Journal of Network and Systems Management (2023) 31:5
https://doi.org/10.1007/s10922-022-09698-w

Network Intrusion Detection System Based on an Adversarial Auto-Encoder with Few Labeled Training Samples

Kohei Shiomoto[1]

Received: 9 March 2022 / Revised: 22 August 2022 / Accepted: 26 September 2022 /
Published online: 7 October 2022

Abstract

Network intrusion detection systems (NIDS) are critical to defending network systems from cyber attacks. Recently, machine learning has been applied to enhance NIDS capability. To train a supervised machine-learning model, a large number of labeled training samples are required to achieve practical performance. However, labeling data samples is a costly task. Additionally, obtaining anomaly data samples is difficult because trends in network traffic that are subject to NIDS change daily, and new attacks continue to be generated. To address this issue, we propose a semi-supervised machine-learning-based NIDS that reduces the required number of labeled training samples by applying an adversarial auto-encoder (AAE) technique. We evaluated the proposed method through a series of experiments and confirmed that the proposed AAE-based NIDS achieves performance comparable to that of multi-layer perceptron-based NIDS with only 0.1% of the labeled training samples. We also confirmed that the selection of data samples for annotation does not affect the performance of the proposed AAE-based NIDS. We also evaluated the relationship between the performance of the proposed method and the dimension of its latent-variable vector. The best performance as measured by recall and F1 score occurred when the dimensionality of the latent variable vector was 10, which suggests that this structure allows for accurate decomposition of attack and normal. This study presents promising results obtained by the proposed semi-supervised learning method with a reduced number of labeled training samples, which reduces the operational costs of a machine-learning-based NIDS.

Keywords Adversarial auto-encoder · Network intrusion detection system · Semi-supervised learning

✉ Kohei Shiomoto
shiomoto@tcu.ac.jp

[1] Department of Intelligent Systems, Faculty of Information Technology, Tokyo City University, 1-28-1 Tamazutsumi, Setagaya, Tokyo 158-8557, Japan

1 Introduction

A network intrusion detection system (NIDS) monitors activities in a network and classifies them as "benign" or "malicious" [1, 2]. Recently, machine learning has been applied to enhance the capabilities of anomaly-detecting NIDS [3]. A problem that hinders widespread application is the required number of labeled training samples to achieve practical performance. The more training data used, the better the performance. However, labeling data samples is a costly task that requires a human operator to examine each data item, classify it, and label it. Additionally, trends in network traffic that are subject to NIDS auditing change daily, and new attacks continue to generate. Hence, labeling work must be done constantly, creating numerous problems.

To address this issue, we propose a semi-supervised machine-learning-based NIDS that reduces the required number of labeled training samples. We use a smaller set that would ordinarily result in poor performance for supervised learning classification. To avoid this, our semi-supervised learning method exploits unlabeled training samples, which does not require costly human labor. We use an adversarial auto-encoder (AAE) to realize semi-supervised learning in this fashion [5] alongside a generative adversarial network (GAN) [4]. These two components comprise the key building blocks of our method [5]. The auto-encoder (AE) reduces the dimensionality of input data by extracting and maintaining important features as a latent variable vector, whereas the GAN employs a generator and a discriminator such that the latent-variable vector of the AE follows an arbitrary distribution for regularization. In our proposed method, we divide the latent-variable vector into two subset vectors: one for classification and the other for traffic feature representation. Using unlabeled data samples only, the AE is trained to extract two latent-variable vectors, and the GAN is trained to allow them to follow categorical and Gaussian distributions, respectively. Then, using labeled data samples, the AE is trained to minimize the cross-entropy error. Finally, the latent-variable vector for classification is used to classify the input data as normal or as an attack.

In our earlier work [6], we reported preliminary evaluation results. In this study, we investigate the performance of the proposed method through a series of detailed experiments to answer the following questions:

- How many unlabeled training samples are required to obtain practical performance?
- Does the selection of labeled training samples affect performance?
- How many dimensions does the latent-variable vector require to extract the traffic features needed to build the NIDS?

