IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received July 1, 2021, accepted July 21, 2021, date of publication July 26, 2021, date of current version August 2, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3100087

Network Anomaly Detection Using Memory-Augmented Deep Autoencoder

BYEONGJUN MIN[1], JIHOON YOO[2], SANGSOO KIM[3], DONGIL SHIN[2], AND DONGKYOO SHIN[2]

[1]Department of Computer Engineering, Convergence Engineering for Intelligent Drones, Sejong University, Seoul 05006, South Korea
[2]Department of Computer Engineering, Sejong University, Seoul 05006, South Korea
[3]Agency for Defense Development, Daejeon 05600, South Korea

Corresponding author: Dongkyoo Shin (shindk@sejong.ac.kr)

This work was supported in part by the Defense Acquisition Program Administration, and in part by the Agency for Defense Development under Contract UD2000014ED.

ABSTRACT In recent years, attacks on network environments continue to rapidly advance and are increasingly intelligent. Accordingly, it is evident that there are limitations in existing signature-based intrusion detection systems. In particular, for novel attacks such as Advanced Persistent Threat (APT), signature patterns have problems with poor generalization performance. Furthermore, in a network environment, attack samples are rarely collected compared to normal samples, creating the problem of imbalanced data. Anomaly detection using an autoencoder has been widely studied in this environment, and learning is through semi-supervised learning methods to overcome these problems. This approach is based on the assumption that reconstruction errors for samples that are not used for training will be large, but an autoencoder is often over-generalized and this assumption is often broken. In this paper, we propose a network intrusion detection method using a memory-augmented deep auto-encoder (MemAE) that can solve the over-generalization problem of autoencoders. The MemAE model is trained to reconstruct the input of an abnormal sample that is close to a normal sample, which solves the generalization problem for such abnormal samples. Experiments were conducted on the NSL-KDD, UNSW-NB15, and CICIDS 2017 datasets, and it was confirmed that the proposed method is better than other one-class models.

INDEX TERMS Network intrusion detection, autoencoder, anomaly detection, memory-augmented autoencoder.

I. INTRODUCTION

Recently, as information and communication technologies advance, the network environment has also expanded very rapidly, and cyber threats to the network environment are increasing. In order to detect cyber threats to a network, many companies operate a network-based intrusion detection system (NIDS) [1]. When various attacks on the network occur, the aim is to report them to the security manager. Most existing network intrusion detection systems utilize misuse detection methods [2] and are sometimes called signature-based methods or knowledge-based methods. These methods detect an attack through pattern matching after defining a pattern for attacks that already occur frequently as identified by security experts. However, intelligent and intelligent threats, such as Advanced Persistent Threat (APT) attacks, have recently emerged, bringing to the forefront limitations in existing signature-based detection methods. Moreover, there are issues of time and cost to continuously generate signature patterns.

The anomaly detection method [3] is a method of detecting abnormal behavior by modeling normal behavior, unlike the misuse detection method, and can also detect zero-day attacks. However, since modeling for normal behavior is not a simple problem, many studies have recently been conducted to solve this problem using machine learning techniques [3]–[7]. Machine learning (ML) is suitable for anomaly detection because it can model data and infer prediction results. In machine learning, the problem of classifying normal and abnormal, such as detecting network attacks, is defined as a binary classification problem. However, most of the data collected in the real world are imbalanced data, which means that the data in the minority class consists of much less data than the data in the majority class. As such,

The associate editor coordinating the review of this manuscript and approving it for publication was Ahmed Farouk.

we fall into the problem of having to train machine learning models with such imbalanced data. Furthermore, applying commonly-used supervised learning models to imbalanced datasets can result in poor classification performance [4]. In particular, the detection rate of minority classes decreases noticeably because the decision boundary of the model is trained to be biased toward the majority class [5].

Many studies have turned to one-class learning based on Support Vector Machine (SVM) and Autoencoder (AE) to solve the anomaly detection problem from such imbalanced data. One-class learning trains a model through samples of only one specific class and is considered a form of semi-supervised learning [6]. This method is very suitable for a network intrusion detection environment where most of the samples are normal. In this study, we conduct a study on an AE-based anomaly detection method and point out the over-generalization problem based on an autoencoder. This method trains AE through only normal samples and assumes that it has a high reconstruction error for attack samples that are not used for training. However, AE often reconstructs some attack samples very well, making these assumptions suspicious. In conclusion, this problem degrades the performance of the AE-based anomaly detection method.

In this paper, we propose an anomaly detection method using the MemAE [7] that can solve the over-generalization problem of the AE. The MemAE model consists of an encoder, a decoder, and a memory module, and the purpose of the memory module is to learn prototypical patterns for the normal inputs used in training. Through this method, the reconstruction results of the abnormal samples received tend to be close to those of normal samples. This is a method of replacing the encoder output for the abnormal input with a new decoder input that is aggregated into similar normal samples from the memory learned only with normal samples. We measure the performance of various network intrusion detection datasets using the MemAE model and compare the performance with other widely used one-class models. Experiments were conducted on the open network intrusion detection datasets NSL-KDD [8], UNSW-NB15 [9], and CICIDS 2017 [10], and the results were compared with an AE and a one-class SVM (OCSVM) [11].

The main contributions of the study are summarized as follow:

- This study concentrates on the over-generalization problem that can occur in the AE-based anomaly detection model that is commonly used in network intrusion detection and describes the reasons for these problems.
- These problems occur relatively often, but there is not much discussion. In this paper, we apply the MemAE model that can solve these problems to confirm the effectiveness in the network intrusion detection domain.
- We used various datasets to evaluate the model and report the actual classification results using ROC curves and thresholds in detail.

II. REALATED STUDIES

A. MACHINE LEARNING-BASED NETWORK INTRUSION DETECTION

Recently, studies on various intrusion detection methods using machine learning have been proposed. Leevy and Khoshgoftaar [12] presented a machine learning study survey on the CICIDS 2018 dataset for network intrusion detection. They pointed out that although the results reported in the entire study are generally high, the bias in the results should be questioned because the entire study did not take into account the imbalanced data problem.

Yang *et al.* [13] proposed a generative model, the Improved Conditional Variational Autoencoder (ICVAE) model, to solve the imbalanced data problem. Unlike CVAE, ICVAE has the advantage of can reusing the weights of the learned Encoder network because class labels are used only as extra inputs to the decoder. They initiated Deep Neural Network (DNN) classifier initial weights through the weights of the encoder network of trained ICVAE. The experimental results were compared with other oversampling techniques, and it is reported that ICVAE showed the best performance.

Kim *et al.* [14] conducted a study to detect denial of service (DoS) attacks using a Convolutional Neural Network. They converted the preprocessed network data into a two-dimensional image format, and composed a total of 18 scenarios considering the number of convolution layers, kernel size, and RGB or grayscale. Through this, a CNN model was trained and its performance was compared with a Recurrent Neural Network (RNN) model.

