 2. In the case of the (64, 2) and (128, 3) model structures, the size of the layer before the latent vector is 32, so the settable latent vector size ranges from 1 to 31. During the experiment, very little changes in the performance were observed when the latent vector is become larger than 10 dimensions. Accordingly, the latent vector sizes of all three autoencoders are equally set only from 1 to 10.

Our code was implemented using PyTorch. Each model was trained for 100 epochs with 8,096 batch size. Adam optimizer was used and learning rate was set to 1e-4.

Table 2. Structure of autoencoders

Model	The structure of the autoencoder model
(64, 2)	[input] - 64 - 32 - [latent] - 32 - 64 - [output]
(128, 2)	[input] - 128 - 64 - [latent] - 64 - 128 - [output]
(128, 3)	[input] - 128 - 64 - 32 - [latent] - 32 - 64 - 128 - [output]

3.3 Evaluation Metrics

We used three metrics for evaluation: accuracy, F1-score, and Matthew's correlation coefficient (MCC). Each metric is calculated as follows, where *TP* denotes true positive, *TN* denotes true negative, *FP* denotes false positive, and *FN* denotes false negative.

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \tag{1}$$

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

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP) \cdot (TP+FN) \cdot (TN+FP) \cdot (TN+FN)}} \tag{3}$$

Accuracy is the number of correctly predicted data divided by the total number of data (equation 1). F1-score is the harmonic average of precision and recall (equation 2). MCC is an evaluation metric used for binary classification and has a value between -1 and 1 (equation 3). A MCC score close to 1 indicates good performance, while a MCC score close to -1 indicates a bad performance. MCC is recognized as a balanced scale since it evaluates performance in all classes fairly, taking into account the bias of the data.

4. The Experiment Result

In this section, we report the result of the experiments and the visualization of how the autoencoder model classifies data.

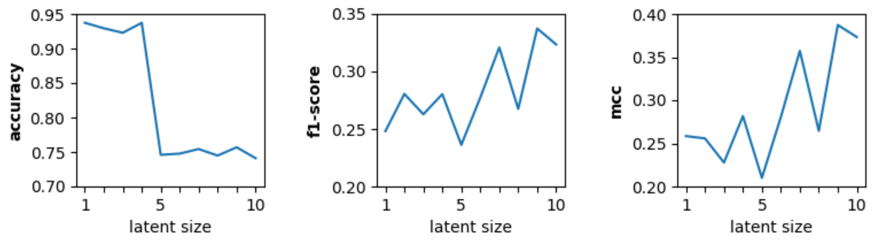


Figure 3. The performance of the autoencoder with the structure of (64, 2) as the latent size increases.

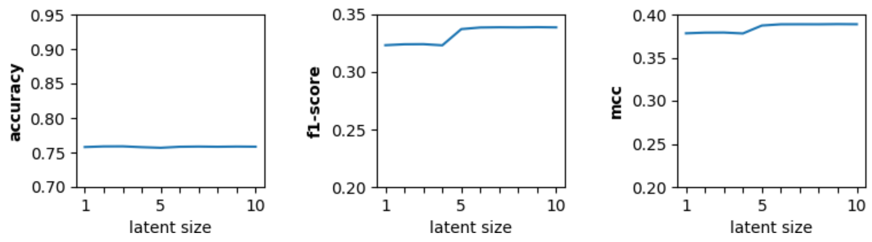


Figure 4. The performance of the autoencoder with the structure of (128, 2) as the latent size increases.

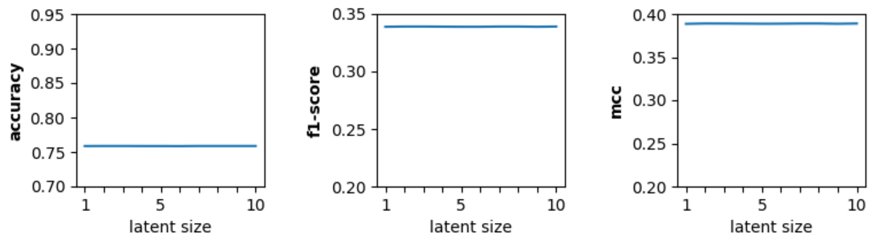


Figure 5. The performance of the autoencoder with the structure of (128, 3) as the latent size increases.