We evaluate the proposed method through a series of experiments and confirm that the proposed AAE-based NIDS achieves performance comparable to that of multi-layer perceptron (MLP)-based NIDS with only 0.1% of the labeled training samples. We confirm that when the number of labeled data samples is small,

the accuracy does not diminish. Moreover, the selection of data samples for annotation does not affect the performance of our proposed AAE-based NIDS. We also confirm that the best performance as measured by recall and F1 score occurs when the dimensionality of the AE's latent variable vector is 10, which suggests that this structure can decompose the attack and normalize communications expediently. Hence, we show that the proposed AAE-based NIDS effectively uses a small number of labeled training data samples to reduce the need for costly human labor while improving performance with the support of unlabeled data. This study demonstrates promising results that will be useful to industry and academia.

The rest of this paper is organized as follows. Section 2 describes related work. Section 3 discusses the dataset used for training the machine-learning-based NIDS. Section 4 presents the proposed method. Section 5 evaluates the proposed method via experiments. Section 6 concludes the paper.

2 Related Work

Various machine-learning algorithms have been applied to anomaly-type NIDS [3, 7, 8]. In the early stages of research on the application of machine learning to NIDS approximately 20 years ago, attention was focused on shallow learning methods [9] such as K-nearest-neighbor, naive Bayes, the C4.5 decision tree algorithms [10], Principal Component Analysis [11], and support vector machines (SVM) [12, 13]. Feature engineering is important in shallow machine learning, and methods have been proposed to extract the 10 top-ranked features from the Network Security Laboratory Knowledge Discovery in Databases (NSL-KDD) dataset by making use of the information gain [14, 15]. Multivariate correlation analyses with the support of a dissimilarity measure were conducted to improve accuracy [16].

It is generally believed that there is no single model that will solve different problems simultaneously. Indeed, even if multiple models were highly effective for a given problem, finding the best model for different data distributions or statistical mixtures would be difficult. Ensemble learning is the practice of combining multiple models to improve predictive performance. Several studies have been conducted on the application of ensemble learning to NIDS [17–21].

Recent advances in deep learning have stimulated research on how to realize NIDS without having to conduct feature engineering. Spectral clustering and a deep neural network have been applied for a NIDS for sensor networks [22]. A recurrent neural network (RNN)-based NIDS [23–25] including a long short-term memory (LSTM)-based one [26] were proposed. A convolutional neural network (CNN) was applied [27]. A CNN and LSTM were used for feature extraction to capture spatio-temporal features [28].

A GAN was used to extract statistical features [29].

The attention mechanism was used to perform feature learning by highlighting the key input of sequential network flow data composed of packet vectors in a bidirectional LSTM model [30].

Auto-encoders (AEs) are considered suitable for anomaly-type NIDS because they can determine whether a deviation from a previously learned normal state is detected based on whether the reconstruction error is within a threshold value [31].

A robust AE was proposed [32] to address issues of denoising [33, 34], requiring noise-free data. A maximum correntropy AE was proposed [35] to provide representations influenced by the outliers and noise. AE reconstruction errors were used to augment and train the classifier [15]. A stacked AE was proposed to extract features from log data of n-gram hypertext transmission protocols [36].

Several authors proposed NIDS comprising AEs and/or deep belief networks to extract feature representations followed by classifiers. NIDSes have been proposed based on an asymmetric AE [37], a stacked AE [38], and a sparse AE [39–41]. Stochastic denoising AE [42] was applied to different NIDSes [43–45], and a stacked contractive AE [46] was applied to yet another NIDS [47]. A deep belief network was employed [48], using a classifier during the second stage based on various shallow-learning algorithms (e.g., random forest [37], SVMs [39, 47, 48], and soft-max classifiers [38, 40, 41, 40–41]). Self-taught [49] learning-based NIDSes were proposed [39–41, 50].

Although there have been several studies in which machine learning algorithms were applied, few have investigated the problems caused by having only a small number of labeled training samples.

Overfitting occurs when a model is trained on a small sample. One-shot learning and few-shot learning, which are inspired by the human ability to learn things from a few examples, have been proven to train a model with a small number of data instances and achieve high performance for image recognition [51, 52]. Recently, one-shot learning and few-shot learning have been applied to network security in the area of malware detection, where malware is converted to image data [53–55].