Research on the one-class classification technique based on the SVM model is also actively being conducted, and OCSVM [11] is representative. The OCSVM model is an unsupervised learning model that uses only normal samples for training and aims to learn a discriminal hyperplane surrounding the normal samples. Extensions of this study include Deep Support Vector Data Description (SVDD) [15] and Deep SVDD [16]. Tian *et al.* [17] pointed out that OCSVM is vulnerable to outliers and noise, and proposed Ramp-OCSVM to overcome this. They reported experimental results on the UNSW-NB15 dataset and the NSL-KDD dataset.

Aygun and Yavuz [18] conducted a study on a network intrusion detection model using a Denoising Autoencoder (DAE), and conducted experiments on the NSL-KDD dataset. DAE is a model that learns how to receive input with added noise from the original data and reconstruct it into the original data. They reported that there was no significant difference when comparing the performance of the proposed method and other hybrid models.

Tang *et al.* [19] conducted a study on a DDoS detection model combining the AE model and the OCSVM model. They first trained the AE model, then converted the input into a low-dimensional feature vector using an encoder, and then combined the two models by training the OCSVM model again. The experimental results were

compared with the OCSVM model and the model combining CNN and LSTM, and the proposed model showed the best performance.

Zavrak and İskefiyeli [20] proposed a study on network anomaly detection through a Variational Autoencoder (VAE). They used the reconstruction probability as an outlier score and conducted experiments on the CICIDS2017 dataset. The Receiver Operating Characteristics (ROC) curve was used for model evaluation, and the results were compared with the AE model and OCSVM.

TABLE 1. Previous studies on machine learning-based network intrusion detection.

Author	Dataset	Technique	Contribution
J. L. Leevy[12]	CICIDS 2018	Survey	Point out the problems of previous studies.
Y. Yang[13]	NSL-KDD UNSW-NB15	ICVAE (supervised)	New approach
J. Kim [14]	CICIDS 2018	CNN (supervised)	DoS detection
Y. Tian [17]	NSL-KDD UNSW-NB15	Ramp-OCSVM (one class)	Improve OCSVM model
R. C. Aygun [18]	NSL-KDD	Denoising AE (one class)	Zero-day attack detection
T. Tang [19]	CICIDS 2017	Enseble of AE and OCSVM (one class)	New approach
S. Zavrak [20]	CICIDS 2017	VAE (one class)	Detection of network attacks from flow-based features

Table 1. shows the comparison and contribution to machine learning-based intrusion detection studies. However, while their studies contribute a lot, there are some problems. The contribution of several previous studies [14], [18]–[20] is meaningful but has been experimented on only one dataset. In addition, since traces of intrusion occur very little in the actual network environment, the supervised learning-based studies in [13], [14] have limitations in their practical application. Therefore, we conduct an experiment using multiple data sets mentioned in these previous studies and conduct an anomaly detection study using only normal flow data that is relatively easy to obtain by conducting one-class learning-based studies.

B. AUTOENCODER

Autoencoder (AE) is a type of unsupervised learning neural network model used to learn efficient data coding. As shown in Fig. 1, the autoencoder is composed of an encoder network and a decoder network. In general, the encoder network and the decoder network have a symmetrical structure from the center bottleneck. The encoder network in equation (1) maps the original data x onto a low-dimensional feature space, while the decoder network in equation (2) attempts to recover $\hat{x}$ from the projected low-dimensional space. The autoencoder aims to learn to encode that preserves as much important information as possible for input reconstruction. The parameters of these two networks are learned with a

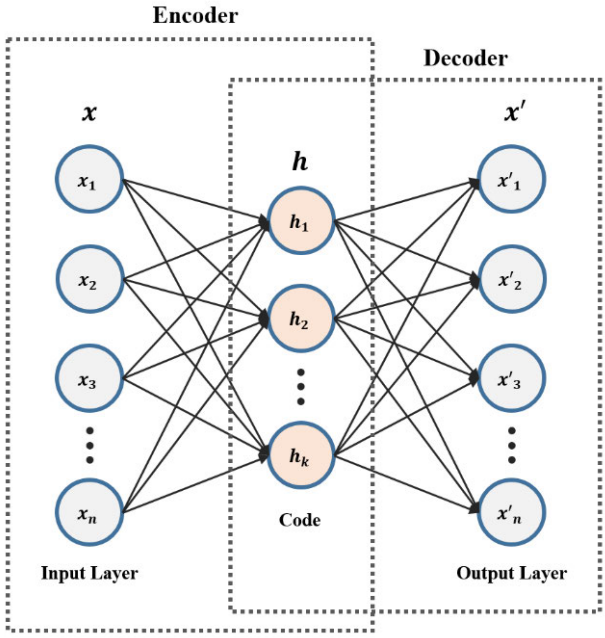


FIGURE 1. Autoencoder structure.

reconstruction loss function.

$$z = \sigma(W_e x + b_e) \tag{1}$$

$$\hat{x} = \sigma(W_d z + b_d) \tag{2}$$

An autoencoder with a single hidden layer has an encoder and decoder as in equation (1) and equation (2), respectively. Where W and b are the weight and bias and σ is the activation function. The activation function σ can use nonlinear functions and linear functions, and when using linear functions, it works similarly to Principal Component Analysis (PCA). z is the hidden presentation called the latent vector or code. The decoder network in equation (2) uses z as the input to output the reconstructed $\hat{x}$. A stacked Auto Encoder is a model in which the autoencoders have multiple hidden layers. Stacked Autoencoder is an autoencoder model with multiple hidden layers.

C. MEMORY-AUGMENTED DEEP AUTOENCODER

Memory-augmented deep Autoencoder (MemAE) [7] consists of an encoder network, a decoder network, and a memory module as shown in Fig. 2. The purpose of the memory module is to learn and record a finite number of prototypical patterns of the input data. The memory module is located between the encoder network and the decoder network as shown in Fig. 3, and it receives the output z of the encoder as an input and outputs the item $\hat{z}$ for delivery to the decoder. So, unlike AE, MemAE does not feed the output z of the encoder directly to the decoder. Internally, the latent vector z is used as a query to retrieve the most relevant item in memory. Then those items are aggregated and passed to the decoder. As a result, this internal process of the memory module makes it possible to induce an output close to a normal sample because

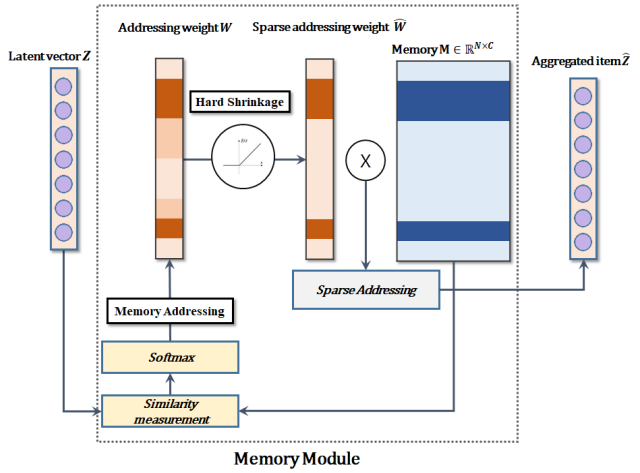


FIGURE 2. Memory module.

even if the latent vector of abnormal input becomes an input, it is aggregated and reconstructed from the normal prototypical patterns learned in memory. Therefore, it is possible to solve the problem of over-generalization in which abnormal samples are well reconstructed.

$$z_i = wM = \sum_{i=1}^{N} w_i m_i \tag{3}$$

$$w_i = \frac{\exp(d\,(z, m_i))}{\sum_{j=1}^{N} \exp(d\,(z, m_j))} \tag{4}$$

The memory is matrix $M \in \mathbb{R}^{N\times C}$ containing N real-valued vectors of fixed dimension C, as shown in Fig. 2. C is of the same dimension as z. As shown in equation (3), aggregated item $\hat{z}$ is obtained through soft addressing of vector w and memory M. Memory M reflects items to record various prototypical patterns of normal data. The weight vector w is obtained from z and is computed through a softmax operation as shown in equation (4). Where d is a similarity measurement and is defined as cosine similarity.

