An Enhanced AI-Based Network Intrusion Detection System Using Generative Adversarial Networks

Cheolhee Park, Jonghoon Lee, Youngsoo Kim, Jong-Geun Park, Hyunjin Kim, and Dowon Hong

***Abstract*—As communication technology advances, various and heterogeneous data are communicated in distributed environments through network systems. Meanwhile, along with the development of communication technology, the attack surface has expanded, and concerns regarding network security have increased. Accordingly, to deal with potential threats, research on network intrusion detection systems (NIDSs) has been actively conducted. Among the various NIDS technologies, recent interest is focused on artificial intelligence (AI)-based anomaly detection systems, and various models have been proposed to improve the performance of NIDS. However, there still exists the problem of data imbalance, in which AI models cannot sufficiently learn malicious behavior and thus fail to detect network threats accurately. In this study, we propose a novel AI-based NIDS that can efficiently resolve the data imbalance problem and improve the performance of the previous systems. To address the aforementioned problem, we leveraged a state-of-the-art generative model that could generate plausible synthetic data for minor attack traffic. In particular, we focused on the reconstruction error and Wasserstein distance-based generative adversarial networks, and autoencoder-driven deep learning models. To demonstrate the effectiveness of our system, we performed comprehensive evaluations over various data sets and demonstrated that the proposed systems significantly outperformed the previous AI-based NIDS.**

***Index Terms*—Anomaly detection, generative adversarial network (GAN), network intrusion detection system (NIDS), network security.**

I. Introduction

With the development of the fifth-generation (5G) mobile communication technology that diversifies the access environments and constructs distributed networks, various and heterogeneous data are communicated through network systems. In general, these data originate from diverse domains, such as sensors, computers, and the Internet of Things (IoT), and the capacity of network systems has been expanded to process these data reliably. However, as the access points are diversified, the attack surface expands, thereby leaving the network systems vulnerable to potential threats. Moreover, cyber-attack techniques have become more complex and sophisticated, and the frequency of attacks has also increased. Accordingly, the importance of cybersecurity is emphasized, and various studies have been actively conducted to prevent potential network threats.

One of the fundamental challenges in cybersecurity is the detection of network threats, and various results have been reported in the field of network intrusion detection systems (NIDSs). In particular, the most recent studies have been focused on applying the artificial intelligence (AI) technology to NIDS, and AI-based intrusion detection systems have achieved remarkable performance. Initially, the research primarily focused on applying traditional machine learning models, such as decision trees [1] (DTs) and support vector machines [2] (SVMs) to existing intrusion detection systems, and it has now been extended to deep learning approaches [3], such as convolutional neural networks (CNNs), long short-term memory (LSTM), and autoencoders. Although these results have achieved remarkable performance in detecting anomalies, there still exist limitations in deploying them in real systems.

In general, most of the network flow data is normal traffic, and malicious behavior that can cause service failure occurs rarely. Moreover, within the category of malicious behavior, most of the data are well-known attacks, and specific types of attacks are extremely rare. Due to this data imbalance problem, AI models deployed in NIDS cannot sufficiently learn the characteristics of specific network threats, and this may leave the network systems vulnerable to the attacks owing to the poor detection performance.

In this study, to address this inherent problem, we propose a novel AI-based NIDS that can resolve the data imbalance problem and improve the performance of the previous systems. To address the aforementioned problem, we leveraged a state-of-the-art deep learning architecture, generative adversarial networks [4] (GANs), to generate synthetic network traffic data. In particular, we focused on the reconstruction error and Wasserstein distance-based GAN architecture [5], which can generate plausible synthetic data for minor attack traffic. By combining the generative model with anomaly detection models, we demonstrated that the proposed systems

Manuscript received 5 April 2022; revised 14 August 2022; accepted 21 September 2022. Date of publication 3 October 2022; date of current version 24 January 2023. This work was supported by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by the Korea Government (MSIT, Development of 5G Edge Security Technology for Ensuring 5G+ Service Stability and Availability) under Grant 2020-0–00952. *(Corresponding author: Cheolhee Park.)*

Cheolhee Park, Jonghoon Lee, Youngsoo Kim, Jong-Geun Park, and Hyunjin Kim are with the Cyber Security Research Division, Electronics and Telecommunications Research Institute, Daejeon 34129, South Korea (e-mail: chpark0528@etri.re.kr; mine@etri.re.kr; blitzkrieg@etri.re.kr; queue@etri.re.kr; be.successor@etri.re.kr).

Dowon Hong is with the Department of Applied Mathematics, Kongju National University, Gongju 32588, South Korea (e-mail: dwhong@kongju.ac.kr).

Digital Object Identifier 10.1109/JIOT.2022.3211346

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

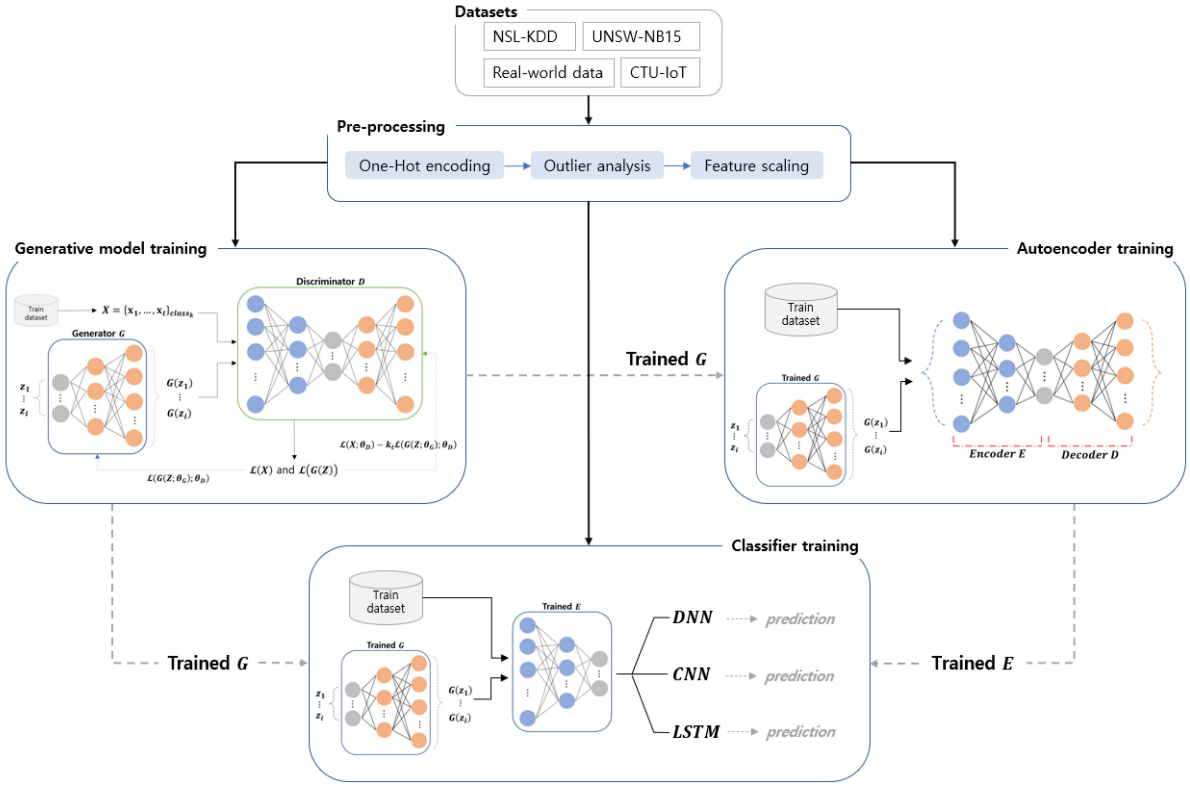

Fig. 1. Entire systemic architecture of our AI-based NIDS.

outperformed previous results in terms of the classification performance.

The entire architecture of our system consists of four main stages (see Fig. 1): 1) preprocessing; 2) generative model training; 3) autoencoder training; and 4) predictive model training. In the preprocessing stage, the system refines the raw data set into a format that deep learning models can learn. After preprocessing, the system sequentially trains generative models and an autoencoder model, where the trained generative models are utilized to train the autoencoder model. Finally, the system trains predictive models by applying the trained generative models and the encoder of the trained autoencoder, where the generative models are used to generate scarce data and the encoder is used as a feature extractor. In the case of the classifier models, we consider three deep learning models that have been widely utilized in AI-based NIDS: 1) deep neural networks (DNNs); 2) CNNs; and 3) LSTM model. To evaluate our system, we experimented with four network flow data sets considering different scenarios: 1) NSL-KDD [6], [7]; 2) UNSW-NB15 [8]; 3) IoT data set [9]; and 4) real-world data set. Through experiments on these various data sets, we show that the proposed system outperformed previous results. Moreover, we demonstrate that our methodology can improve the performance of existing AI-based NIDS by resolving the data imbalance problem.

The main contributions of the proposed approach can be summarized as follows.

1) By combining the state-of-the-art GAN model that can generate plausible synthetic data and measure the convergence of training, we show that the proposed system outperforms existing AI-based NIDS in terms of detection rate.
2) Through comparative experiments with various deep learning models, we present that the detection performance for rare attacks can be improved by applying our methodology it as a base module.
3) By experimenting with data sets collected from various scenarios, we show that the proposed system can be effectively applied to real-world environments.

The remainder of this article is organized as follows. Section II briefly reviews related research from the perspective of NIDS based on machine learning and deep learning approaches, and Section III provides a background with a focus on autoencoders and GANs. In Section IV, we describe our methodology and the proposed framework as well as the four main stages in detail. In Section V, we evaluate the proposed system in various environments and present experimental results with detailed analysis. Finally, we present concluding remarks and future work directions of this study in Section VI.