More recently, one-shot learning with Siamese networks was applied to a NIDS [56] to learn pair similarities rather than features that are unique to each class, although open questions remain, including that careful consideration should be given to ensure the same number of training pairs for all class combinations when creating the training set.

Semi-supervised learning is another approach to solving the problem of overfitting due to a small quantity of labeled training data.

Semi-supervised learning models based on the variational auto-encoder (VAE) have been proposed [57], and in our previous work [6], Adversarial Auto-Encoder (AAE) for semi-supervised learning NIDS was proposed and preliminary evaluated. While the VAE assumes that the underlying distribution of the latent variable is Gaussian, our AAE-based method is more general and assumes that any distribution can be used as the underlying distribution of the latent variable.

A hybrid method using an LSTM auto-encoder and a one-class SVM that uses only normal class examples in the training dataset has also been proposed [58].

A stacked sparse autoencoder (SSAE)-based semi-supervised deep learning models [59] and their extension to federated learning (FL) [60], which combines unsupervised feature extraction and supervised classification algorithms to make use of information from both unlabeled and labeled data, have been proposed in recent years.

Those approaches are in the early stages of research. Even if a model is very effective for a given problem, it is very difficult to find the best model for different data distributions and statistical mixtures. In order to find the appropriate approach for the problem, the characteristics of these approaches should be investigated in detail. In this paper, we present the performance of the method proposed in our earlier work [6] through a series of detailed experiments.

3 Training Dataset

3.1 Limited Quantity of Labeled Data

A typical machine learning workflow comprises a pre-training phase in which clustering is performed using unsupervised learning, a data annotation phase in which human operators manually examine data samples and attach labels, and a supervised learning phase in which the classifier is trained using the set of labeled data. For the classification task, existing supervised-learning algorithms require a dataset of high quality containing a sufficient quantity of human-annotated data samples for training. Because annotation is an extremely costly and time-consuming task, a new method is needed to enable more efficient classifier training. Furthermore, obtaining anomaly data samples is difficult because trends in network traffic that are subject to NIDS oversight change daily, and new attacks continue to be generated. This quickly leads to unbalanced datasets that must be updated continuously .

3.2 The NSL-KDD Dataset

To investigate the effect of the number of labeled data samples, we use the NSL-KDD dataset [9] as a benchmark. This dataset has been used extensively to evaluate machine-learning-based NIDS methods. NSL-KDD is an enhanced version of the KDD CUP 99 [61, 62] dataset, used in the Third International Knowledge Discovery and Data Mining Tools Competition, which was held in conjunction with the Fifth International Conference on Knowledge Discovery and Data Mining (KDD-99). The competition task was to build a predictive network intrusion detector capable of distinguishing between intrusions and normal connections. The database contains a standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment. A major criticism of the KDD CUP 99 dataset [61] pertains to its large degree of redundancy. Therefore, the authors of the NSL-KDD paper [9] removed duplicates and created more sophisticated subsets. This dataset is divided into KDDTrain+ (125,973 data records) for training and KDDTest+ (22,544 data records) for testing.

This dataset consists of records of traffic sent and received between source and destination internet protocol (IP) addresses. It was created from transport control protocol (TCP) dump data from 7 weeks of network traffic processed into about five million connection records. Similarly, two weeks of testing data yielded approximately two million connection records. Each traffic sample has 41 features

categorized into three types: basic, content-based, and traffic-based. Among these, some are categorical protocol types that have three possible values (i.e., tcp, udp, and icmp), a flag that has 11 possible values (SF, S1, REJ, etc.), and a service that has 70 possible values (http, telnet, ftp, etc.). Instead of coding each categorical data into scalar values, we adopt a one-hot vector representation, resulting in 122 features.

All data are labeled as "normal" or "anomaly". Attacks in the dataset are classified into four categories according to their characteristics: denial-of-service (DOS), remote-to-local (R2L) (i.e., unauthorized access from a remote machine), user-to-remote (U2R) (i.e., unauthorized access to local superuser (root) privileges), and probes (e.g., surveillance and port scanning).