In addition, as shown in equation (5), memory modules can increase sparsity through hard shrinkage operations on soft addressing vector w. The sparse addressing encourages the model to represent an example using fewer, but more relevant memory items, leading to learning more informative representations in memory. However, since it is difficult to directly calculate backward the discontinuous function, they use the continuous ReLU activation function to override it in the same form as equation (6), considering that not all input values are negative. The value of λ is the shrinkage threshold, which uses the value of interval [1/N, 3/N].

$$\hat{w}_i = h\,(w_i; \lambda) = \begin{cases} w_i, & if\ w_i > \lambda \\ 0, & otherwise \end{cases} \tag{5}$$

$$\hat{w}_i = \frac{\max(w_i - \lambda, 0) \cdot w_i}{|w_i - \lambda| + \varepsilon} \tag{6}$$

After the hard shrinkage operation, re-normalization is performed with $\hat{w} = \hat{w} / \left\|\hat{w}\right\|_1$. There are two types of memory modules: sparse memory (SM) and non-sparse memory. Samples reconstructed through memory modules tend to be close to the normal samples used for learning. Therefore, the abnormal sample is reconstructed close to the normal sample, which means that the reconstruction error value for the abnormal inputs increases.

III. NETWORK ANOMALY DETECTION BASED ON MEMORY-AUGMENTED AUTOENCODER

To minimize the overall reconstruction error, the retained information is required to be as relevant as possible to the dominant instances [3]. Therefore, AE is trained to minimize reconstruction errors using only normal class data that can be easily collected. As a result, the encoder network learns only hidden representations for normal samples, and thus it is difficult to adequately reconstruct abnormal behaviors such as attacks. Therefore, the data reconstruction error can be used directly as an anomaly score. Nevertheless, in the AE model, some attack samples are reconstructed very well. The reason for this problem is that the auto-encoder is trained to be over-generalized, or in a latent vector compressed for reconstruction, where the attack sample and the normal sample share some common construction patterns

To solve this problem, we propose a Network Anomaly Detection study using MemAE, as shown in Fig. 3. The study aims to improve performance by reducing the reconstruction performance of attack instances. After training, the memory module transforms the latent vector of the attack input through the most relevant normal memory items so that the reconstruction of the decoder is closer to the normal instance. Fig. 3 shows that the latent vector of the attack sample, which is expressed in red, passes through the memory module and then switches to output close to the normal sample, which is expressed in yellow so that the decoder can only receive inputs related to the normal sample. The result is an increased anomaly score for attack instances, making classification easier.

After that, the next goal is to find an appropriate threshold that can separate the two classes. It is not desirable to obtain the thresholds directly for the test set, so the thresholds were calculated from the validation set and used in the test phase. If MemAE is well-trained, the threshold value obtained from the validation set will show good results in the testing process as well.

A. DATASET PREPROCESSING

The open network intrusion detection datasets for training the model contain symbol data that cannot be used as input. These datasets also contain missing and infinite values. The process of pre-processing NIDS datasets containing these flaws is as follows:

1) Drop missing values: removed all instances of data containing missing values by confirming that the ratio of the missing values was not large in each dataset.

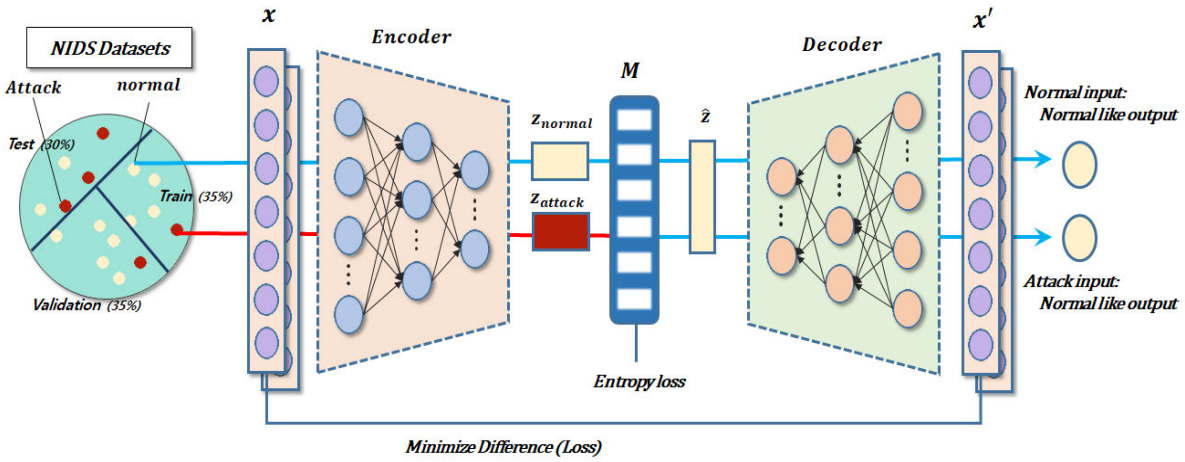


FIGURE 3. Network anomaly detection using memory-augmented autoencoder.

2) Replace infinite values: replace with the maximum value for the columns containing the infinity values of that label class.
3) Remove single-value columns: if all the values in the columns are the same, it will not affect the learning, but it will be removed as the input data dimension increases. (e.g., The column 'num_outbound_cmds' in nsl-kdd consists of all zero values)
4) One-hot encoding: the symbol data were represented through one-hot encoding because all of the symbol data had to be changed to a real vector.
5) Minmax normalization: the numeric data scaled all data to [0, 1] with min-max normalization.

Through the above pre-process, the NSL-KDD dataset, the UNSW-NB15 dataset, and the CICIDS2017 dataset were finally transformed into features of 121 dimensions, 196 dimensions, and 70 dimensions.

