Building Autoencoder Models for Detecting Real Network Attacks

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

[1] Department of Computer Software, Sungkyunkwan University, South Korea, zyoa@skku.edu, flamecracker1220@gmail.com
[2] Department of Artificial Intelligence, Sungkyunkwan University, South Korea, {tjdckdals, aimecca}@skku.edu
[3] Electronics and Telecommunications Research Institute, South Korea, {mine, queue}@etri.re.kr

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*

I. 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 [15]. NIDS can be implemented in two ways: signature based approach and anomaly detection based approach [10]. While signature based NIDS employ rules to check whether a given input is an attack or not [20], anomaly detection based NIDS determines attacks relying on how much the input deviates from the majority of the data [22]. Therefore, signature based NIDS is suitable for capturing well-known attacks, while anomaly based NIDS is suitable for capturing unknown attacks [10].

In the past, classical supervised machine learning models such as Naive Bayes, Decision Tree, Random Forest, K-Nearest Neighbors, and Support Vector Machine(SVM) were widely used for building signature-based AI-NIDS [4, 5, 18, 14]. 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 [1, 2, 17, 7].

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 everyday, and as a result, intrusion patterns are changing rapidly [21]. Supervised learning models can show poor performance in detecting such new attacks [12]. To handle this problem, unsupervised machine learning models are employed. Song et al. [17] reports that NIDS using autoencoder can achieve good performance on network benchmark datasets such as NSL-KDD and IoT datasets. The study suggests to experiment 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.

II. 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 [6]. 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 [8]. 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 [23]. 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 [16].

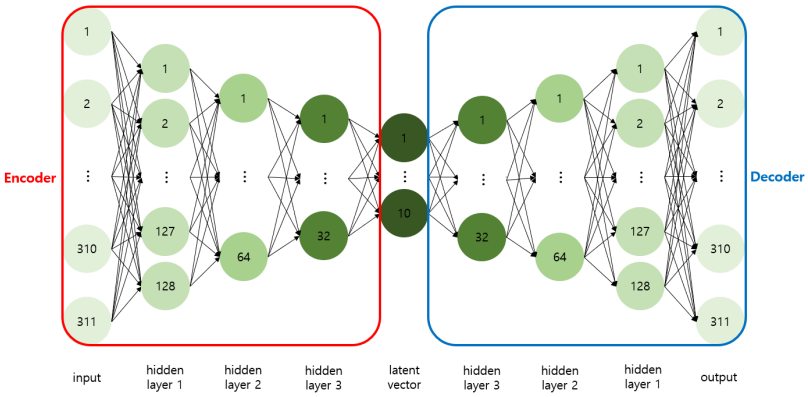

Fig. 1. An example of the structure of the autoencoder.

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 [9]. Denoising autoencoder improved the performance of the autoencoder model by adding noise to the input data [19]. 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 [3]. Mirsky et al. apply an ensemble scheme to autoencoder to detect attack data [13]. 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.

III. Experiments

A. *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 [11]. 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 warm attacks (the categorizing was conducted by the SOC analysts). In total, the data include approximately 4.7 millions security event data, of which 0.23 millions 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 [11]. In the preprocessing stage, an event vector is converted from an event set, and the final event profile of correlated events is transformed by event embedding from an event vector. 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 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.

	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. Statistics of security event dataset

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]

Table 2. Structure of autoencoders

B. 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. 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.

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 [17]. 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 became larger than 10 dimensions. Accordingly, the latent vector sizes of the all three autoencoders are equally set only from 1 to 10. Each model was trained for 100 epochs with 8,096 batch size. Adam optimizer was used and learning rate was set to 1e-4.

C. Evaluation Metrics

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

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \quad (1)$$

Accuracy is the number of correctly predicted data divided by the total number of data. F1-score is the harmonic average of precision and recall. MCC is an evaluation metric used for binary classification and has a value between -1 and 1 (equation 1). 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.

IV. The Experiment Results

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

A. Attack Detection Performance

Fig. 2 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. 3 shows the performance results of the autoencoder with the structure of (128, 2). It is noted that the overall accuracy is less than that of the structure of (64, 2) and maintains similar performance regardless of the latent size. In the F1-score and MCC graphs, the performance improves a little as the latent size increases. Overall, these scores are greater than those with the structure (64, 2).