II. Related Work

In the field of AI-based NIDSs, many studies have been conducted to apply machine learning and deep learning technologies as anomaly detection. Ingre and Yadav [10] proposed multilayer perceptron-based intrusion detection system and showed that the proposed approach achieve 81% and 79.9% accuracy in experiments on the NSL-KDD data set for binary and multiclassification, respectively. Gao et al. [11]

proposed a semi-supervised learning approach for NIDSs based on fuzzy and ensemble learning and reported that the proposed system achieved 84.54% accuracy on the NSL-KDD data set. By applying the deep belief network (DBN) model, Alrawashdeh and Purdy, [12] developed an anomaly intrusion detection system and showed that the proposed DBN-based IDS exhibited a superior classification performance in subsampled testing sets (sampled subsets from the original data set). By considering the software-defined networking environment, Tang et al. [13] proposed a DNN-based anomaly detection system and reported that the DNN-based approach outperformed traditional machine learning model approaches (e.g., Naïve Bayes, SVM, and DT). Imamverdiyev and Abdullayeva [14] proposed a restricted Boltzmann machine (RBM)-based intrusion detection system and showed that the Gaussian–Bernoulli RBM model outperformed other RMB-based models (such as Bernoulli-Bernoulli RBM and DBN). From the perspective of utilizing both behavioral (network traffic characteristics) and content features (payload information), Zhong et al. [15] introduced a big data and tree architecture-driven deep learning system into the intrusion detection system, where the authors combined shallow learning and deep learning strategies and showed that the system is particularly effective at detecting subtle patterns for intrusion attacks. With the ensemble model-like approach, Haghighat et al. [16] proposed an intrusion detection system based on deep learning and voting mechanisms. Haghighat and Li [16] aggregated the best model results and showed that the system can provide more accurate detections. Moreover, they showed that the false alarms can be reduced up to 75% compared to the conventional deep learning approaches. Considering data streams in industrial IoT environments, Yang et al. [17] proposed a tree structure-based anomaly detection system, where the authors incorporate the window sliding, detection strategy changing, and model updating mechanisms into the locality-sensitive hashing-based iForest model [18], [19] to handle the infiniteness of data streams in real-time scenario. Similarly, Qi et al. [20] proposed an intrusion detection system for multiaspect data streams by combining locality-sensitive hashing, isolation forest, and principal component analysis (PCA) techniques. Qi et al. [20] showed that the proposed system can effectively detect group anomalies while dealing with multiaspect data and process each data row faster than the previous approaches.

From the perspective of dealing with time-series data, several results have been reported focusing on recurrent models. Kim et al. [21] proposed an LSTM-based IDS model and proved the efficiency of the proposed IDS. Yin et al. [22] proposed a recurrent neural network-based intrusion detection system and achieved 83.3% accuracy and 81.3% accuracy in binary and multiclassification, respectively. Xu et al. [23] developed a recurrent neural network-based intrusion detection model and reported that the gated recurrent unit was more suitable as a memory unit for intrusion detection than the LSTM unit. By considering supervisory control and data acquisition (SCADA) networks, Gao et al. [24] proposed an omni-intrusion detection system. Gao et al. [24] combined LSTM and a feedforward neural network through an ensemble approach and showed that the proposed system can effectively detect intrusion attacks regardless of temporal correlation. Moreover, they demonstrated that the proposed omni-IDS outperformed previous deep learning approaches through experiments on a SCADA testbed.

In addition to the previous approach of applying supervised learning as an anomaly detection model, several studies have focused on the application of unsupervised learning, especially autoencoder models. Javaid et al. [25] proposed a sparse autoencoder-based NIDS and reported that the proposed model achieved 79.1% accuracy for multiclassification on the NSL-KDD data set. Similarly, Yan and Han [26] leveraged the sparse autoencoder model to extract high-level feature representations of intrusive behavior information and demonstrated that the stacked sparse autoencoder model could be applied as an efficient feature extraction method. Shone et al. [27] proposed a stacked nonsymmetric deep autoencoder-based intrusion detection system. Shone et al. [27] showed that the proposed model could achieve 85.42% accuracy in multiclassification. As one of the significant results, Ieracitano et al. [28] proposed an autoencoder-driven intrusion detection model. Ieracitano et al. [28] proposed autoencoder-based and LSTM-based IDS models and compared their performance with conventional machine learning models. Through experiments on the NSL-KDD data set, they reported that the proposed autoencoder-based systems outperformed other models and achieved 84.21% and 87% accuracy for binary and multiclassification, respectively.

As another approach to applying unsupervised learning, several studies have investigated using generative models to improve the performance of existing NIDS. In particular, they have focused on applying the basic GANs [4], which are based on the Jensen–Shannon divergence (or Kullback–Leibler divergence) [29], [30], [31]. Thereafter, along with the development of various GAN models, studies have been conducted to apply appropriate GAN models for specific purposes. Li et al. [32] and Lee et al. [33] utilized the Wasserstein divergence-based GAN model to generate the synthetic data, and Dlamini et al. [34] proposed a conditional GAN-based anomaly detection model to improve the classification performance in the minority classes. By focusing on specific industrial environments, Li et al. [35] and Alabugin and Sokolov [36] proposed LSTM-GAN and bidirectional GAN-based anomaly detection models, respectively. Through experiments on the secure water treatment (SWaT) data set, they demonstrated that GAN models could be effectively applied to IDS. Siniosoglou et al. [37] proposed an anomaly detection model that could simultaneously detect anomalies and categorize the attack types. Siniosoglou et al. [37] encapsulated the autoencoder architecture into the structure of the basic GAN model (i.e., deploying the encoder as a discriminator and the decoder as a generator) and proved the efficiency of the proposed model in various smart grid environments.

Unlike previous GAN approaches that are based on the distance between data distributions, we considered the reconstruction error-based GAN model to generate more plausible synthetic data. In particular, we leveraged the boundary equilibrium GAN (BEGAN) model [5], which is based on the

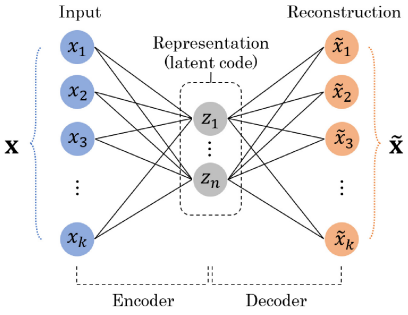

Fig. 2. Basic architecture of autoencoder.

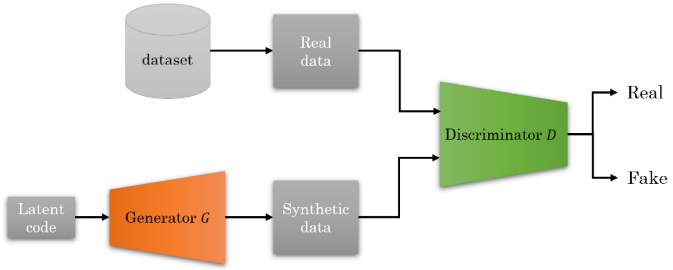

Fig. 3. Basic architecture of generative adversarial networks.

concept of autoencoders and the Wasserstein distance between reconstruction error distributions of samples (real and synthetic samples). Moreover, we incorporated the autoencoder model into the detection models to extract meaningful features from the data and extend the adaptability and demonstrated that the proposed framework outperforms previous AI-based network intrusion detection models.

III. Background

In this section, we briefly illustrate the concepts of autoencoders and GAN, which are key components of our anomaly detection system.

A. Autoencoder

The autoencoder [38], [39] is one of the fundamental deep learning models and is trained with an unsupervised learning process. The objective of autoencoders is to return the output as close to the original input as possible. Therefore, the parameters are updated progressively during the training process to minimize the reconstruction error. In general, the architecture of an autoencoder consists of two components: 1) an encoder and 2) a decoder (see Fig. 2). The encoder is responsible for mapping the given raw input data $\mathbf{x}$ into the latent space of representation

$$\mathbf{z} = f(\mathbf{x}W + b) \tag{1}$$

where f denotes the activation function of the encoder, and W and b represent the weight matrix and the bias vector, respectively. Conversely, the decoder plays the role of reconstructing the representation $\mathbf{z}$ into the corresponding input data as close as possible (i.e., $\tilde{\mathbf{x}}$)

$$\tilde{\mathbf{x}} = g\big(\mathbf{z}W' + b'\big) \tag{2}$$

where g denotes the activation function of the decoder, and W' and b' are the weight matrix and the bias vector, respectively. Therefore, the autoencoder is trained to minimize the reconstruction error $\mathcal{L}_{\mathrm{RE}}$

$$\begin{aligned}\mathcal{L}_{\mathrm{RE}}\big(\mathbf{x}, \tilde{\mathbf{x}}; W, W'\big) &= \|\mathbf{x} - \tilde{\mathbf{x}}\|_2^2 \\ &= \|\mathbf{x} - g\big(W' \cdot f(\mathbf{x}W + b) + b'\big)\|. \end{aligned} \tag{3}$$

One of the fundamental characteristics of the autoencoder is to represent high-dimensional input data as lower dimensional information (summarized but meaningful information). Herein, we utilized autoencoders with the aim of feature extraction (dimension reduction) on the input data. Although PCA has traditionally been utilized to project high-dimensional data into a lower dimensional space, we leveraged the autoencoders for nonlinear transformations on complex data sets. Although we only present the basic architecture of autoencoders, models can be built in multiple layers and an asymmetric manner.

B. Generative Adversarial Networks

Generative models are designed to approximate the probability distribution of a training data set and aim to generate synthetic data that is close to the real data (training data). Recently, among these generative models, research on GAN [4] has been of significant interest. Accordingly, various GAN models have been proposed to improve the performance and advance functionality (e.g., [40], [41], and [42]). A GAN model consists of two neural network-based models: 1) a generator G and 2) a discriminator D (see Fig. 3). The generator G aims to generate synthetic data (fake data) that is close to the real data, while the discriminator D aims to discriminate between the real and fake data. In other words, these two components have opposing objectives during the training process.

More formally, let $p_{\mathbf{z}}$ and p_{data} be the probability distributions of the latent code and the real data, respectively. Then, the objective function $V(D, G)$ of a GAN that consists of a generator G and a discriminator D is a minimax game and can be formulated as follows:

$$\begin{aligned}V(D, G) = \min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}}\big[\log\big(D_{\theta_D}(\mathbf{x})\big)\big] \\ + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}\big[\log\big(1 - D_{\theta_D}\big(G_{\theta_G}(\mathbf{z})\big)\big)\big]\end{aligned} \tag{4}$$

where θ_D and θ_G denote the model parameters of D and G, respectively. Therefore, the discriminator is trained to output a higher confidence value in real data, and the generator is trained to generate synthetic data that can maximize the confidence score in the discriminator. After a sufficient number of iterations of this training process, both the discriminator and generator will settle to a point, where there is no scope for further improvement (i.e., a Nash equilibrium is achieved).