4.1 Attack Detection Performance

Fig. 3 shows the performance of the autoencoder with the structure of (64, 2). In the accuracy graph, the performance drops significantly when the latent is greater than or equal to 5. In the F1-score and MCC graph, on the other hand, the performance gradually improves when the latent is greater than or equal to 6. Since the data are highly imbalanced, we believe that the MCC metric best represents the performance. In other words, when the model size and the layers are small, the performance improves as the latent size increases.

Fig. 4 shows the performance results of the autoencoder with the structure of (128, 2). Unlike the trend obtained from the structure of (64, 2) where the three metrics fluctuate as the latent size alters, the performance from the structure of (128, 2) changes only little as the latent size increases. It is noted that the overall accuracy is less than that of the structure of (64, 2), whereas the F1-score and MCC scores are greater than those with the structure of (64, 2).

Fig. 5 shows the performance results of the autoencoder with the structure of (128, 3). All three metrics show that the level of performance is maintained regardless of the latent size. The F1 and MCC scores are greater than those with the structure of (128, 2).

Table 3. The performance results in each structure of autoencoder at the best MCC score.

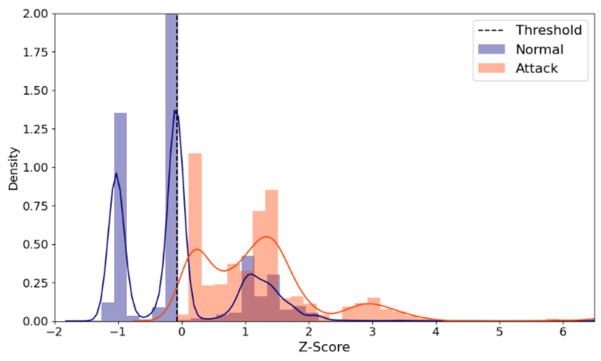
Structure	Latent Size	Accuracy	F1-score	MCC
(64, 2)	9	0.7568	0.3374	0.3873
(128, 2)	9	0.7586	0.3390	0.3889
(128, 3)	8	**0.7587**	**0.3391**	**0.3890**

Table 3 shows the performance in accuracy, F1-score, and MCC for each structure where the autoencoder model with the highest MCC score is selected. The bold text indicates the highest performance. The autoencoder with the structure of (128, 3) recorded 0.7587 in accuracy, 0.3391 in F1-score, and 0.3890 in MCC, which are best performances among the three autoencoders.

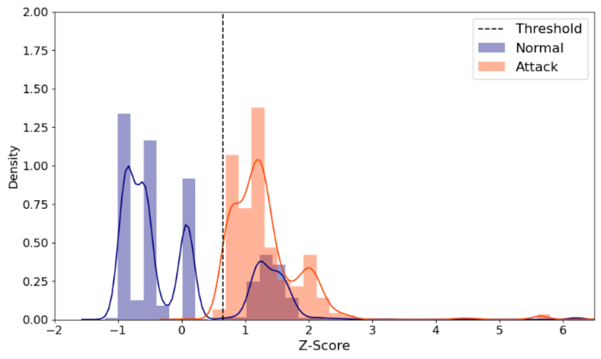
4.2 Data Distribution

In our study, Z-score is computed using only normal samples from the validation dataset to calculate the mean and variance values of the reconstruction distances between input and output. Then, we standardize the reconstruction distances. Fig. 6 shows the Z-score distribution density histograms for the three different autoencoders shown in Table 3. The threshold described in Section 3.2 is indicated by the dashed vertical line. In the graphs, we are interested in how clearly the normal data and the attack data are separated by the threshold. Since NIDS using autoencoder considers the data as an attack when the reconstruction error of the data is greater than the threshold, an ideal histogram is divided into normal data on the left side and attack data on the right side, based on the threshold line. All three graphs show that some normal data are located on both sides of the threshold, whereas most of the attack data are located on the right. This suggests that the autoencoder can detect most of the attack data correctly, and it can misjudge some normal sample as an attack. The reason we used the density when drawing the histogram is to make the distribution in each data type clearly visible despite the large quantitative difference between the attack data and the normal data as shown in Table 1. Looking at the right side of the graphs in Fig. 6, where data is classified as an attack, the amount of attack data seems to be much larger than the amount of normal data. Nevertheless, since they are ratios within each data type, comparing the actual quantity means that there are much more normal data than attack data even on the right side.