Fig. 4 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).

Structure	Latent	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. The performance results in each structure of autoencoder at the best MCC score.

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.

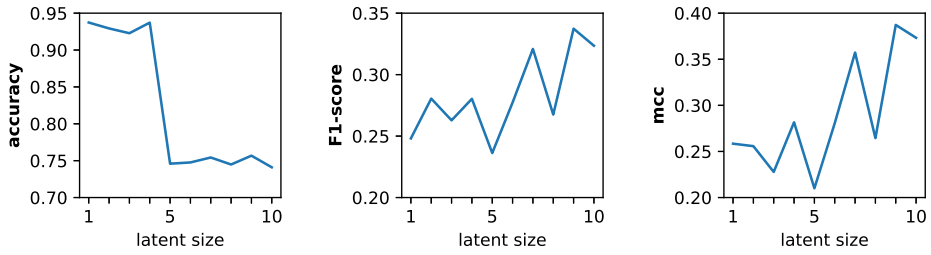

Fig. 2. The performance of the autoencoder with the structure of (64, 2) as the latent size increases.

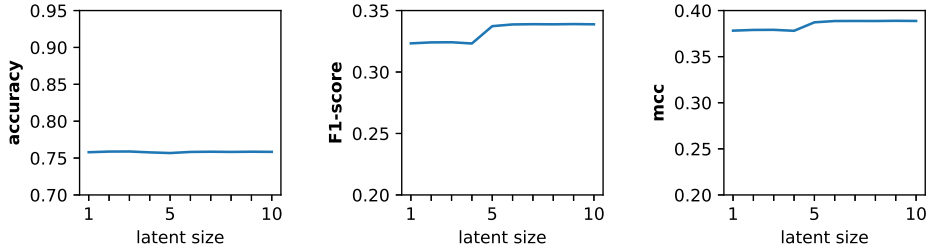

Fig. 3. The performance of the autoencoder with the structure of (128, 2) as the latent size increases.

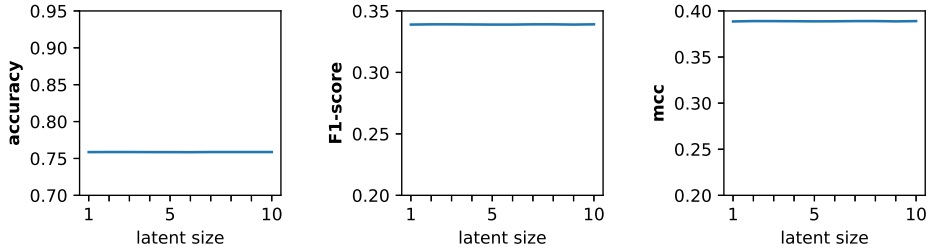

Fig. 4. The performance of the autoencoder with the structure of (128, 3) as the latent size increases.

B. 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 layer 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 [17], 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.

Second, we observe the finding reported in [17] that IDS performance enhances as the latent dimension increases, only when the model size is small. While [17] 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 [17], and confirmed by our study.

V. 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 [17]. 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 partly supported by Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No.2019-0-00421, Artificial Intelligence Graduate School Program(Sungkyunkwan University)) and 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] Zeeshan Ahmad, Adnan Shahid Khan, Cheah Wai Shiang, Johari Abdullah, and Farhan Ahmad. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, 32(1):e4150, 2021.

[2] Sara A Althubiti, Eric Marcell Jones, and Kaushik Roy. Lstm for anomaly-based network intrusion detection. In *2018 28th International telecommunication networks and applications conference (ITNAC)*, pages 1–3. IEEE, 2018.

[3] R Can Aygun and A Gokhan Yavuz. Network anomaly detection with stochastically improved autoencoder based models. In *2017 IEEE 4th international conference on cyber security and cloud computing (CSCloud)*, pages 193–198. IEEE, 2017.

[4] Anna L Buczak and Erhan Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications surveys & tutorials*, 18(2):1153–1176, 2015.