B. TRAINING MEMAE

MemAE Loss consists of reconstruction loss and entropy loss. Given the training set $D = \{x_i | i = 1, 2, 3 \cdots, T\}$ containing T samples, reconstruction loss is the distance between a given input x_i and its reconstructed $\hat{x}_i$. The reconstruction error on each sample is minimized as follows:

$$L_{rec}(x, \hat{x}) = \left\| x - \hat{x} \right\|_2^2 \tag{7}$$

where the ℓ 2 -norm is used to measure the reconstruction error. Entropy loss is used to promote the sparsity of the generated addressing weights during training. Entropy loss minimizes a sparsity regularizer with a shrinkage operation during the training phase, and is as follows:

$$L_{entropy}(\hat{w}) = \sum_{i=1}^{T} \hat{w}_i \cdot \log(\hat{w}_i) \tag{8}$$

Finally, the loss function to train the MemAE model is a combination of construction loss and entropy loss, as follows:

$$L = \frac{1}{T} \sum_{i=1}^{T} \left(L_{rec}\left(x, \hat{x}\right) + \alpha L_{entropy}(\hat{w}) \right) \tag{9}$$

where the α is a hyper-parameter that determines the importance between two different losses. During the training phase, MemAE is trained using only the normal dataset of the training dataset and memory M records the prototypical patterns of normal data used in the learning through gradient descents and backpropagation.

C. DETECTING ATTACKS

During the testing phase, the trained MemAE with a normal dataset uses the learned memory that was fixed without updating it. MemAE models have low reconstruction errors for new data (normal) similar to the inputs used in training and expect high reconstruction errors if non-similar data (attacks) are used as inputs, thus defining ℓ 2-norm as an anomaly score. Subsequently, a threshold θ is required to detect normal patterns and attacks from the anomaly score. The threshold is determined as the n-th percentile from the reconstruction error values of the normal sample. Therefore, it is considered normal if the anomaly score is small based on the threshold, or an attack if it is large.

The threshold is determined by exploring the values with the best F1-Score among the reconstruction loss percentiles of normal samples in the validation set. However, even based on the F1 score, it was found that when the difference in ratio between the two classes was very large, the obtained threshold value was biased toward many classes. Therefore, the threshold to detect an attack properly is calculated by sampling the same number of samples from both classes from the validation set. Fig. 4 shows the anomaly score distribution of the MemAE models trained on NSL-KDD datasets, with threshold values found through percentiles and shows that the two classes are very separated.

IV. EXPERIMENTS

In this section, we confirm the validity of the MemAE model in network intrusion detection environments. Experiments were conducted on three datasets, and the results were

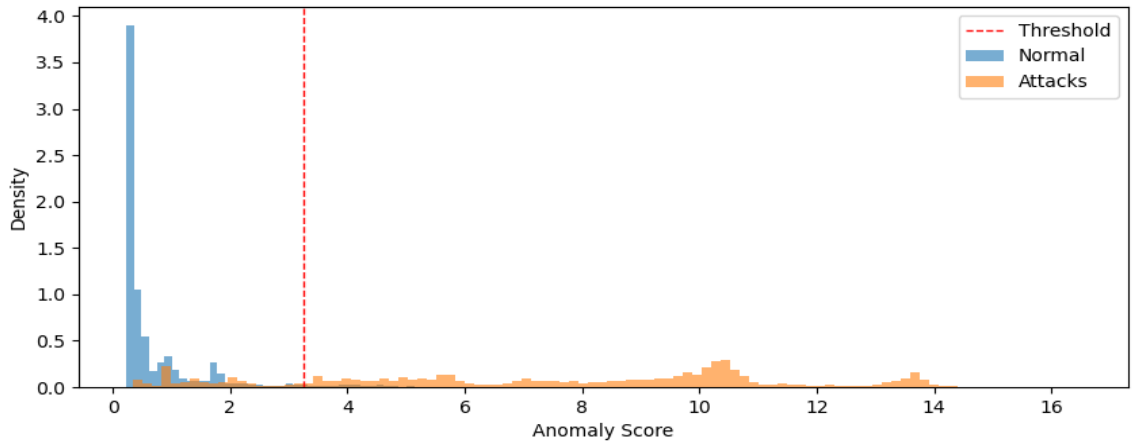


FIGURE 4. Histogram of the NSL-KDD dataset anomaly score in the MemAE model.

compared with the AE and OCSVM models. The two models used for comparison are unsupervised learning models that are trained only using normal data, the same as MemAE, and thus correspond to the same One-class Anomaly Detection methodology. There are two types of MemAE models used in the experiment: the non-sparse MemAE model and the sparse MemAE model. The two types of models are compared to ensure that sparse encoding works effectively in network intrusion detection problems. All of the deep learning models used in the experiment were implemented via Keras, and the parameters used in the learning are shown in Table 2. The memory size N was determined through a grid search, and the values used are shown in Table 2.

TABLE 2. Parameters used in IHE experiments.

Parameter	Value
Epoch & Batch Size	100 / 64
Activation Function	LeakyReLU (hidden) / Linear (output)
Optimizer & Learning Rate	Adam /1e-4
N (memory size)	[100, 300, 500, 1000, 2000]
λ (shrinkage threshold)	1 / N
α (loss weight)	1e-4
Loss Function	Mean Squared Error & Entropy Loss

The model used in the experiments is composed of an encoder and a decoder in a symmetrical structure based on a latent vector. Hidden layers of the encoder and decoder were composed of fully-connected layers, and the Leaky-ReLU function was used for the activation function. Configuration of the hidden layer is constructed by increasing in multiples from the dimension of latent vector to 512. The output layer uses a linear function as an activation function. The latent vector dimension was 64 in the NSL-KDD UNSW-NB15 dataset and 32 in the CICIDS 2017 dataset.

A. DATASETS

Although there are currently many open datasets for network intrusion detection, in this paper, we conducted the experiments with the NSL-KDD dataset, the UNSW-NB15 dataset, and the CICIDS2017 dataset. The NSL-KDD dataset is a proposed dataset by M. Tavallaee *et al.* [8] that improves the KDD CUP 99 dataset created through the Defense Advanced Research Projects Agency (DARPA) intrusion detection evaluation program, which includes four types of simulated attacks: denial of service (DoS), user-to-user root (U2R), remote local-to-local (R2L), and probe. The NSL-KDD dataset is more suitable for the evaluation of intrusion detection models as it removes duplication and redundant records from the KDD Cup 99. The NSL-KDD dataset has been used by many researchers for network intrusion detection, but it has a problem with not reflecting modern network traffic and footprints of intrusion because the dataset is outdated. For this reason, we additionally used the more modern UNSW-NB15 dataset and the CICIDS 2017 dataset in the experiments. The UNSW-NB15 dataset was created by the IXIA PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS), which contains nine attacks: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. Relatively, the CICIDS2017 dataset is the most recent compared to the two datasets introduced above. The CICIDS 2017 dataset is a five-day collection of normal and attacks traffic data from the Canadian Institute of Cybersecurity. The collected data is divided into a total of 8, and the attacks made over the five days differ in types for each day of the week.

The NSL-KDD dataset and the UNSW-NB15 dataset provided a training set and testing set in pairs, so they were used as they are as shown in Table 3. CICIDS 2017 data was used by combining all data provided by day of the week.