Since the basic concept of the GAN model was introduced, numerous variants have been proposed to develop the original model by adjusting the objective function or by modifying the model architecture. Among these various models, we focus on the BEGAN model [5], which is based on the concept of autoencoders and reconstruction errors. Unlike other GAN models wherein the objective function is defined based on the distance of distributions between confidence vectors (on real

and synthetic samples), the objective of BEGAN is defined based on the Wasserstein distance between reconstruction error distributions as follows:

$$\begin{cases} \mathcal{L}_D &= \mathcal{L}(x;\theta_D) - k_t \cdot \mathcal{L}(G(z;\theta_G);\theta_D) \\ \mathcal{L}_G &= \mathcal{L}(G(z;\theta_G);\theta_D) \\ k_{t+1} &= k_t + \lambda_k \cdot (\gamma \cdot \mathcal{L}(x;\theta_D) - \mathcal{L}(G(z;\theta_G);\theta_D) \end{cases} \tag{5}$$

where the hyperparameter $\gamma \in [0, 1]$ is the diversity ratio,[1] and λ_k serves as the learning rate for k. Note that $\mathcal{L}(\cdot)$ denotes the reconstruction error of the autoencoder, and t indicates the iteration step.

IV. Proposed Methodology

As shown in Fig. 1, the entire architecture of the proposed AI-based NIDS consists of four main streams: 1) preprocessing; 2) generative model training; 3) autoencoder training; and 4) predictive model training. In this section, we describe the proposed methodology and each module (process) in detail.

A. Preprocessing

Before building and training AI models, the system refines a given raw data set via the preprocessing module that consists of three subprocesses: 1) outlier analysis; 2) one-hot encoding; and 3) feature scaling.

In the outlier analysis phase, the system eliminates outliers, which can negatively affect the model training. Typically, outliers are detected by quantifying the statistical distribution of the data sets via robust measures of scale. There are several standard robust measures of scale for detecting outliers, such as interquartile range (IQR) and median absolute deviation (MAD). Among these measures, we leveraged the MAD.

For a numeric attribute $\mathcal{A} = \{x_1, x_2, \ldots, x_n\}$, the MAD of the attribute is defined as follows:

$$\text{MAD} = \text{median}(|x_i - \text{median}(\mathcal{A})|). \tag{6}$$

We assume that numeric attributes appearing in the data set follow a normal distribution. Then, a consistent estimator $\hat{\sigma}$ for the estimation of the standard deviation is $1.4826 \times \text{MAD}$. In terms of this estimator, we determine that for a given numeric attribute, values exceeding $10 \times \hat{\sigma}$ are outliers. Obviously, outlier analysis is performed only on the numerical attributes and conducted independently for each class. Note that outlier removal should be performed before scaling features, as it can potentially obscure information about outliers.

After filtering out the outliers, the system transforms nominal attributes into one-hot vectors. Each nominal (categorical) attribute is represented as a binary vector with the size of the number of attribute values, where 1 is assigned only to a point corresponding to the expressed value and 0 to all others. For example, in the case of the "protocol" attribute (commonly included in network traffic data) with the values tcp, udp, and icmp, the attribute is transformed into a binary vector of length 3, and the attribute values are converted into [1, 0, 0], [0, 1, 0], and [0, 0, 1], respectively. Together with the one-hot encoding process, the system scales the numeric attributes. In general, normalization (e.g., [28]) and standardization (e.g., [24]) can be considered as scaling for numeric features. Between these two approaches, we adopted the min-max normalization method.[2] The normalization function $f_{\mathcal{A}}(\cdot)$ for a numeric attribute $\mathcal{A}$ that maps $\forall x \in \mathcal{A}$ into a range [0, 1] can be defined as follows:

$$f_A(x_i) = \tilde{x}_i = \frac{x_i - \min x_j}{\max x_j - \min x_j} \tag{7}$$

where x_i denotes the ith attribute value in the attribute A.

In general, existing deep-learning-based approaches consider feature extraction (e.g., PCA, Pearson correlation coefficient, etc.) at this step to feed the model as many informative features as possible, and, consequently, feature extraction can significantly impact the performance of models in anomaly detection. However, we do not consider the computational feature extraction process, as our framework embeds an autoencoder model that can replace functionalities of feature extraction. Note that, in our framework, the model with a computational feature extraction process did not show significant improvement compared with the model without the feature extraction. A detailed description for deploying the autoencoder as a feature extractor is presented later.

B. Synthetic Data Generation With Generative Model

The synthetic data generation module builds and trains generative models using the data set refined in the data preprocessing module. In the case of the generative model, we utilize a state-of-the-art GAN model, BEGAN, which is based on the concept of autoencoders and the reconstruction error-based objective function. For the model architecture, we built the discriminator as a symmetric autoencoder model with five layers and the generator with the same architecture as the decoder of the discriminator (autoencoder). Fig. 4 illustrates the entire architecture of the BEGAN model. Before training the BEGAN model, the system first splits the given data set according to the classes and then builds generative models for each split subdata set. That is, generative models are built in a number equal to the number of classes, and (after training) each generative model produces only synthetic data corresponding to a particular class.

One of the important factors that must be considered when applying GAN models to NIDS is the determination of the termination criteria of training, which has a significant impact on the performance of anomaly detection, as it is directly related to the quality of the synthetic data to be trained on the detection model. The determination of the termination criteria stems from the tracking of the training convergence, and this is a difficult problem, as the objective function of GAN models is defined to have the properties of a zero-sum game. In general, monitoring the training progress has been conducted indirectly through visual inspection of synthetic (generated) data. However, even this approach is not feasible in NIDS environments because the data being handled is not in the form of an image. Fortunately, unlike other

[1] Originally, the diversity ratio γ is defined as $\gamma = \mathbb{E}[\mathcal{L}(G(z))]/\mathbb{E}[\mathcal{L}(x)]$.

[2] In our experiments, there was no significant difference between the two feature scaling methods in terms of the performance of detection models.

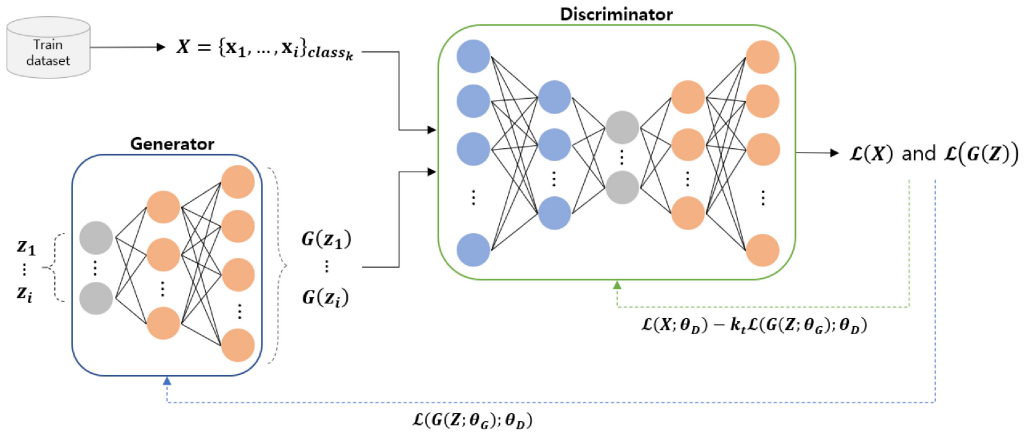

Fig. 4. Architecture of generative model in our system.

GAN models, BEGAN can approximate the convergence of training through the concept of equilibrium, and this characteristic facilitates the determination of the criteria for the training termination. The convergence measure $\mathcal{M}$ of BEGAN is formulated as follows:

$$\mathcal{M} = \mathcal{L}(x) + |\gamma\mathcal{L}(x) - \mathcal{L}(G(z))| \tag{8}$$

where $\mathcal{L}(\cdot)$ is the reconstruction error function, and γ is the diversity ratio.

By utilizing the convergence measure, the system terminates the generative model's training process. That is, when training the generative model, the system considers a threshold as an input parameter and terminates the training process if the convergence measure $\mathcal{M}$ outputs a value less than the given threshold. In the experiment, we set the threshold of the convergence measure $\mathcal{M}$ to 0.058.[3]

After training the generative model, the system generates synthetic data according to the classes using the trained generator and integrates the generated data set into the original training data set. This expanded data set is used to train the autoencoder and detection model in the next stage. Note that although we designed the synthetic data generation module to build multiple generative models according to the number of classes, it can be built as a single model by integrating the concept of the conditional GAN architecture [41], where class attributes are embedded in the input space.

C. Learning the Autoencoder and Detection Model

To build the intrusion detection model, the system first trains an autoencoder model that can provide feature extraction and dimensionality reduction functionalities. In our framework, we designed the autoencoder to possess the same architecture as the discriminator of the generative model. Because the deployed generative model is BEGAN, the discriminator has the form of an autoencoder, as depicted above, and is compatible in terms of the model architecture, as it handles the same data format as the detection model. After building an autoencoder model, the system trains it using the expanded data set composed in the previous module and then utilizes the trained encoder as the feature extraction module. Algorithm 1 presents a detailed process for autoencoder training, where m_i $(1 \leq i \leq k)$ indicates the magnitude of synthetic data to be generated for the class i. Note that the trained encoder is placed at the forefront (input layer) of the detection models as a feature extractor and is set not to learn any more when training detection models (i.e., we fix the model parameters of the trained encoder when training the detection models).