The details of each category are described in Table 1. The KDDTrain+ dataset contains 22 types of data, and the KDDTest+ dataset contains 38 types. Some specific attack types (boldface) in the testing dataset do not appear in the training dataset. KDDTest+ contains 17 types of attack data that are not in KDDTrain+, and KDDTrain+ contains two types of attack data that are not in KDDTest+. This renders the detection scenario more realistic. Therefore, the KDDTest+ dataset is a reliable indicator of performance with respect to attacks that have not been seen previously, such as zero-day attacks and variants of existing attack types.

4 Semi-supervised Learning

4.1 AAE

Supervised learning requires a large number of data instances to achieve practical performance. The more training data used, the better the classifier performance. Unfortunately, obtaining a large quantity of training data is costly. Unlike supervised models, semi-supervised learning requires just a small set of labeled data for training.

Table 1 Attack types in the NSL-KDD dataset [9] Attack types written in **bold** in the testing dataset do not appear in the training dataset.

Category	Training Dataset	Testing Dataset
DoS	back, land, neptune, pod, smurf, teardrop	back, land, neptune, pod, smurf, teardrop, **mailbomb**, **processtable**, **udpstorm**, **apache2**, **worm**
R2L	fpt-write, guess-passwd, imap, multihop, phf, spy, warezclient, warezmaster	fpt-write, guess-passwd, imap, multihop, phf, spy, warezmaster, **xlock**, **xsnoop**, **snmpguess**, **snmpgetattack**, **httptunnel**, **sendmail**, **named**
U2R	buffer-overflow, loadmodule, perl, rootkit	buffer-overflow, loadmodule, perl, rootkit, **sqlattack**, **xterm**, **ps**
Probe	ipsweep, nmap, portsweep, satan	ipsweep, nmap, portsweep, satan, **mscan**, **saint**

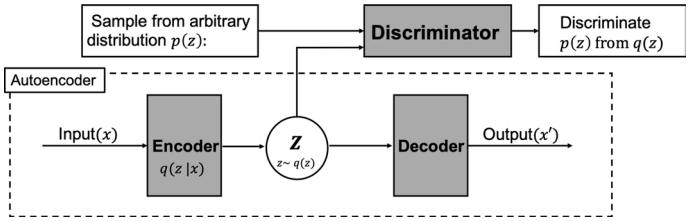

Fig. 1 AAE Architecture

Therefore, to improve the performance of NIDS, we propose to use semi-supervised learning to take advantage of unlabeled training data and reduce the need for human intervention.

We apply AAE to implement the semi-supervised learning algorithm. Figure 1 presents the architecture of the AAE, which comprises an AE and a GAN as its key building blocks. The AE reduces the dimensionality of input data by extracting and maintaining important features as a latent variable vector, z, whereas the GAN employs the generator and discriminator so that z follows an arbitrary distribution for regularization. Hence, an AAE can be viewed as an AE that forces hidden variables to follow any desired distribution.

In an AAE, the latent variable vector of the AE is regularized by the discriminator of the GAN in order to match an arbitrary prior, $p(z)$, to the aggregated posterior, $q(z)$, of the latent variable vector, z. Let x be the input and z be the latent variable vector of the AE. Let $q(z \mid x)$ be an encoding distribution and $p(x \mid z)$ be a decoding distribution. Further, let $p_d(x)$ be the data distribution and $p(z)$ be the prior distribution we want to force the latent variable vector to follow. The encoding function of the AE, $q(z \mid x)$, defines a posterior distribution of $q(z)$ on the latent variable vector of the AE as follows:

$$q(z) = \int_x q(z \mid x) p_d(x) dx \tag{1}$$

The adversarial network and the AE are trained jointly using the stochastic gradient descent (SGD) method. As the learning progresses, the AAE can match the arbitrary prior, $p(z)$, to the aggregated posterior, $q(z)$. Then, z follows the prior distribution. Thus, the encoder of the AE is regarded as the GAN generator. As such, z is regarded as the generated data. The discriminator discriminates the z generated by the AE that follows the distribution of $q(z)$ from the sample generated by the generator that follows the distribution of $p(z)$. It yields the probability that input data come from the sample of the arbitrary prior, $p(z)$.