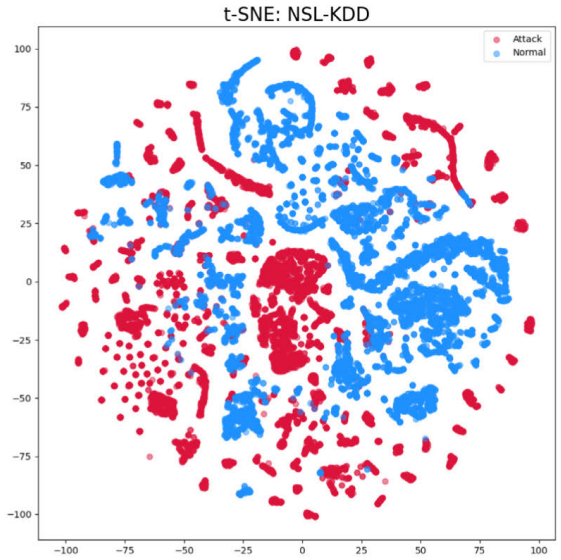


FIGURE 5. t-SNE visualization for NSL-KDD.

TABLE 3. Configuration of dataset used in experiments.

Dataset	Training set		Testing set	
	Normal	**Attack**	**Normal**	**Attack**
NSL-KDD	67343	58630	12833	9711
UNSW-NB15	119341	56000	45332	37000
CICIDS 2017	1590881	265817	681807	113922

30% of the training dataset was used as a validation set, which was used to find the threshold. In the remaining 70%, only normal data were extracted and used for training. Since the CICIDS 2017 dataset has a dominant number of normal samples, 300,000 instances are sampled and used for training. Additionally, the CICIDS 2017 dataset used only DoS attacks for evaluation.

Network traffic data is represented by high-dimensional vectors and can be visualized thru various techniques, such as t-distributed Stochastic Neighbor Embedded (t-SNE). The t-SNE techniques can embed in a low dimension while preserving the ''neighbor structure'' between data represented by a high-dimensional vector. Figs. 5 and 6 show a visualization of the NSL-KDD dataset and the UNSW-NB15 dataset through t-SNE techniques, which shows that normal samples and attack samples share some of the same feature spaces, making linear separation impossible. This shows the difficulty of anomaly detection problems in network traffic.

B. PERFORMANCE EVALUATION

The network traffic anomaly detection problem can be defined as a binary classification problem. In binary classification problems, the Receiver Operator Characteristic (ROC) Curve is commonly used to evaluate the model. The ROC curve is a graph showing the performance of the classification

TABLE 4. Confusion matrix.

		Actual	
		Positive	**Negative**
Predicted	**Positive**	TP	FP
	Negative	FN	TN

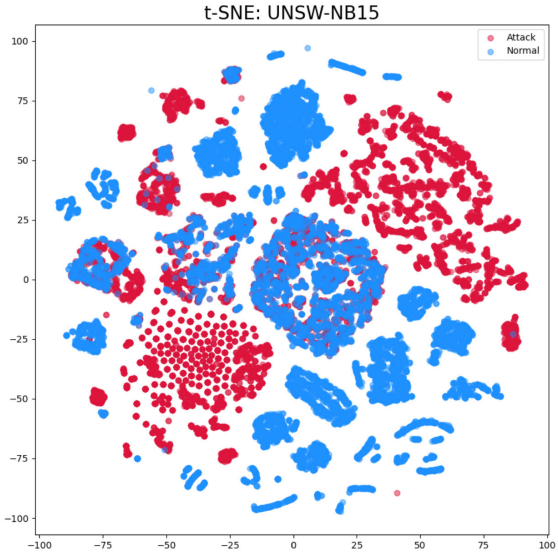


FIGURE 6. t-SNE visualization for UNSW-NB15.

model at all classification thresholds, by plotting the true positive rate (TPR) against the false positive rate (FPR). Area Under Curve (AUC) can be used as a way to quantitatively evaluate the ROC Curve. AUC is an indicator of the area under the curve, and the closer it is to one, the more likely it is to be judged as a perfect classifier. When this is applied to the ROC curve, it is called Area Under the Receiver Operator Characteristic (AUROC).

In addition, the F1 score is used as an evaluation metric and is used to evaluate the classification performance in the acquisition of the threshold and the testing phase of the model. The F1 Score means the harmonized average of precision and recall and is a metric mainly used for accurate evaluation in imbalanced data. These metrics are calculated based on the confusion matrix in Table 4, and are as follows:

$$TPR = \frac{TP}{TP + FN} \tag{10}$$

$$FPR = \frac{FP}{FP + TN} \tag{11}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{12}$$

$$Precision = \frac{TP}{TP + FP} \tag{13}$$

$$Recall = \frac{TP}{TP + FN} \tag{14}$$

$$F_1 - Score = 2\frac{Precision \times Recall}{Precision + Recall} \tag{15}$$

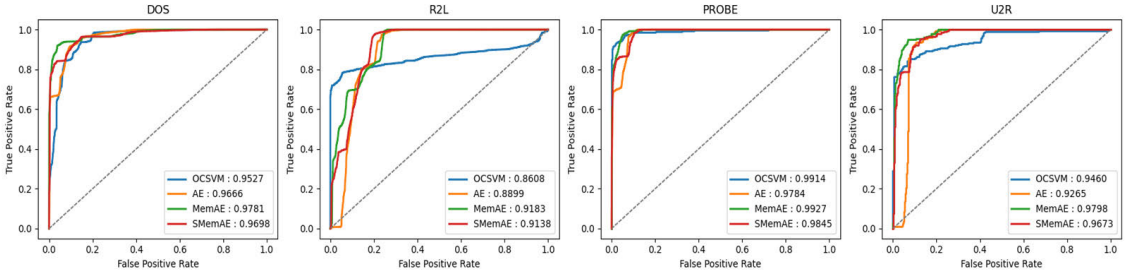


FIGURE 7. Evaluation value of the ROC curve of each model for the NSL-KDD test set.

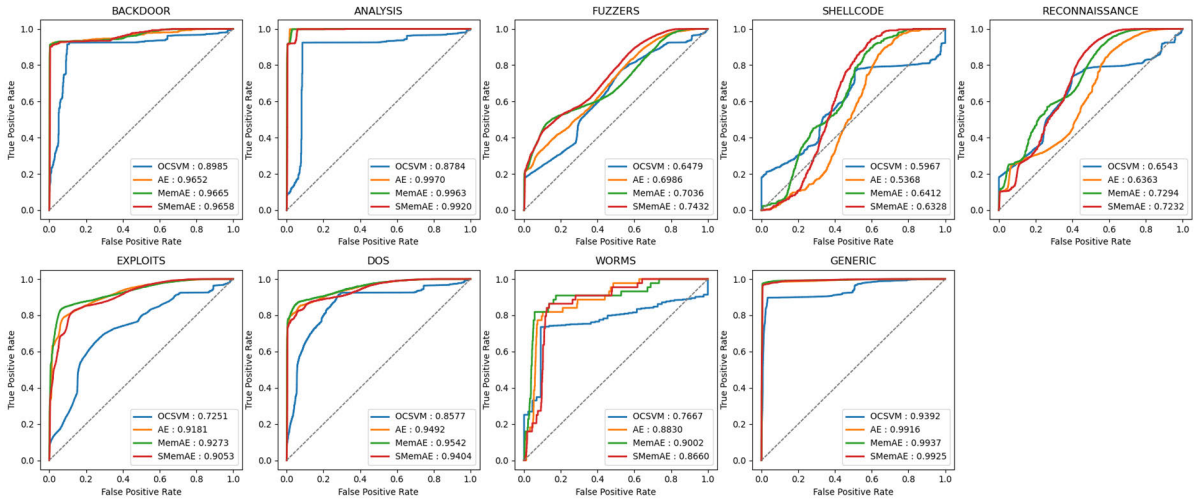


FIGURE 8. Evaluation value of the ROC curve of each model for the UNSW-NB15 test set.