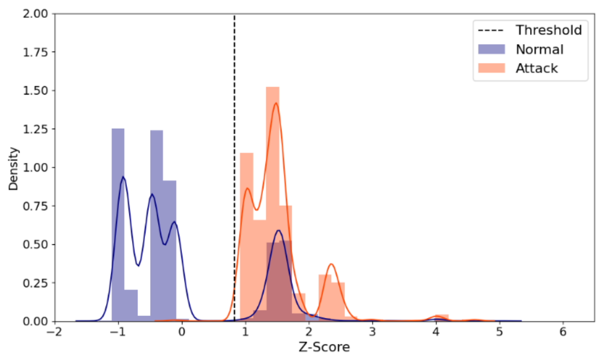
This can be confirmed through the precision and recall from the confusion matrix. In the case of (c) where the (128, 3) structure is employed, precision is 0.2042 and recall is 0.9985. A high recall value close to 1 means that it can detect almost all attacks, and a low precision means that it can generate a lot of false alerts, when this AI-NIDS is situated in a real environment.



(a) Autoencoder (64, 2) with latent vectors of size 9



(b) Autoencoder (128, 2) with latent vectors of size 9



(c) Autoencoder (128, 3) with latent vectors of size 8

Figure 6. Z-score distribution density histograms for the three autoencoder models. Blue denotes normal data, orange denotes attack data. For each color, the solid line indicates the kernel density estimated plot, and the bar plot indicates the histogram.

4.3 Discussion

This section summarizes and discusses our findings. When discussing the result, we use MCC to measure the performance of the model as the real network dataset tends to be imbalanced. We obtained the best performance (MCC = 0.389) when the layer size is 128 (the greatest), the number of layers is 3 (the greatest), and the latent dimension is 8. When the model is small, the performance in MCC improves as the latent size increases. However, the model shows stable and better performance regardless of the latent size when the model size and the layers are greater. Our findings are partially in line with the results found in the previous study [5], which evaluates NIDS using the network benchmark NSL-KDD and the IoT datasets.

First, we confirm that the model capacity impacts the NIDS performance. In general, the models with large capacities tend to perform better. Furthermore, we discover that the model depth (i.e., number of hidden layers) is positively associated with the performance; the performance improves as the model depth grows. This finding is also supported by the Z-score distributions illustrated in Fig. 6, separating attack and normal data clearly as the model size gets larger.

Second, we observe the finding reported in [5] that IDS performance enhances as the latent dimension increases, only when the model size is small. While [5] obtained some best performances when the latent sizes are very small, we obtained the best MCC values when the latent size are close to the maximum value.

Third, we observe that the performance is stable as the model size grows. This finding was only suggestive in [5] and confirmed by our study. Additionally, the Z-score distribution analysis suggest that our current model tends to detect almost all attacks producing lots of false alarms. Although we set the threshold to obtain the highest MCC value, the NIDS operators can set the threshold taking into other practical issues. For instance, if precision matters more than recall, the threshold can be set at a greater value to filter out some normal samples.

5. Conclusion

In this work, we evaluate unsupervised intrusion detection using real security event data. We conducted experiments testing different autoencoder models to test how the model structure impacts the performance following the methodology [5]. We observe that the model structure impacts the NIDS performance in terms of two characteristics. Autoencoder models tend to produce stable and better performance as the model size gets larger. We also suggest to set the threshold that determines if a given sample is an attack or not, taking into account the performance as well as the practical efficiency of running the system in the real environment. We believe that these findings can help AI-NIDS developers design optimal unsupervised models in the real network environment.

Acknowledgement

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2020-0-00952, Development of 5G Edge Security Technology for Ensuring 5G+ Service Stability and Availability).

References

[1] Raghunath, Bane Raman, and Shivsharan Nitin Mahadeo. "Network intrusion detection system (NIDS)." 2008 First International Conference on Emerging Trends in Engineering and Technology. IEEE, 2008. DOI: 10.1109/ICETET.2008.252.

[2] Buczak, Anna L., and Erhan Guven. "A survey of data mining and machine learning methods for cyber security intrusion detection." IEEE Communications surveys & tutorials 18.2 (2015): 1153-1176.
DOI: 10.1109/COMST.2015.2494502.

[3] Zavrak, Sultan, and Murat İskefiyeli. "Anomaly-based intrusion detection from network flow features using variational autoencoder." IEEE Access 8 (2020): 108346-108358.
DOI: 10.1109/ACCESS.2020.3001350.

[4] McHugh, John, Alan Christie, and Julia Allen. "Defending yourself: The role of intrusion detection systems." IEEE software 17.5 (2000): 42-51.
DOI: 10.1109/52.877859.