[5] Zina Chkirbene, Aiman Erbad, Ridha Hamila, Amr Mohamed, Mohsen Guizani, and Mounir Hamdi. Tidcs: A dynamic intrusion detection and classification system based feature selection. *IEEE Access*, 8:95864–95877, 2020.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[7] Wooyeon Jo, Sungjin Kim, Changhoon Lee, and Taeshik Shon. Packet preprocessing in cnn-based network intrusion detection system. *Electronics*, 9(7):1151, 2020.

[8] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[9] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[10] Sailesh Kumar. Survey of current network intrusion detection techniques. *Washington Univ. in St. Louis*, pages 1–18, 2007.

[11] Jonghoon Lee, Jonghyun Kim, Ikkyun Kim, and Kijun Han. Cyber threat detection based on artificial neural networks using event profiles. *IEEE Access*, 7:165607–165626, 2019.

[12] John McHugh, Alan Christie, and Julia Allen. Defending yourself: The role of intrusion detection systems. *IEEE software*, 17(5):42–51, 2000.

[13] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. Kitsune: an ensemble of autoencoders for online network intrusion detection. *arXiv preprint arXiv:1802.09089*, 2018.

[14] Mrutyunjaya Panda, Ajith Abraham, Swagatam Das, and Manas Ranjan Patra. Network intrusion detection system: A machine learning approach. *Intelligent Decision Technologies*, 5(4):347–356, 2011.

[15] Bane Raman Raghunath and Shivsharan Nitin Mahadeo. Network intrusion detection system (nids). In *2008 First International Conference on Emerging Trends in Engineering and Technology*, pages 1272–1277. IEEE, 2008.

[16] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International conference on information processing in medical imaging*, pages 146–157. Springer, 2017.

[17] Youngrok Song, Sangwon Hyun, and Yun-Gyung Cheong. Analysis of autoencoders for network intrusion detection. *Sensors*, 21(13):4294, 2021.

[18] Ravi Vinayakumar, Mamoun Alazab, KP Soman, Prabaharan Poornachandran, Ameer Al-Nemrat, and Sitalakshmi Venkatraman. Deep learning approach for intelligent intrusion detection system. *Ieee Access*, 7:41525–41550, 2019.

[19] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[20] Handong Wu, Stephen Schwab, and Robert Lom Peckham. Signature based network intrusion detection system and method, September 9 2008. US Patent 7,424,744.

[21] Sultan Zavrak and Murat İskefiyeli. Anomaly-based intrusion detection from network flow features using variational autoencoder. *IEEE Access*, 8:108346–108358, 2020.

[22] Jiong Zhang and Mohammad Zulkernine. Anomaly based network intrusion detection with unsupervised outlier detection. In *2006 IEEE International Conference on Communications*, volume 5, pages 2388–2393. IEEE, 2006.

[23] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

SUMMARY OF THIS PAPER

A. *Problem Setup*

As network technology develops, the importance of network security technology has also increased. A Network Intrusion Detection System (NIDS) can be implemented in two ways: signature based approach and anomaly detection based approach. Signature based NIDS employs rules to check whether a given input is an attack or not, and anomaly detection based NIDS determines attacks relying on how much the input deviates from the majority of the data. Signature based NIDS using supervised learning model is suitable for capturing well-known attacks, but has drawbacks such as the cost for labeled data and difficulty in responding to rapidly changing network intrusion. To handle this problem, anomaly detection using unsupervised machine learning models (MS-NIDS) have been deployed. In this paper, we expand the previous approach that leverage autoencoder models which investigate model structure and parameters for achieving good performance on network benchmark datasets such as NSL-KDD and IoT datasets.

B. *Novelty*

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.

C. *Algorithms*

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. 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.

D. *Experiments*

We configured three representative structures of autoencoder setting the number of layers for each encoder and decoder and the size of the first layer. We evaluate those models using real security event data. The results of our evaluation show that the autoencoder with 3 layers and latent vectors of size 8 recorded 0.7587 in accuracy, 0.3391 in F1-score, and 0.389 in MCC, which are best performances among the three autoencoders.

We observe that the model structure impacts the NIDS performance in terms of model size and latent dimension. Autoencoder models tend to produce stable and better performance as the model size gets larger regardless of the latent size. When the model is small, the performance in MCC improves as the latent size increases.