C. EXPERIMENT RESULTS

The experiments measure classification performance results over thresholds determined by a validation set of each data and check the AUROC metric to see if each model is sensitive to changes in thresholds. In addition, by analyzing the performance metrics for each attack, we also confirmed attacks that were difficult for the model to detect. Finally, we also verified that sparse encoding works efficiently.

Table 5 shows the AUROC performance of each model for the entire set of network traffic data used in the experiment. The SparseMemAE model is an autoencoder using the sparse memory module, which shows good performance compared to the OCSVM model and the AE model but does not show better performance compared to the MemAE model. These results are thought to use a network consisting of fully-connected layers and are expected to show good results only when used with models capable of extracting various features, such as convolutional networks. In all experiments, OCSVM showed the lowest performance for the entire dataset, and the MemAE model showed the best performance.

Figs. 7, 8, and 9 show the ROC Curve and AUROC values for each attack for each dataset, which can be used to identify which attack techniques are difficult to detect. These results show that the attacks that each model finds difficult to detect are similar. The closer the ROC curve is to the upper left, the better the model, and the maximum value is one when calculating the AUC value. The MemAE model generally shows an AUROC value of 0.9 for all attacks, indicating that the distribution of normal and attack reconstruction errors are very far apart. It also means that the performance does not react sensitively to small changes in the determined threshold. Therefore, the thresholds determined from the validation set will also operate as good decision boundaries during testing. Tables 6, 7, and 8 show the classification results of the model according to these decision boundaries.

TABLE 5. AUROC performance for NIDS datasets.

Models	AUROC		
	NSL-KDD	**UNSW-NB15**	**CICIDS2017**
OCSVM	0.9402	0.8139	0.7684
AE	0.9517	0.8990	0.8758
MemAE	0.96.81	0.9113	0.9101
SparseMemAE	0.9605	0.9088	0.8961

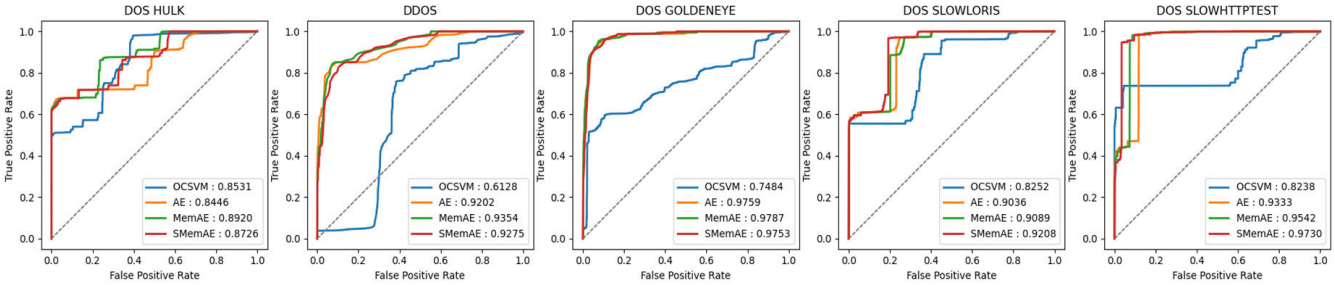


FIGURE 9. Evaluation value of the ROC curve of each model for the CICIDS 2017 test set.

TABLE 6. Attack detection result in NSL-KDD.

Attacks	Threshold θ=3.2467 / N=100		
	Precision	Recall	F1-Score
DoS	0.9423	0.9423	0.9422
Probe	0.9600	0.9570	0.9578
R2L	0.8513	0.8586	0.8469
U2R	0.9839	0.9579	0.9677

Table 6 shows the results of the classification performance in the test phase of MemAE, using the thresholds determined in the validation set for the NSL-KDD dataset. Thresholds were explored for the entire attack and were not individually calculated for each attack. In the test phase, the average reconstruction error of the normal samples was about 0.92, and the standard deviation was about 1.01. In comparison, the average of the attack samples was about 10.28 and the standard deviation was about 3.46, confirming that there was quite a difference between the two groups. The threshold obtained from the validation set is about 1.79, corresponding to the 96th percentile of the reconstruction error value of the normal samples. If the normal and attack reconstruction errors were completely separated, the 100th percentile would be optimal. This means that the closer that determined thresholds are to the 100th percentile, the better the model can separate the two groups. Fig. 4 is a graph that visualizes these results and shows that the reconstruction errors of normal and abnormal classes are easy to visualize separately. From the results in Table 6, the proposed MemAE model can generally detect all attacks well in NSL-KDD and ensured that the threshold is valid.

Table 7 shows the results of the classification performance in the test phase of MemAE, using the thresholds determined in the validation set for the UNSW-NB15 dataset. In the test phase, the average reconstruction error of the normal samples was about 0.72, and the standard deviation was about 0.62. In comparison, the average of the attack samples was about 4.79 and the standard deviation was about 2.69, confirming that the two groups were not well separated compared to NSL-KDD. The threshold used for the test is about 1.79,

TABLE 7. Attack detection result in UNSW-NB15.

Attacks	Threshold θ=1.7972 / N=100		
	Precision	Recall	F1-Score
Analysis	0.9859	0.9349	0.9547
Backdoor	0.9862	0.9337	0.9549
DoS	0.9462	0.9277	0.9336
Exploits	0.9127	0.9107	0.9115
Fuzzers	0.8397	0.8535	0.8453
Generic	0.9567	0.9523	0.9529
Reconnaissance	0.8724	0.8750	0.8737
Shellcode	0.9796	0.9246	0.9512
Worms	0.9986	0.9336	0.9645

corresponding to the 96th percentile threshold for the normal reconstruction error. We confirmed that Fuzzers and Reconnaissance attacks showed relatively low performance compared to other attacks, and the AUROC values for both attacks were 0.7 and 0.64, as shown in Fig. 7. However, while the shellcode attack also has a low AUROC value, the actual detection performance is high. This is because the number of two class samples is very different from 27900 to 59, so the detection performance is biased as the normal class. The proposed model detects all of the above three attacks well. In addition, as shown in Table 5, when the model was evaluated with AUROC, it was confirmed that it is about 0.15 better than OCSVM.