Algorithm 1 Autoencoder Training With Generators

Input: training dataset $\mathcal{D}_{train}$, a set of generators $\mathbf{G}$

1: **Initialize** Autoencoder parameters θ^0_{AE}
2: **for** $G_i \in \mathbf{G}$, where $1 \leq i \leq k$ **do**
3: sample $\mathbf{z} = \{z_j\}_{j=1,\ldots,m_i}$ from the latent space
4: $\hat{\mathcal{D}}_i = G_i(\mathbf{z})$
5: **end for**
6: $\tilde{\mathcal{D}} = \mathcal{D}_{train} \cup \hat{\mathcal{D}}_1 \cup \cdots \cup \hat{\mathcal{D}}_k$
7: $\theta_{AE} =$ **Train_Autoencoder**$(\theta^0_{AE}, \tilde{\mathcal{D}})$
8: $\theta_{enc} =$ **Extract_Encoder**(θ_{AE})

Output: trained encoder θ_{enc}

For detection models, we utilized the basic DNN, CNN, and LSTM as classifiers. We designed the DNN model to possess two hidden layers, and it could naturally process the refined network traffic data in terms of the model training and classification task. In the case of the CNN model, because the model was originally designed to be more suitable for analyzing image data, it required additional transformation processes in the input data space or the layers of the model depending on the approach followed. In our system, we built the CNN model with one-dimensional (1-D) convolutional layers to process the network traffic data, rather than converting the input data (i.e., network traffic data) into a 2-D space. As shown in Fig. 5, we configured the CNN classifier to have two 1-D convolutional layers and one fully connected layer. For LSTM, we designed the model to possess two recurrent layers with the LSTM units and a fully connected layer, as shown in Fig. 6. LSTM is known to be particularly effective

[3]In the learning process, if the convergence measure $\mathcal{M}$ does not fall below a given threshold, the process may fall into an infinite loop. To prevent this, we additionally set the maximum number of iterations.

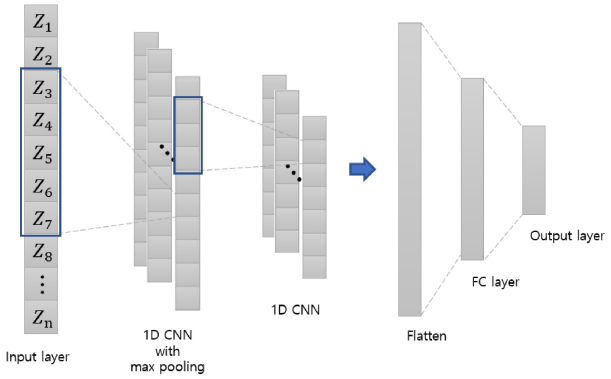

Fig. 5. CNN model architecture in our system.

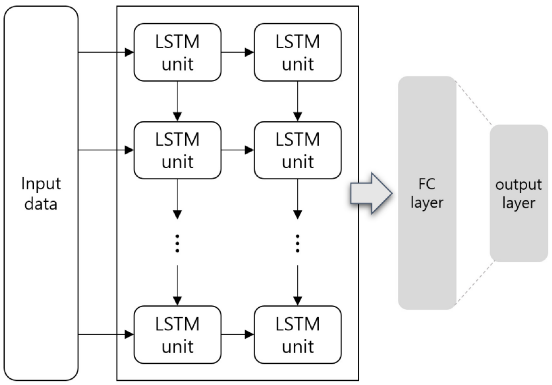

Fig. 6. LSTM model architecture in our system.

in analyzing temporally correlated features [24]. Taking these characteristics into account, we omitted the process of combining with the autoencoder model for the LSTM model, since the encoder may obscure the temporal features. For all models, we designed the output layer with a binary field when the task was to detect anomalies, and with multivalued fields when the purpose was to distinguish not only the anomalies but also the detailed threat types. Algorithm 2 presents a detailed workflow for training a detection model with the trained generators and the trained encoder. As with the autoencoder training process, the magnitude m_i $(1 \leq i \leq k)$ of synthetic data generation can be set differently depending on the weight of each class. Note that the process of combining with the trained encoder (lines 7 and 8 in Algorithm 2) can be omitted according to the predictive model.

From the perspective of the entire framework, the system sequentially processes the data preprocessing, synthetic data generation, and detection model training modules, and we refer to the whole system as G-DNN$_{\text{AE}}$, G-CNN$_{\text{AE}}$, and G-LSTM, according to the type of the detection model. Additionally, we subdivide the whole system into subsystems for a comprehensive comparison. In particular, we consider the DNN, CNN, and LSTM models as naïve deep learning models and DNN$_{\text{AE}}$ and CNN$_{\text{AE}}$, which are models combined with the autoencoder, as advanced deep learning models. In the experiment, we conducted a comparative analysis of G-LSTM, G-DNN$_{\text{AE}}$, and G-DNN$_{\text{AE}}$ with the subsystems.

Algorithm 2 Classifier Training With Generators

Input: training dataset $\mathcal{D}_{train}$, a set of generators $\mathbf{G}$, trained encoder θ_{enc}

1: **Initialize** classifier parameters $\mathcal{W}^0$
2: **for** $G_i \in \mathbf{G}$, where $1 \leq i \leq k$ **do**
3: sample $\mathbf{z} = \{z_j\}_{j=1,\ldots,m_i}$ from the latent space
4: $\hat{\mathcal{D}}_i = G_i(\mathbf{z})$
5: **end for**
6: $\tilde{\mathcal{D}} = \mathcal{D}_{train} \cup \hat{\mathcal{D}}_1 \cup \cdots \cup \hat{\mathcal{D}}_k$
7: **Set** Trainable_State on θ_{enc} = **False**
8: **Build** $\mathcal{W}^0_{\theta_{enc}}$ = **Concatenate_Models**$(\theta_{AE}, \mathcal{W}^0)$
9: $\mathcal{W}_{\theta_{enc}}$ = **Train_Classifier**$(\mathcal{W}^0_{\theta_{enc}}, \tilde{\mathcal{D}})$

Output: trained classifier $\mathcal{W}_{\theta_{enc}}$

V. Experiments and Evaluations

In this section, we first review the target data sets and describe the detailed implementation of each component. Then, we present the experimental results with comparative analysis and evaluate the proposed systems.

A. Data Set Description

In this work, we focused on three network traffic data sets that are widely used as benchmark data sets in the field of intrusion detection systems. Furthermore, we collected the real data from a large enterprise system and analyzed the performance of the proposed model on the real data set.

1) NSL-KDD Data Set: The NSL-KDD data set is a refined version of the KDDcup99 data set [6], [7] and consists of training and testing data sets, KDDTrain and KDDTest, with 125 973 and 22 544 rows, respectively.[4] In each data point, there exist 41 attributes (3 nominal, 6 binary, and 32 numeric attributes) presenting different features of the network flow and a label indicating an attack type or normal behavior. For the attack type, there exist four distinct attack profiles: 1) Denial of Service (DoS); 2) Probing; 3) Remote to Local (R2L); and 4) User to Root (U2R). DoS is an attack that depletes resources by sending excessive traffic to the target system, thereby rendering it incapable of handling legitimate network traffic or service access. In the case of a probing attack, attacker's objective is to gain information about the target system (e.g., scanning ports in use and sweeping IP addresses). R2L is an attack that attempts to obtain local access from a remote machine by sending remote fraudulent traffic to the target, and behaviors, such as password guessing and HTTP tunneling, are considered R2L attacks. In the case of U2R, an attacker first gains access to the target system as an honest user and then attempts to gain root privileges by causing system faults (e.g., buffer overflow and rootkit). Table I presents the entire distribution of the NSL-KDD data set with respect to the classes (attack classes and normal).

2) UNSW-NB15 Data Set: Together with the NSL-KDD data set presented above, the UNSW-NB15 data set [8], which was created by the IXIA PerfectStorm tool, has been widely

[4]The original configuration of the data set includes several subdata sets. However, we only present the main training and testing data sets.

TABLE I
DATA DISTRIBUTION IN NSL-KDD

Class	Training	Weight (%)	Testing	Weight (%)
Normal	67,342	53.46%	9,710	43.07%
DoS	45,927	36.46%	7,460	33.09%
Probing	11,656	9.25%	2,421	10.74%
R2L	995	0.79%	2885	12.79%
U2R	52	0.041%	67	0.29%
Total	125,973	100%	22,543	100%

TABLE II
DATA DISTRIBUTION IN UNSW-NB15

Class	Training	Weight (%)	Testing	Weight (%)
Normal	56,000	31.94%	37,000	44.94%
Generic	40,000	22.81%	18,871	22.92%
Exploits	33,393	19.04%	11,132	13.52%
Fuzzers	18,184	10.37%	6,062	7.36%
DoS	12,264	6.99%	4,089	4.97%
Reconnaissance	10,491	5.98%	3496	4.25%
Analysis	2,000	1.14%	677	0.82%
Backdoors	1,746	0.99%	583	0.71%
Shellcode	1,133	0.65%	378	0.46%
Worms	130	0.07%	44	0.05%
Total	175,341	100%	82,332	100%

used as an experimental data set in the field of anomaly detection systems. Similarly, UNSW-NB15 consists of training and testing data sets, UNSW-NB15_training and UNSW-NB15_testing, with 175 341 and 82 332 records, respectively. Each record possesses 43 attributes that present network flow features and two class attributes.[5] The class attributes consist of an attribute that indicates whether or not the record is normal traffic (binary-valued attribute) and the type of attack (when the record is abnormal). For the attack type, there are nine distinct attack profiles that are intuitively labeled as follows: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. Table II presents the entire distribution of the UNSW-NB15 data set. Note that we excluded any unnecessary attribute that did not affect the training of the models ("id" field) and combined the two class attributes into a single field. Therefore, the data set is considered to have 42 attributes (4 nominal, 2 binary, and 36 numeric attributes) and a class attribute.

3) IoT Data Set: In addition to the data sets NSL-KDD and UNSW-NB15, we evaluated the performance of our system on a network traffic data set, called IoT-23 [9], collected from the IoT devices. The IoT-23 data set consists of 20 subdata sets collected from malicious IoT scenarios and three subdata sets collected from benign scenarios. For these data sets, we utilized the data set collected on the Mirai botnet scenario (named CTU-IoT-Malware-Capture-34-1). The data set contains 23 145 IoT network flows, where each data point belongs to one of the following four classes: 1) Benign; 2) C&C; 3) DDos; and 4) PortScan. Benign matches the normal class, and the others are treated as threats. C&C indicates communication connected to the command and control server, and PortScan refers to the activity of scanning ports to gather information in order to conduct further attacks. For each data point, there are 21 attributes (11 nominal, 2 binary, and 8 numeric attributes) presenting different features of network flow, and we removed four features that did not affect the learning, such as id and IP address. To adjust the magnitude of normal class data considering the data imbalance scenario, we randomly sampled 98 077 data from data sets in the benign scenarios. Consequently, we configured the IoT data set to have 100 000 Benign data, 6706 C&C data, 14 394 DDos data, and 122 PortScan data.