[5] Song, Youngrok, Sangwon Hyun, and Yun-Gyung Cheong. "Analysis of autoencoders for network intrusion detection." Sensors 21.13 (2021): 4294.
DOI: 10.3390/s21134294.

[6] Kumar, Sailesh. "Survey of current network intrusion detection techniques." Washington Univ. in St. Louis (2007): 1-18.

[7] Wu, Handong, Stephen Schwab, and Robert Lom Peckham. "Signature based network intrusion detection system and method." U.S. Patent No. 7,424,744. 9 Sep. 2008.

[8] Zhang, Jiong, and Mohammad Zulkernine. "Anomaly based network intrusion detection with unsupervised outlier detection." 2006 IEEE International Conference on Communications. Vol. 5. IEEE, 2006.
DOI: 10.1109/ICC.2006.255127.

[9] Ahmad, Zeeshan, et al. "Network intrusion detection system: A systematic study of machine learning and deep learning approaches." Transactions on Emerging Telecommunications Technologies 32.1 (2021): e4150.
DOI: 10.1002/ett.4150.

[10] Chkirbene, Zina, et al. "TIDCS: A dynamic intrusion detection and classification system based feature selection." IEEE Access 8 (2020): 95864-95877.
DOI: 10.1109/ACCESS.2020.2994931.

[11] Vinayakumar, Ravi, et al. "Deep learning approach for intelligent intrusion detection system." Ieee Access 7 (2019): 41525-41550.
DOI: 10.1109/ACCESS.2019.2895334.

[12] Panda, Mrutyunjaya, et al. "Network intrusion detection system: A machine learning approach." Intelligent Decision Technologies 5.4 (2011): 347-356.
DOI: 10.3233/IDT-2011-0117.

[13] Althubiti, Sara A., Eric Marcell Jones, and Kaushik Roy. "LSTM for anomaly-based network intrusion detection." 2018 28th International telecommunication networks and applications conference (ITNAC). IEEE, 2018.
DOI: 10.1109/ATNAC.2018.8615300.

[14] Jo, Wooyeon, et al. "Packet Preprocessing in CNN-based network intrusion detection system." Electronics 9.7 (2020): 1151.
DOI: 10.3390/electronics9071151.

[15] Goodfellow, Ian, et al. "Generative adversarial networks." Communications of the ACM 63.11 (2020): 139-144.
DOI: 10.1145/3422622.

[16] Karras, Tero, Samuli Laine, and Timo Aila. "A style-based generator architecture for generative adversarial networks." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019.
DOI: 10.1109/cvpr.2019.00453.

[17] Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." Proceedings of the IEEE international conference on computer vision. 2017.
DOI: 10.1109/ICCV.2017.244.

[18] Schlegl, Thomas, et al. "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery." International conference on information processing in medical imaging. Springer, Cham, 2017.
DOI: 10.1007/978-3-319-59050-9_12.

[19] Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." arXiv preprint arXiv:1312.6114 (2013). DOI: 10.48550/arXiv.1312.6114.

[20] Vincent, Pascal, et al. "Extracting and composing robust features with denoising autoencoders." Proceedings of the 25th international conference on Machine learning. 2008.}
DOI: 10.1145/1390156.1390294.

[21] Aygun, R. Can, and A. Gokhan Yavuz. "Network anomaly detection with stochastically improved autoencoder based models." 2017 IEEE 4th international conference on cyber security and cloud computing (CSCloud). IEEE, 2017.
DOI: 10.1109/CSCloud.2017.39.

[22] Mirsky, Yisroel, et al. "Kitsune: an ensemble of autoencoders for online network intrusion detection." arXiv preprint arXiv:1802.09089 (2018).
DOI: 10.48550/arXiv.1802.09089.

[23] Lee, Jonghoon, et al. "Cyber threat detection based on artificial neural networks using event profiles." IEEE Access 7 (2019): 165607-165626.
DOI: 10.1109/ACCESS.2019.2953095.

[24] Chen, Z., Peng, Z., Zou, X., Sun, H. (2022). Deep Learning Based Anomaly Detection for Muti-dimensional Time Series: A Survey. In: Lu, W., Zhang, Y., Wen, W., Yan, H., Li, C. (eds) Cyber Security. CNCERT 2021. Communications in Computer and Information Science, vol 1506. Springer, Singapore.
DOI: https://doi.org/10.1007/978-981-16-9229-1_5