Table 8 shows experimental results for the CICIDS 2017 dataset and shows the detection performance for each dos technique. In the test phase, the average reconstruction error of the normal samples was about 0.18, and the standard deviation was about 0.22. On the other hand, the average of the attack samples was about 2.83 and the standard deviation was about 2.47, confirming that the two groups were not well separated compared to NSL-KDD. The threshold obtained from the validation set is about 0.41, corresponding to the 92th percentile of the reconstruction error value of the normal samples. We confirmed that the proposed model has high performance against all dos attacks. Fig. 8 shows that the

TABLE 8. Attack detection result in CICIDS 2017.

Attacks	Threshold θ=0.4125 / N=300		
	Precision	Recall	F1-Score
DDoS	0.9578	0.9162	0.9309
DoS goldeneye	0.9955	0.9202	0.9544
DoS hulk	0.9194	0.8978	0.9062
DoS slowhttptest	0.9975	0.9202	0.9561
DoS slowloris	0.9964	0.9193	0.9555

TABLE 9. Comparison of experimental results with other existing studies on the NSL-KDD dataset.

Methods	Metrics			
	Accuracy	Precision	Recall	F1-Score
RNN-IDS [21]	0.8328	-	-	-
DCNN [22]	0.8500	-	-	-
Sparse AE and MLP [23]	0.8839	0.8544	0.9595	0.9040
RandomTree [24]	0.8846	-	0.8260	-
AE [18]	0.8828	0.9123	0.8786	0.8951
DAE [18]	0.8865	0.9648	0.8308	0.8928
LSTM [22]	0.8900	-	-	-
MemAE	0.8951	0.9062	0.8951	0.8993

TABLE 10. Comparison of experimental results with other existing studies on the UNSW-NB15 dataset.

Methods	Metrics			
	Accuracy	Precision	Recall	F1-Score
NB [25]	0.7639	0.7820	0.7640	0.7729
SVM [26]	0.8190	0.9730	0.7560	0.8210
KNN [25]	0.8449	0.8550	0.8450	0.8500
RF [25]	0.8363	0.8690	0.8360	0.8522
FCN [26]	0.8570	0.7010	0.9380	0.8070
MemAE	0.8530	0.8774	0.8530	0.8526
SSAE [27]	-	0.7310	0.9630	0.8320
VLSTM [27]	-	0.8600	0.9780	0.9070

TABLE 11. Comparison of time spent on learning the AE and MemAE parameters.

Methods	64 Embedding Dimension	
	Learning Time (s)	Trainable Parameters
AE	107.12	469,945
MEM 100	109.47	476,345
MEM 500	114.25	501,945
MEM 1000	115.18	533,945
MEM 3000	116.49	661,945
MEM 5000	121.76	789,945

OCSVM model shows low AUROC performance against DDoS attacks, but the AE-based models show high overall performance. This shows that AE-based models work well in the network anomaly detection problem.

Tables 9 and 10 show the results compared to other existing studies in the NSL-KDD and UNSW-NB15 datasets. CICIDS2017 dataset was excluded because the results were confirmed only for DOS attacks. In the NSL-KDD dataset, it was found that the proposed method showed the best performance when evaluated based on accuracy, and it was confirmed that the Sparse AE and MLP model showed better performance when evaluated based on the F1 score. In addition, it was confirmed from Table 9 that the proposed model is better even for the AE and DAE methods. Both models are AE-based studies conducted in other previous studies. In the UNSW-NB15 data, compared with previous studies, the VLSTM model showed the best performance when evaluated based on the F1 score. Since accuracy was not provided, the comparison was not possible, and it was confirmed that the proposed model shows high performance compared to the rest of the models. In addition, we confirmed that most of the existing models have precision and recall metrics skewed to one side, whereas the proposed model sets a threshold based on the f1 score, so there is no significant difference between the two metrics.

In deep learning, most architectures use similar computational elements (e.g., convolutional layers and linear layers). Thus, it is a convention to use the number of parameters as a stand-in for complexity. Table 11 shows the time spent for the AE model and MemAE model to train 47140 normal samples of the NSL-KDD train set and the number of parameters used for model training. The difference in time spent on large learning according to the memory size is not large, and this is the same as the memory trainable parameter multiplied by the memory size and the embedding dimension.

V. CONCLUSION

In this study, we pointed out that over-generalization problems may occur in AE-based models commonly used in network anomaly detection domains, and to solve this problem, we proposed a method using the Memory-augmented Deep Autoencoder (MemAE) method. The reason for this problem is that the auto-encoder is trained to be over-generalized, or in a latent vector compressed for reconstruction, where the attack sample and the normal sample share some common

reconstruction patterns. MemAE solves this problem by bringing the reconstruction of the attack inputs closer to the normal sample through the memory module. Experiments were conducted using the NSL-KDD dataset, which has been widely used in the past, and the UNSW-NB15 and CICIDS 2017 datasets to reflect recent attack environments. First, we evaluated the model based on the AUROC values of all models, and the results confirmed that the MemAE model performed better for all datasets. In addition, it was confirmed that the SparseMemAE model using sparse addressing showed better performance than the OCSVM and AE-based models, but did not show better performance than the basic MemAE model. We analyzed the result as the cause of the narrow feature extraction ability of the fully-connected layer. Overall, it was confirmed that the proposed model has an AUROC value of at least 0.9 for attacks of all datasets. Second, we evaluated the classification performance of the test set through the threshold value. The threshold value is obtained through the validation set, and the classification performance evaluated through this is related to the previously evaluated AUROC score. This is because a model with an AUROC value close to 1 is not sensitive to subtle changes in the threshold value, so the obtained threshold value acts as a valid decision boundary even during the test. Unlike the NSL-KDD test set, the UNSW-NB15 and CICIDS 2017 test sets are unbalanced data. Therefore, we evaluated the classification results for each data based on F1-Score, which can be usefully used when evaluating unbalanced data. The classification result was about 95% for the NSL-KDD dataset, about 83% for the UNSW-NB15 dataset, and about 88% for the CICIDS 2017 dataset.

REFERENCES

[1] L. N. Tidjon, M. Frappier, and A. Mammar, "Intrusion detection systems: A cross-domain overview," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3639–3681, 2019, doi: 10.1109/comst.2019.2922584.

[2] O. Depren, M. Topallar, E. Anarim, and M. K. Ciliz, "An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks," *Expert Syst. Appl.*, vol. 29, no. 4, pp. 713–722, 2005, doi: 10.1016/j.eswa.2005.05.002.

[3] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection," *ACM Comput. Surv.*, vol. 54, no. 2, pp. 1–38, Apr. 2021, doi: 10.1145/3439950.

[4] R. Longadge and S. Dongre, "Class imbalance problem in data mining review," 2013, *arXiv:1305.1707*. [Online]. Available: http://arxiv.org/abs/1305.1707

[5] S. Barua, M. M. Islam, X. Yao, and K. Murase, "MWMOTE–majority weighted minority oversampling technique for imbalanced data set learning," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 2, pp. 405–425, Feb. 2014, doi: 10.1109/tkde.2012.232.