4) Real Data Set: To evaluate the performance of our system in real-world environments, we collected raw security events from a large enterprise system. The data were collected over five months, where threats were logged separately by security operations center (SOC) analysts whenever an intrusion occurred. In the data set, we investigated 798 cyber threats, which occurred evenly over the collection period (not focused on a specific period) and observed 547 system attacks, 240 scanning, and 11 warm attacks (the categorizing was conducted by the SOC analysts). In terms of the categories, the system attack includes cross-site scripting, DDoS, brute force attack, and injection attack, whereas the scanning attack includes Trojan and backdoor attacks. In total, we collected 4 782 342 security event data, of which 230 026 were identified as cyber threats (i.e., 4 552 316 data were labeled as "Normal," and 230 026 data were labeled as "Threat"). Each raw data has 16 basic features for network flow information, such as the protocol type, service, and source bytes (eight nominal and eight numeric attributes). Moreover, because the collected data are raw security events, each data includes information regarding the suspicious security event.[6] Table III presents a distribution of the collected data set with respect to the suspicious security events, and it can be seen that the false

[5]The raw data set contains 47 attributes (excluding class attributes), including source/destination IPs and ports. However, we used the provided training/test data set, in which features that do not affect AI training are excluded.

[6]Note that the suspicious security event can be different from the labels classified by the SOC analysts.

TABLE III
DISTRIBUTION OF RAW SECURITY EVENTS IN THE REAL DATA SET

Event ID	Prefix	Count	Weight (%)
E_2	UDP Packet Flooding	1,048,926	21.9%
E_4	UDP Source-IP Flooding	718,788	15.2%
E_{40}	SIP Vulnerability Scanner	644,683	13.5%
E_7	TCP Connect DoS	553,362	11.6%
⋮	⋮	⋮	
E_{23}	HTTPD Overflow	115,477	2.4%
E_{29}	NTP Amplification DDoS	107,617	2.3%
⋮	⋮	⋮	

positives are relatively high (see [43] for a detailed description of the collected real data set). Note that, although there were several detailed classes of detected attacks, each data was categorized as Normal and Threat only (related to the privacy issues of the enterprise).

B. Implementation and Hyperparameters Tuning

As described in the previous section, we set the discriminator of the generative model to be a symmetric autoencoder model with three layers. For this model, we constructed the first hidden layer with 80 neurons and a latent space dimension with a size of 50. Therefore, the generator is set to have the latent space of size 50 and a hidden layer of size 80. Additionally, we applied batch normalization to each hidden layer for the stability of learning and used the rectified linear unit (ReLU) as the activation function. Note that because we configured the autoencoder as a feature extractor with the same architecture as the discriminator, the above configuration corresponds to that of the autoencoder as well. In the case of the generative model, we set the convergence threshold to 0.058 and terminated training when the convergence measure fell below the given threshold, or the number of epochs reached 250. For autoencoder learning, we set the default number of epochs to 300 and stop training when the reconstruction accuracy was above 0.97.

For the classifier models, we deployed three distinct deep learning models: DNN, CNN, and LSTM. Considering the number of features, we explored the depth of the models up to three layers. In the experiment, the one-layer structure showed high volatility, and the three-layer structure showed a tendency to overfit. As a result, the models were most stable in the two-layer structure and showed the highest performance.

For the DNN model, we set the first hidden layer to have 32 neurons and the second layer to have 16 neurons. For CNN, we used a 1-D-CNN model with two convolutional layers. The convolutional layers are configured to have 32 convolution filters with windows of size 5, and a fully connected layer of 16 neurons follows. Additionally, we applied a max-pooling layer with windows of size 3 to the first convolutional layer, and the batch normalization layer after each convolutional layer. For the activation function, we used ReLU as in the generative model. In the case of LSTM, we connected 64 LSTM cells in each layer and concatenated a fully connected layer with 32 neurons. For these detection models, we set the default number of epochs to 300 and applied the early stop technique (we stopped learning when relative differences of loss are less than 10^{-6} consecutively for 35 epochs [24]).

We utilized two additional basic machine learning models as comparative models.

1) SVM is a supervised learning model based on the statistical learning theory and aims to locate the best hyperplane that can optimally separate input domains according to the classes. In the experiment, we implemented the linear kernel SVM model [2].
2) DT is a nonparametric supervised learning model, and it recursively splits input domains based on the correlation between each feature and class. In this study, we implemented the C4.5 algorithm [1].

For a more extensive comparison, we subdivided the components of our system, DNN, CNN, LSTM, DNN_{AE}, and CNN_{AE}, and utilized them as comparative models with the whole system. Note that we regard these submodels to correspond to the existing AI-based NIDS. In particular, DNN, CNN, and LSTM are considered as naïve deep learning approaches. In the case of DNN_{AE} and CNN_{AE}, they are considered as advanced deep learning approaches combined with autoencoders.[7]

In the experiment, we utilized four metrics to evaluate the performance of AI models: Accuracy, Precision, Recall, and $F1$-score. Accuracy refers to the fraction of correctly inferred results and is commonly used to quantify the performance of AI models. For a given class in a data set, Precision presents the fraction of positive values inferred by the model that is correct, while Recall refers to the fraction of data with positive values that are correctly inferred by the model. The $F1$-score is the harmonic mean of Precision and Recall. The formulas of these metrics are defined as follows:

1) $\text{Accuracy} = \dfrac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}$
2) $\text{Precision} = \dfrac{\text{TP}}{\text{TP} + \text{FP}}$
3) $\text{Recall} = \dfrac{\text{TP}}{\text{TP} + \text{FN}}$
4) $F1\text{-score} = 2 \times \dfrac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$

where TP, TN, FN, and FP denote the true positive, true negative, false negative, and false positive, respectively.

Using these metrics, we evaluated each model on the experimental data sets. Note that, although we built the models with a stable structure, there was still the issue of volatility. Accordingly, with respect to comparison and evaluation, we independently trained each model 100 times and displayed the results for the model with the best detection rate in the test data set.

[7]Although the detailed architecture and configurations may differ from those of the previous approaches, we stress that the implemented models are comparable or outperform in terms of performance to the existing systems.

TABLE IV
BINARY CLASSIFICATION RESULTS FOR THE TEST DATA SET IN NSL-KDD

Classifier	*Accuracy*	Normal			Abnormal		
		Recall	*Precision*	*F*1-score	*Recall*	*Precision*	*F*1-score
SVM	72.1%	97.8%	61.2%	75.2%	53.1%	96.9%	68.6%
DT	81.5%	97.3%	70.8%	81.9%	69.6%	97.1%	81.0%
DNN	79.5%	96.2%	67.7%	79.6%	67.8%	96.2%	79.6%
CNN	80.5%	96.5%	68.7%	80.3%	69.5%	96.6%	80.8%
LSTM	82.0%	97.5%	71.0%	82.1%	70.0%	97.2%	81.3%
DNN_{AE}	85.5%	98.8%	78.0%	87.2%	72.5%	98.5%	83.5%
CNN_{AE}	86.4%	98.8%	79.0%	87.8%	74.1%	98.4%	84.6%
G-LSTM	85.5%	98.5%	78.3%	87.2%	72.5%	98.5%	83.5%
G-DNN_{AE}	89.8%	98.2%	84.3%	90.6%	81.5%	97.9%	89.0%
G-CNN_{AE}	90.3%	97.2%	85.3%	90.9%	83.5%	96.8%	89.7%

C. Experiments on the NSL-KDD Data Set

For the NSL-KDD data set, we explored both binary and multiclassification tasks. Note that NSL-KDD is provided separately as a training data set and a test data set as mentioned above, and we used these data sets in our experiments as provided. In other words, we used KDDTrain (125 973 rows) as a training data set and KDDTest (22 544 rows) as a test data set, and there was no data shuffling between the two data sets. In the experiments on our system (i.e., G-DNN_{AE}, G-CNN_{AE}, and G-LSTM), we generated synthetic data for each class via the generative model and integrated them into the training data set. Obviously, the evaluation of all models was conducted on the original test data set (KDDTest) for unbiased comparisons.

1) Binary Classification: Table IV presents the experimental results for the binary classification task on the NSL-KDD data set. Note that the data belonging to the attack classes are naturally considered anomalies in the binary classification task (labeled as abnormal). In the experiments on our system, we generated a total of 35 000 additional data (synthetic data) for each class via the trained generative module. Fig. 7 shows a comparison of experimental results for the NSL-KDD data set in the binary classification scenario.

Overall, the models output relatively high recall values for the data belonging to the normal class and, conversely, showed relatively high precision values for the abnormal class. For the basic machine learning models, the DT outperformed the SVM model, with an accuracy of 81.5%. Moreover, the DT model performed better than the naïve DNN and CNN models, where DNN achieved an accuracy of 79.5% and CNN achieved an accuracy of 80.5%. Among the basic models and the naïve models, the LSTM model outperformed others with an accuracy of 82.0%. For the advanced deep learning approaches, both DNN_{AE} and CNN_{AE} exhibited better results than the basic machine learning and the naïve deep learning models. The advanced models, DNN_{AE} and CNN_{AE} achieved an 85.5% accuracy and 86.4% accuracy, respectively. The proposed models, to which the generative and autoencoder had been applied, were found to significantly outperform all the aforementioned models. In particular, both G-DNN_{AE} and G-CNN_{AE} achieved an accuracy close to 90%, and it was observed that G-CNN_{AE} produced the highest performance with an accuracy of 90.3%. In the case of LSTM, the generator combined LSTM model performed slightly better than the naïve LSTM, but was measured to be inferior to CNN_{AE}.

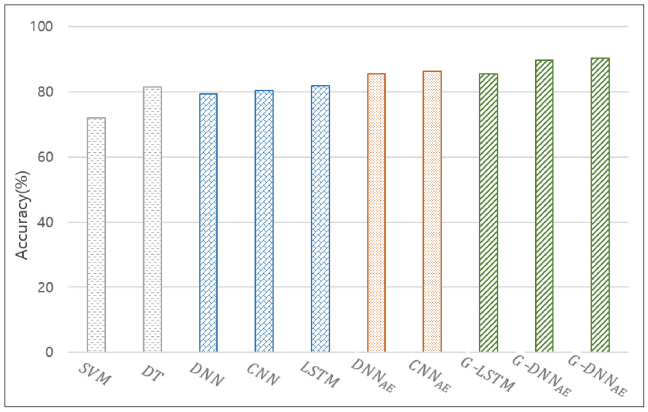

Fig. 7. Comparison of binary classification results on the NSL-KDD data set.