[6] L. M. Manevitz and M. Yousef, "One-class SVMs for document classification," *J. Mach. Learn. Res.*, vol. 2, pp. 139–154, Dec. 2001, doi: 10.5555/944790.944808.

[7] D. Gong, L. Liu, V. Le, B. Saha, M. R. Mansour, S. Venkatesh, and A. Van Den Hengel, "Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Oct. 2019, pp. 1705–1714, doi: 10.1109/iccv.2019.00179.

[8] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, Jul. 2009, pp. 1–6, doi: 10.1109/cisda.2009.5356528.

[9] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6, doi: 10.1109/milcis.2015.7348942.

[10] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, 2018, pp. 108–116, doi: 10.5220/0006639801080116.

[11] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 12. 1999, pp. 582–588.

[12] J. L. Leevy and T. M. Khoshgoftaar, "A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 big data," *J. Big Data*, vol. 7, no. 1, pp. 1–19, Dec. 2020, doi: 10.1186/s40537-020-00382-x.

[13] Y. Yang, K. Zheng, C. Wu, and Y. Yang, "Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network," *Sensors*, vol. 19, no. 11, p. 2528, Jun. 2019, doi: 10.3390/s19112528.

[14] J. Kim, J. Kim, H. Kim, M. Shim, and E. Choi, "CNN-based network intrusion detection against denial-of-service attacks," *Electronics*, vol. 9, no. 6, p. 916, 2020, doi: 10.3390/electronics9060916.

[15] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4393–4402.

[16] D. M. J. Tax and R. P. W. Duin, "Support vector data description," *Mach. Learn.*, vol. 54, no. 1, pp. 45–66, 2004, doi: 10.1023/B:MACH.0000008084.60811.49.

[17] Y. Tian, M. Mirzabagheri, S. M. H. Bamakan, H. Wang, and Q. Qu, "Ramp loss one-class support vector machine; a robust and effective approach to anomaly detection problems," *Neurocomputing*, vol. 310, pp. 223–235, Oct. 2018, doi: 10.1016/j.neucom.2018.05.027.

[18] R. C. Aygun and A. G. Yavuz, "Network anomaly detection with stochastically improved autoencoder based models," in *Proc. IEEE 4th Int. Conf. Cyber Secur. Cloud Comput. (CSCloud)*, Jun. 2017, pp. 193–198, doi: 10.1109/cscloud.2017.39.

[19] L. Mhamdi, D. McLernon, F. El-Moussa, S. A. R. Zaidi, M. Ghogho, and T. Tang, "A deep learning approach combining autoencoder with one-class SVM for DDoS attack detection in SDNs," in *Proc. IEEE Int. Conf. Commun. Netw.*, Oct. 2020, pp. 1–6, doi: 10.1109/ComNet47917.2020.9306073.

[20] S. Zavrak and M. İskefiyeli, "Anomaly-based intrusion detection from network flow features using variational autoencoder," *IEEE Access*, vol. 8, pp. 108346–108358, 2020, doi: 10.1109/access.2020.3001350.

[21] 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, doi: 10.1109/ACCESS.2017.2762418.

[22] S. Naseer, Y. Saleem, S. Khalid, M. K. Bashir, J. Han, M. M. Iqbal, and K. Han, "Enhanced network anomaly detection based on deep neural networks," *IEEE Access*, vol. 6, pp. 48231–48246, 2018, doi: 10.1109/access.2018.2863036.

[23] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proc. 9th EAI Int. Conf. Bio-Inspired Inf. Commun. Technol.*, 2016, pp. 21–26, doi: 10.4108/eai.3-12-2015.2262516.

[24] J. Kevric, S. Jukic, and A. Subasi, "An effective combining classifier approach using tree algorithms for network intrusion detection," *Neural Comput. Appl.*, vol. 28, no. 1, pp. 1051–1058, 2017, doi: 10.1007/s00521-016-2418-1.

[25] F. A. Khan and A. Gumaei, "A comparative study of machine learning classifiers for network intrusion detection," in *Proc. Int. Conf. Artif. Intell. Secur.* Cham, Switzerland: Springer, 2019, pp. 75–86, doi: 10.1007/978-3-030-24265-7_7.

[26] 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 *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Mar. 2018, pp. 893–898, doi: 10.1109/ICCNC.2018.8390278.

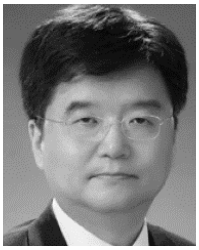
[27] X. Zhou, Y. Hu, W. Liang, J. Ma, and Q. Jin, "Variational LSTM enhanced anomaly detection for industrial big data," *IEEE Trans. Ind. Informat.*, vol. 17, no. 5, pp. 3469–3477, May 2021, doi: 10.1109/TII.2020.3022432.

BYEONGJUN MIN received the B.S. degree in computer science from Seoul Hoseo Technical College, in 2017, and the M.S. degree in computer science from Sejong University, in 2019, where he is currently pursuing the Ph.D. degree. His research interests include anomaly detection, data mining, and reinforcement learning.

JIHOON YOO received the B.S. degree in computer science from Seoul Hoseo Technical College, in 2016, and the M.S. degree in computer science from Sejong University, in 2018, where he is currently pursuing the Ph.D. degree. His research interests include machine learning, data mining, and cyber security.

SANGSOO KIM received the B.S. degree in electronic engineering and the M.S. degree in computer engineering from Kyungpook National University, Daegu, South Korea, in 1997 and 2003, respectively. Since 2003, he has been a Principal Researcher with Agency for Defense Development, South Korea. His research interests include cyber security, machine learning, and situational awareness.

DONGIL SHIN received the B.S. degree in computer science from Yonsei University, Seoul, South Korea, in 1988, the M.S. degree in computer science from Washington State University, Pullman, WA, USA, in 1993, and the Ph.D. degree from the University of North Texas, Denton, TX, USA, in 1997. He was a Senior Researcher with the System Engineering Research Institute, Deajeon, South Korea, in 1997. Since 1998, he has been with the Department of Computer Engineering, Sejong University, South Korea, where he is currently a Professor. His research interests include information security, bio-signal data processing, data mining, and machine learning.



DONGKYOO SHIN received the B.S. degree in computer science from Seoul National University, South Korea, in 1986, the M.S. degree in computer science from Illinois Institute of Technology, Chicago, IL, USA, in 1992, and the Ph.D. degree in computer science from Texas A&M University, College Station, TX, USA, in 1997. From 1986 to 1991, he was with Korea Institute of Defense Analyses, where he developed database application software. From 1997 to 1998, he was a Principal Researcher with the Multimedia Research Institute, Hyundai Electronics Company, South Korea. He is currently a Professor with the Department of Computer Engineering, Sejong University, South Korea. His research interests include machine learning, ubiquitous computing, bio-signal data processing, and information security.