2) Multiclassification: Table V presents the experimental results for the multiclassification task on the NSL-KDD data set.[8] Unlike the binary classification scenario, the system could further recognize the type of threat that the data belonged to and hence, generate synthetic data with different magnitudes based on weights in the population. In the experiments on our system, we generated synthetic data for minor classes with less than 10% weight in the distribution. That is, we generated synthetic data for Probe, R2L, and U2R classes (10 000 synthetic data for each class) via the trained generative model.

[8]In the multiclassification scenario, we only presented experimental results for the attack classes. Experimental results for the normal class follow the previous experiment (i.e., experiments in the binary classification scenario).

TABLE V
MULTICLASSIFICATION RESULTS FOR THE TEST DATA SET IN NSL-KDD

Algorithm	*Accuracy*	DoS			Probe		
		Recall	*Precision*	*F*1-score	*Recall*	*Precision*	*F*1-score
SVM	75.4%	76.3%	96.7%	80.0%	33.3%	80.0%	47.1%
DT	80.5%	84.2%	94.1%	88.9%	50.0%	85.7%	63.2%
DNN	79.6%	83.8%	94.8%	89.0%	31.5%	65.4%	42.5%
CNN	80.1%	89.6%	94.4%	91.9%	30.8%	69.5%	42.7%
LSTM	82.6%	92.1%	98.4%	95.1%	28.7%	69.5%	40.6%
DNN_{AE}	88.3%	94.9%	99.0%	96.9%	93.0%	78.6%	85.2%
CNN_{AE}	88.5%	95.7%	99.0%	97.3%	93.7%	78.4%	85.4%
G-LSTM	87.3%	92.2%	98.4%	95.1%	94.7%	80.2%	84.8%
G-DNN_{AE}	92.7%	96.0%	98.2%	97.1%	95.2%	81.1%	87.6%
G-CNN_{AE}	93.2%	96.0%	97.3%	96.7%	98.2%	80.5%	88.5%

Algorithm	R2L			U2R		
	Recall	*Precision*	*F*1-score	*Recall*	*Precision*	*F*1-score
SVM	-	-	-	-	-	-
DT	11.1%	50.0%	18.2%	-	-	-
DNN	26.0%	48.6%	33.9%	4.6%	74.9%	8.8%
CNN	24.1%	65.1%	35.2%	6.2%	79.9%	11.5%
LSTM	21.8%	63.4%	32.4%	5.9%	76.8%	10.9%
DNN_{AE}	42.7%	93.7%	58.7%	7.8%	83.3%	14.2%
CNN_{AE}	41.1%	93.2%	57.0%	9.3%	85.7%	16.8%
G-LSTM	54.9%	80.4%	65.2%	10.2%	81.7%	18.1%
G-DNN_{AE}	79.8%	80.5%	80.1%	12.4%	79.9%	21.5%
G-CNN_{AE}	92.8%	70.2%	80.0%	11.4%	81.9%	20.1%

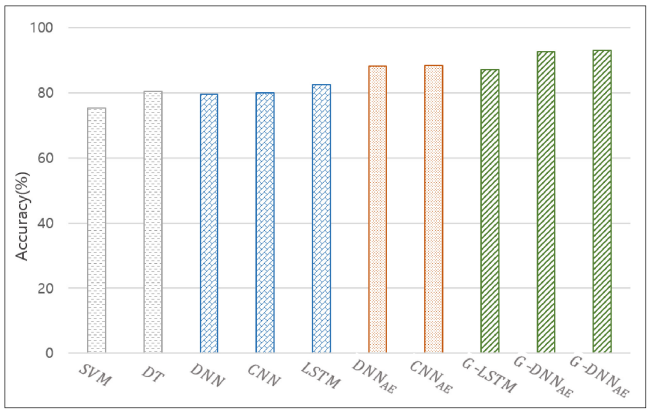

Fig. 8. Comparison of multiclassification results on the NSL-KDD data set.

Fig. 8 shows a comparison of experimental results for the NSL-KDD data set in the multiclassification scenario.

In the case of the basic machine learning models and the naïve deep learning approaches, the results obtained were similar to those obtained for the binary classification scenario. In particular, the DT model showed better results than the SVM, DNN, and CNN models in terms of the accuracy metric. However, the basic machine learning models showed poor results for the minor classes. In particular, they showed extremely low *F*1-scores for the R2L and U2R classes, and even failed to classify. On the contrary, although the detection performance was insufficient, the neural network-based models performed better than the SVM and DT models in the minor classes R2L and U2R. In comparison with the basic models and the naïve models, the LSTM model outperformed others as with the binary classification scenario, and showed better performance in the temporally correlated attack (i.e., DoS attack).

The advanced deep learning models, however, achieved better overall classification performance than the basic machine learning models and the naïve deep learning models, where DNN_{AE} achieved an accuracy of 88.3%, and CNN_{AE} achieved an accuracy of 88.5%. In particular, the models combined with autoencoder demonstrated significant improvement in the Dos, Probe, and R2L classes. However, compared with the naïve deep learning models, they did not improve the classification

TABLE VI
CLASSIFICATION ACCURACY FOR EACH THREAT CLASS ON THE UNSW-NB15 DATA SET

Algorithm	Generic	Exploit	Fuzzers	DoS	Reconnaissance	Analysis	Backdoors	Shellcode	Worms
DNN	76.8%	48.5%	72.9%	28.0%	49.8%	76.3%	87.3%	57.9%	52.2%
CNN	76.7%	48.6%	75.2%	27.9%	49.9%	76.8%	88.3%	58.2%	52.2%
LSTM	76.8%	48.6%	73.2%	29.4%	49.9%	76.6%	87.3%	58.0%	52.2%
DNN_{AE}	76.8%	48.4%	74.1%	28.4%	50.1%	77.1%	88.5%	58.7%	52.2%
CNN_{AE}	76.8%	49.1%	74.3%	28.4%	49.9%	77.5%	88.5%	58.2%	54.5%
G-LSTM	80.1%	49.0%	79.6%	29.4%	50.1%	77.1%	90.8%	58.2%	56.8%
G-DNN_{AE}	80.6%	50.3%	81.6%	29.4%	51.3%	77.2%	91.4%	58.7%	56.8%
G-CNN_{AE}	82.0%	50.2%	81.9%	29.1%	51.3%	77.5%	91.5%	58.9%	56.8%

performance for U2R, which is extremely minor. Note that, although the results seem to have improved numerically, there is not much difference in terms of the number of data. In the case of our models, G-DNN_{AE} and G-CNN_{AE} achieved the best performance compared with that of the other models and achieved an accuracy of 92.7% and 93.2%, respectively. From the perspective of the minor classes, the proposed models comprehensively improved the classification performance and showed a notable improvement in the classification for the R2L class. Note that we did not generate additional synthetic data for the DoS class (a major class) as mentioned above, and it can be observed that the results were measured in a manner similar to the advanced deep learning models.

In summary, we found that neural network-based models combined with autoencoders could significantly improve the classification performance in both the binary and multiclassification tasks, and they can be further improved by applying the generative model. From the perspective of the base model architecture, the DNN-based model and the CNN-based model showed similar classification performance under the same conditions, and no significant differences were found between the two models.

D. Experiments on the UNSW-NB15 Data Set

To compare the performance of models in a data set with more diverse classes, we conducted experiments on the UNSW-NB15 data set as another multiclassification scenario. As described above, UNSW-NB15 has ten classes, including a normal class, three major, and six minor classes. For the minor classes, we determined that classes with a weight of less than 1% are extremely minor. As with the experiments on the NSL-KDD data set, we used the original UNSW-NB15 training and testing data sets (175 341 and 82 332 records, respectively). Similarly, we generated synthetic data for each class via the generative model in the experiments on our system and integrated them into the training data set. Note that the evaluation of all models was conducted on the original UNSW-NB15 testing data set.

Table VI presents the experimental results for the multiclassification scenario on the UNSW-NB15 data set. In the experiments, we generated synthetic data for all classes. In particular, we generated synthetic data to reach a total size of 50 000 for each major class and a total size of 30 000 for each minor class. Additionally, we assumed that for a given threat data, the classification was correct if the model classified the data into one of the classes corresponding to the attack category (even if the model did not predict the exact class). Accordingly, we only indicated the accuracy of the performance measure in the experiment on the UNSW-NB15 data set, considering whether the attack was well classified as an attack. Fig. 9 shows a comparison of experimental results for the UNSW-NB15 data set in the multiclassification scenario.

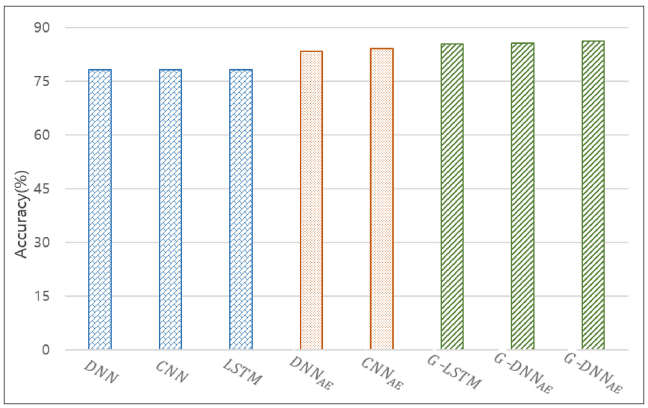

Fig. 9. Comparison of multiclassification results on the UNSW-NB15 data set.

As shown in Table VI, G-DNN_{AE} and G-CNN_{AE} outperformed other models in terms of the classification performance. For the major classes, Generic, Exploit, and Fuzzers, the naïve and advanced deep learning models showed similar performance, and it was observed that the proposed models could improve the classification performance for the major classes even in the LSTM-based model. In particular, the generator combined models showed significant performance improvement in the Generic and Fuzzers classes (up to about 5%). In the case of the minor classes, the proposed models showed a moderate performance improvement overall. Especially, G-LSTM, G-DNN_{AE}, and G-CNN_{AE} achieved about 3% performance improvement in the Backdoors class

TABLE VII
EXPERIMENTAL RESULTS ON THE IOT-23 DATA SET FOR MULTICLASSIFICATION TASKS

Classifier	*Accuracy*	DDoS			C&C			PortScan		
		Recall	*Precision*	*F*1-score	*Recall*	*Precision*	*F*1-score	*Recall*	*Precision*	*F*1-score
DNN	93.1%	100%	100%	100%	47.0%	100%	63.9%	100%	83.7%	91.1%
CNN	93.7%	100%	100%	100%	48.4%	100%	65.2%	100%	86.4%	92.7%
LSTM	93.5%	100%	100%	100%	47.0%	100%	63.9%	100%	85.4%	92.1%
DNN_{AE}	93.7%	100%	100%	100%	48.4%	100%	65.2%	100%	86.4%	92.7%
CNN_{AE}	93.7%	100%	100%	100%	48.4%	100%	65.2%	100%	86.4%	92.7%
G-LSTM, DNN_{AE}, CNN_{AE}	95.9%	100%	100%	100%	80.0%	100%	88.9%	100%	90.4%	95.0%

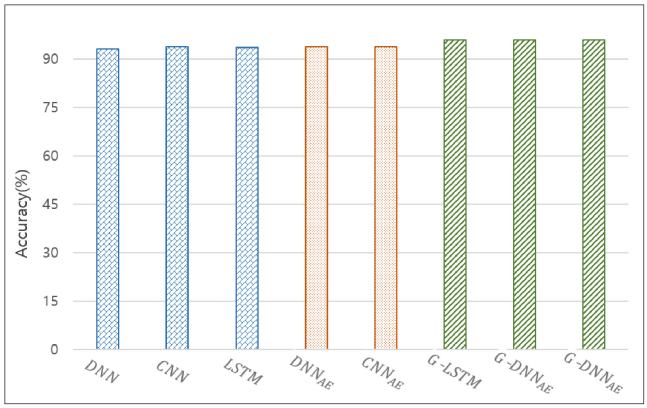

Fig. 10. Comparison of multiclassification results on the IoT-23 data set.

(which possessed weights of approximately 1% within the distribution) compared with the other models. Through experiments on the UNSW-NB15 data set containing more diverse classes, we found that the proposed model could improve the classification performance for major classes. Moreover, we found that the implemented generative model could further improve the classification performance in minor and extremely minor classes.

Although the proposed framework can improve the classification performance, there is still the problem of relatively low detection rates for some classes. In particular, all the experimented models were observed to have relatively low detection rates for the DoS class, even in the LSTM-based model, which is suitable for detecting temporally correlated attacks. Regarding these results, we infer that the domain space between classes is heavily overlapping [34], resulting in low detection rates for some classes.

E. Experiments on the IoT Data Set

To evaluate the performance of the proposed systems in IoT environments, we conducted experiments on the IoT-23 data set. As described above, we utilized the data set collected on the Mirai botnet scenario (CTU-IoT-Malware-Capture-34-1) and intentionally simulated an extreme data imbalance scenario. For evaluation, we randomly split the data set into training and test data sets at a ratio of 7:3 in each class (i.e., 84 855 training data and 36 367 test data). In the experiments on the proposed system, we generated synthetic data to attain a total size of 30 000 for each malicious class in the training data set and evaluated the performance of all models using the previously separated test data (36 367 rows).

Table VII presents the experimental results for the multiclassification task on the IoT-23 data set, and Fig. 10 shows a comparison of experimental results. Overall, all the models achieved an accuracy greater than 93%, and the models were observed to have perfect classification performance for the DDoS class, even in the naïve deep learning approach. Moreover, we observed that there was no significant difference in the performance between the advanced model and the naïve model. In the case of the C&C class, all models showed 100% probability in precision. For the proposed models, all the generator combined models showed the same performance and achieved significant improvement in recall with a probability of 80%. These results are presumably due to the fact that the IoT data set is very simple and has features that contain powerful information related to the nature of the attack (e.g., "history"). In addition, regarding these results, we conjecture that the trained generative model has generated plausible data points that fall within a certain region of the C&C distribution (appearing in the test data set, but not in the training data set), and partially covered the missing region in the (extended) training data set. Moreover, since there is a portion of the data in the corresponding region in the test data set, we estimate that G-LSTM, G-DNN_{AE}, and G-CNN_{AE} performed significantly higher than other models. For the PortScan class, which is extremely minor, all models achieved 100% probability in recall, and the proposed systems achieved the highest precision value with a probability of 90.4%.

F. Experiments on the Collected Real Data Set

To analyze the feasibility of the proposed system in a real environment, we collected real network flow data with raw security events from a large enterprise system and conducted experiments on this real data set. As in the above experiment, we randomly split the collected data set into training and test data sets at a ratio of 7:3 in both normal and abnormal classes (i.e., 3 347 639 training data and 1 434 703 test data). Note that

TABLE VIII
EXPERIMENTAL RESULTS ON THE REAL DATA SET FOR BINARY CLASSIFICATION TASKS

Classifier	*Accuracy*	Normal			Abnormal		
		Recall	*Precision*	*F*1-score	*Recall*	*Precision*	*F*1-score
DNN	94.7%	97.0%	93.0%	94.9%	89.5%	76.9%	82.7%
CNN	95.0%	97.0%	94.5%	95.7%	90.0%	77.5%	83.2%
LSTM	95.2%	97.4%	94.6%	95.9%	89.8%	77.4%	83.1%
DNN_{AE}	95.2%	97.2%	94.7%	95.9%	90.0%	77.5%	83.2%
CNN_{AE}	95.2%	97.3%	94.6%	95.9%	90.2%	77.3%	83.2%
G-LSTM	95.2%	97.3%	94.6%	95.9%	89.8%	77.4%	83.1%
G-DNN_{AE}	95.5%	97.2%	94.5%	95.8%	95.2%	92.5%	93.8%
G-CNN_{AE}	95.6%	97.2%	94.5%	95.8%	95.2%	92.5%	93.8%

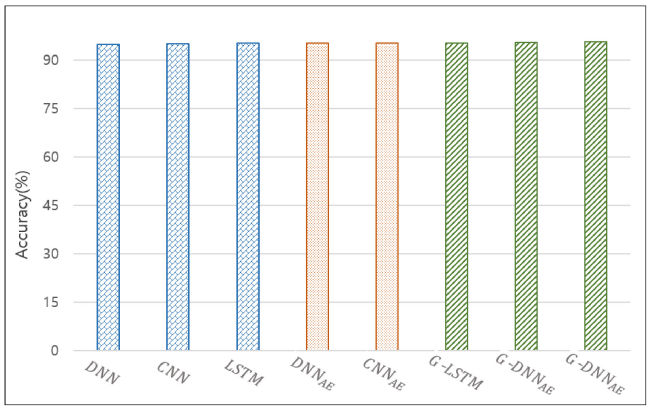

Fig. 11. Comparison of binary classification results on the real data set.

we only considered the binary classification scenario in experiments on the real environment. As shown in Table VIII, the data set possesses a severe imbalance between the normal and abnormal classes. In the experiments on the proposed system, we generated synthetic data for the abnormal class to be the same size as the normal class and evaluated the performance of all models using the previously partitioned test data set as in the previous experiments.

Table VIII presents the experimental results on the real data set, and Fig. 11 shows a comparison of experimental results. First, it can be seen that all models achieve a superior performance in terms of the accuracy, as the data set consists of 95.1% normal data and 4.9% anomalous data. Moreover, there was no significant difference between the naïve and advanced models in terms of the classification performance, as in the experiment on the IoT data set. From the perspective of each class, the models achieved high *F*1-scores for normal data as expected, but relatively low recall values were measured for abnormal data. In the case of the proposed model, G-DNN_{AE} and G-CNN_{AE} achieved 93.8% *F*1-scores in the abnormal class, and we observed that the deployed generative model could significantly improve the classification performance of minor classes even in the real system.

G. Evaluation

Through comprehensive experiments on various data sets, we demonstrated that the proposed system significantly outperform previous deep learning approaches and showed that the classification performance for minor classes can be greatly improved through the generative model. In particular, the proposed models showed a noticeable performance improvement for the R2L and Probe classes on the NSL-KDD data set. In addition, we confirmed that the proposed model can significantly improve the detection rate for most classes on the UNSW-NB15 data set. Moreover, through experiments on the IoT data set, we observed that our system can efficiently detect network threats in a distributed environment. To demonstrate the feasibility in real-world environments, we collected real data and tested our system in the binary classification scenario. Through experiments on the real data set, we demonstrated that the proposed model could improve the detection performance of network anomalies by resolving the data imbalance problem, and that the proposed system can be effectively applied in real-world environments.

VI. CONCLUSION

In this study, we presented a novel AI-based NIDS that can efficiently resolve the data imbalance problem and improve the classification performance of the previous systems. To address the data imbalance problem, we leveraged a state-of-the-art generative model that could generate plausible synthetic data and measure the convergence of training. Moreover, we implemented autoencoder-driven detection models based on DNN and CNN and demonstrated that the proposed models outperforms previous machine learning and deep learning approaches. The proposed system was analyzed on various data sets, including two benchmark data sets, an IoT data set, and a real data set. In particular, the proposed models achieved accuracies of up to 93.2% and 87% on the NSL-KDD data set and the UNSW-NB15 data set, respectively, and showed remarkable performance improvement in the minor classes. In addition, through experiments on an IoT data set, we demonstrated that the proposed system can efficiently detect network

threats in a distributed environment. Moreover, in order to investigate the feasibility in real-world environments, we collected real data from a large enterprise system and evaluated the proposed model on the collected data set. Through this experiment, we demonstrated that the proposed model can significantly improve the detection rate of network threats by resolving the data imbalance problem in the real environment.

In the future, by considering practical distributed environments, we will focus on applying our framework to federated learning systems and ensemble AI systems to enhance network threat detection. In addition, we will study adversarial attacks that can bypass AI-based NIDS through vulnerabilities in AI models and conduct research on enhanced NIDS that can resist these attacks in real-world environments.

REFERENCES

[1] J. R. Quinlan, *C4.5: Programs for Machine Learning* (Morgan Kaufmann Series in Machine Learning). San Mateo, CA, USA: Morgan Kaufmann, 1993.

[2] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[4] I. J. Goodfellow et al., "Generative adversarial nets," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2014, pp. 2672–2680.

[5] D. Berthelot, T. Schumm, and L. Metz, "BEGAN: Boundary equilibrium generative adversarial networks,"2017, *arXiv:1703.10717*.

[6] S. Hettich and S. D. Bay. "KDD cup 1999 data." 1999. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[7] 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.

[8] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set),"in *Proc. Military Commun. Inf. Syst. Conf. (MilCIS)*, 2015, pp. 1–6.

[9] A. Parmisano, S. Garcia, and M. J. Erquiaga, "A labeled dataset with malicious and benign IoT network traffic." 2020. [Online]. Available: https://www.stratosphereips.org/datasets-iot23

[10] B. Ingre and A. Yadav, "Performance analysis of NSL-KDD dataset using ANN," in *Proc. Int. Conf. Signal Process. Commun. Eng. Syst.*, Andhra Pradesh, India, Jan. 2015, pp. 92–96.

[11] Y. Gao, Y. Liu, Y. Jin, J. Chen, and H. Wu, "A novel semi-supervised learning approach for network intrusion detection on cloud-based robotic system," *IEEE Access*, vol. 6, pp. 50927–50938, 2018.

[12] K. Alrawashdeh and C. Purdy, "Toward an online anomaly intrusion detection system based on deep learning," in *Proc. IEEE 15th Int. Conf. Mach. Learn. Appl. (ICMLA)*, Anaheim, CA, USA, 2016, pp. 195–200.

[13] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *Proc. Int. Conf. Wireless Netw. Mobile Commun. (WINCOM)*, 2016, pp. 258–263.

[14] Y. Imamverdiyev and F. Abdullayeva, "Deep learning method for denial of service attack detection based on restricted Boltzmann machine," *Big Data*, vol. 6, no. 2, pp. 159–169, Jun. 2018.

[15] W. Zhong, N. Yu, and C. Ai, "Applying big data based deep learning system to intrusion detection," *Big Data Min. Anal.*, vol. 3, no. 3, pp. 181–195, Sep. 2020.

[16] M. H. Haghighat and J. Li, "Intrusion detection system using votingbased neural network," *Tsinghua Sci. Technol.*, vol. 26, no. 4, pp. 484–495, Aug. 2021.

[17] Y. Yang et al., "ASTREAM: Data-stream-driven scalable anomaly detection with accuracy guarantee in IIoT environment," *IEEE Trans. Netw. Sci. Eng.*, early access, Mar. 8, 2022, doi: 10.1109/TNSE.2022.3157730.

[18] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *ACM Trans. Knowl. Discov. Data*, vol. 6, no. 1, pp. 1–39, Mar. 2012.

[19] X. Zhang et al., "LSHiForest: A generic framework for fast tree isolation based ensemble anomaly analysis," in *Proc. IEEE 33rd Int. Conf. Data Eng. (ICDE)*, Apr. 2017, pp. 983–994.

[20] L. Qi, Y. Yang, X. Zhou, W. Rafique, and J. Ma, "Fast anomaly identification based on multi-aspect data streams for intelligent intrusion detection toward secure industry 4.0," *IEEE Trans. Ind. Informat.*, vol. 18, no.9, pp. 6503–6511, Sep. 2022.

[21] J. Kim, J. Kim, H. L. T. Thu, and H. Kim, "Long short term memory recurrent neural network classifier for intrusion detection," in *Proc. Int. Conf. Platform Technol. Service (PlatCon)*, 2016, pp. 1–5.

[22] 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.

[23] C. Xu, J. Shen, X. Du, and F. Zhang, "An intrusion detection system using a deep neural network with gated recurrent units," *IEEE Access*, vol. 6, pp. 48697–48707, 2018.

[24] J. Gao et al., "Omni SCADA intrusion detection using deep learning algorithms," *IEEE Internet Things J.*, vol. 8, no. 2, pp. 951–961, Jan. 2021.

[25] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," *EAI Endorsed Trans. Security Safety*, vol. 3, no. 9, p. e2, May 2016,

[26] B. Yan and G. Han, "Effective feature extraction via stacked sparse autoencoder to improve intrusion detection system," *IEEE Access*, vol. 6, pp. 41238–41248, 2018.

[27] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.

[28] C. Ieracitano, A. Adeel, F. C. Morabito, and A. Hussain, "A novel statistical analysis and autoencoder driven intelligent intrusion detection approach," *Neurocomputing*, vol. 387, pp. 51–62, Apr. 2020.

[29] J. Y. Kim, S. J. Bu, and S. B. Cho, "Malware detection using deep transferred generative adversarial networks," in *Proc. Int. Conf. Neural Inf. Process.*, 2017, pp. 556–564.

[30] M. H. Shahriar, N. I. Haque, M. A. Rahman, and M. Alonso, "G-IDS: Generative adversarial networks assisted intrusion detection system," in *Proc. IEEE 44th Annu. Comput., Softw., Appl. Conf. (COMPSAC)*, Jul. 2020, pp. 376–385.

[31] I. Yilmaz, R. Masum, and A. Siraj, "Addressing imbalanced data problem with generative adversarial network for intrusion detection," in *Proc. IEEE 21st Int. Conf. Inf. Reuse Integr. Data Sci. (IRI)*, Las Vegas, NV, USA, 2020, pp. 25–30.

[32] D. Li, D. Kotani, and Y. Okabe, "Improving attack detection performance in NIDS using GAN," in *Proc. IEEE 44th Annu. Comput., Softw., Appl. Conf. (COMPSAC)*, Jul. 2020, pp. 817–825.

[33] W. Lee, B. Noh, Y. Kim, and K. Jeong, "Generation of network traffic using WGAN-GP and a DFT filter for resolving data imbalance," in *Proc. Int. Conf. Internet Distrib. Comput. Syst. (IDCS)*, Oct. 2019, pp. 306–317.

[34] G. Dlamini and M. Fahim, "DGM: A data generative model to improve minority class presence in anomaly detection domain," *Neural Comput. Appl.*, vol. 33, pp. 13635–13646, Apr. 2021.

[35] D. Li, D. Chen, J. Goh, and S.-K. Ng, "Anomaly detection with generative adversarial networks for multivariate time series," 2018, *arXiv:1809.04758*.

[36] S. K. Alabugin and A. N. Sokolov, "Applying of generative adversarial networks for anomaly detection in industrial control systems," in *Proc. Global Smart Ind. Conf. (GloSIC)*, Nov. 2020, pp. 199–203.

[37] I. Siniosoglou, P. Radoglou-Grammatikis, G. Efstathopoulos, P. Fouliras, and P. Sarigiannidis, "A unified deep learning anomaly detection and classification approach for smart grid environments," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 2, pp. 1137–1151, Jun. 2021.

[38] D. E. Rumelhart and J. L. McClelland, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*, vol. 1. Cambridge, MA, USA: MIT Press, 1987, pp. 318–362.

[39] G. E. Hinton and R. S. Zemel, "Autoencoders, minimum description length and helmholtz free energy," in *Proc. 6th Int. Conf. Neural Inf. Process. Syst.*, 1993, pp. 3–10.

[40] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2016, *arXiv:1511.06434*.

[41] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014, *arXiv:1411.1784*.

[42] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 214–223.

[43] J. Lee, J. Kim, I. Kim, and K. Han, "Cyber threat detection based on artificial neural networks using event profiles," *IEEE Access*, vol. 7, pp. 165607–165626, 2019.

Cheolhee Park received the B.S. degree from the Department of Applied Mathematics, Kongju National University, Gongju, South Korea, in 2014, and the M.S. and Ph.D. degrees from the Department of Mathematics, Kongju National University in 2017 and 2021, respectively.

He joined Electronics and Telecommunications Research Institute, Daejeon, South Korea, in 2021, where he is currently working as a Researcher. His research interests include data privacy, differential privacy, machine learning, deep learning, AI security, and network security.

Jonghoon Lee received the B.S., M.S., and Ph.D. degrees in computer engineering from Kyungpook National University, Daegu, South Korea, in 2000, 2002, and 2020, respectively.

He joined Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea, in 2002. Since 2002, he has been involving in various research projects for cyber security and network fields. He is currently a Principle Researcher with the Cyber Security Research Division, ETRI. His research interests include cyber security, 5G network security, AI-based network intrusion detection, AI-SIEM techniques for 5G, and network big data analytics for cyber security.

Youngsoo Kim received the B.S. degree from the Department of Information Engineering, Sungkyunkwan University, Seoul, Republic of Korea, in 1998, and the M.S. and Ph.D. degrees from the Department of Computer Engineering, Sungkyunkwan University in 2000 and 2009, respectively.

He joined Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea, in 2000, where he is currently working as a Principal Researcher. From 2012 to 2015, he was an Adjunct Professor with Chungnam National University, Daejeon. He is currently interested in 5G security, network security, digital forensics, cryptography, and AI security.

Jong-Geun Park received the B.S. and M.S. degrees from the Department of Industrial Engineering, Sungkyunkwan University, Seoul, Republic of Korea, in 1997 and 1999, respectively, and the Ph.D. degree from the Department of Computer Engineering, Chungnam National University, Daejeon, Republic of Korea, in 2013.

From 1999 to 2001, he was a Researcher with ADD, Daejeon. Then, he joined Electronics and Telecommunications Research Institute, Daejeon, in 2001, where he is currently working as a Principal Researcher. He is currently interested in mobile network security, SDN/NFV, cloud security, and AI security.

Hyunjin Kim received the B.S. degree in information communications engineering and the M.S. and Ph.D. degrees in computer science and engineering from Chungnam National University, Daejeon, South Korea, in 2015, 2017, and 2021, respectively.

He is currently a Researcher with Electronics and Telecommunications Research Institute, Daejeon. He is interested in information security, both theoretical and practical, and his recent research is largely about network security and applied cryptography.

Dowon Hong received the B.S., M.S., and Ph.D. degrees in mathematics from Korea University, Seoul, South Korea, in 1994, 1996, and 2000, respectively.

He has been a Principal Member of Engineering Staff of Electronics and Telecommunications Research Institute, Daejeon, South Korea, from 2000 to 2012. He joined the Department of Applied Mathematics, Kongju National University, Gongju, South Korea, in 2012, where he has been a Full Professor since 2015. His research interests include cryptography, data privacy, differential privacy, and network security.