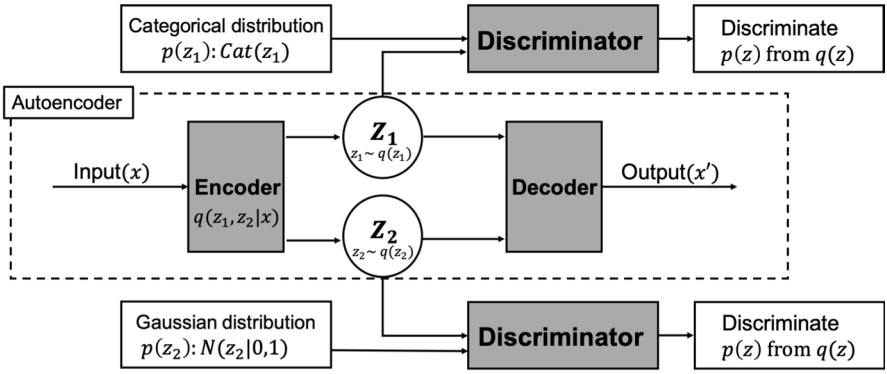

Fig. 2 Architecture of an AAE using semi-supervised learning

4.2 Proposed Method

4.2.1 Architecture

Figure 2 shows the architecture of the proposed method. We assume that there should be complex latent expressions behind normal and anomaly communications. The AE extracts two latent variable vectors, z_1 and z_2, to hold the features of the input data. The encoder, $q(z_1, z_2 \mid x)$, generates z_1 and z_2, where z_1 holds the features representing the class information (i.e., "normal" or "attack")and z_2 holds the other features. The z_1 corresponding to the categorical distribution is designed to record the label associated with the input data. We use z_2 to impose the Gaussian distribution in order to preserve detailed features other than class information. The z_2 corresponding to the Gaussian distribution is designed to separate clusters.

The proposed AAE employs two pairs of generative models and discriminators to regularize the latent variables z_1 and z_2. One follows a categorical distribution and the other a Gaussian distribution. A categorical distribution, $cat(z_1)$, is used to force z_1 to represent only the class datum, whereas a Gaussian distribution, $N(z_2 \mid 0, I)$, is used to force z_2 to represent other information. The categorical distribution takes the same number of one-hot values as the number of classes. In our method there are two classes, "normal" or "attack". The latent variable z_1 is learned by the discriminator to hold a one-hot vector. As such, classification can be performed by referring to the value of z_1 estimated by the encoder.

4.2.2 Workflow of the Proposed Method

We train the AAE using unlabeled data. A latent variable z_1 corresponding to the categorical distribution is designed to record the label associated with the input data. A latent variable z_2 corresponding to the Gaussian distribution is designed to separate clusters. Therefore, it is assumed that input data are generated by a latent class variable z_1 that comes from a categorical distribution and a latent variable z_2 that

comes from a Gaussian distribution. When labeled data are available, we train the AAE by using the label instead of the categorical generative model. When the AAE is trained, it is used to classify new incoming data. The latent variable z_1 in the middle hidden layer indicates the inferred class associated with the input data. As such, at the time of detection, classification is performed by a latent variable z_1 that represents the class information, indicating whether the input data are normal or anomaly.

The middle of Fig. 2 is the neural network of the AE that reduces the dimensionality of the input data (*Input*(X)) and generates latent-variable vectors (z_1, z_2). The top of Fig. 2 is the neural network of the discriminator that imposes a categorical distribution on a latent class variable vector z_1, as a prior distribution. Therefore, the distribution of z_1 is guaranteed to match the categorical distribution. Hence, only class information that is required to detect an attack is extracted from the latent-variable vectors. The bottom of Fig. 2 is the neural network of the discriminator that imposes a Gaussian distribution on a latent class variable z_2, as a prior distribution. Therefore, the distribution of z_2 is guaranteed to match a Gaussian distribution. The semi-supervised AAE is trained with the SGD in three phases as follows:

1. Reconstruction phase: the encoder and decoder are updated. Optimizes the parameters of the AE to minimize reconstruction error for input and output data. Only unlabeled data are used in this phase. The AE generates the latent-variable vectors z_1 and z_2 from unlabeled data in this phase.
2. Regularization phase: each discriminator is trained to discriminate the latent-variable vector, z_1, or sample the categorical distribution, and to discriminate the latent variable vector, z_2, or sample the Gaussian distribution. The AE is optimized based on the determination result of the discriminator. This training is based on Eq. (1).
3. Semi-supervised classification phase: the AE is updated to minimize the cross-entropy error on labeled data. In this phase, we conduct semi-supervised learning using labeled data.

After the AAE is trained, the latent variable z_1 is used to classify whether the input data are normal.

5 Evaluation

5.1 Neural Network Parameters

We implemented the proposed AAE method using Pytorch [63]. To compare the proposed method to an existing one, we also implemented an MLP-based deep neural network (DNN). We use an ADAM optimizer [64] for both models. We also added dropout and batch normalization to prevent overfitting during the training phase. The architecture of the MLP model and that of our AAE model are shown in Fig. 3. In this example, regarding the AAE, the encoder has 122 inputs that are compressed into the important features, yielding 52 (= 2 + 50) outputs (two for z_1 and

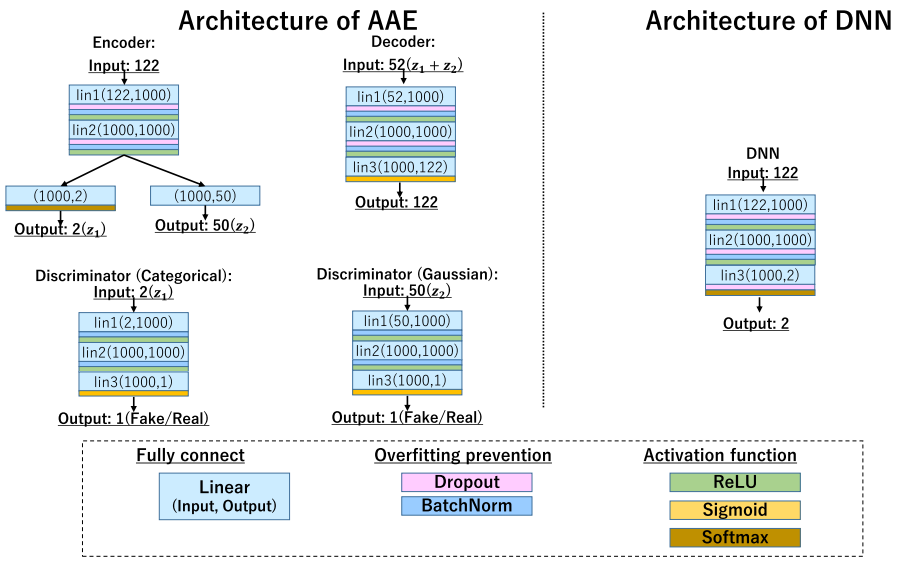

Fig. 3 Architecture of the AAE and DNN

50 for z_2). The decoder receives 52 inputs from the hidden middle layer that has the latent variable vectors (z_1 and z_2) and yields 122 outputs. Both the encoder and the decoder have a middle fully-connected layer with a size of 1000×1000. They also have a drop-out layer with batch normalization between layers. The discriminator for the categorical distribution (z_1) receives two inputs and yields one output ("Fake" or "Real"). The discriminator for the Gaussian distribution (z_2) receives 50 inputs and yields one output ("Fake" or "Real"). Both discriminators for the categorical distribution (z_1), and the Gaussian distribution (z_2) have a middle fully-connected layer with a size of 1000×1000. They also have batch normalization between layers.

5.2 Dataset

To investigate the impact of the *small* number of labeled data samples, we created 10 training datasets by randomly selecting data samples from KDDTrain+ (125,973 data items) as labeled data. p_{sample} is a fraction of labeled samples randomly selected over the total from KDDTrain+. We used KDDTest+ (22,544 data items) for testing.

When p_{sample} is small, the way the selected data samples are distributed will strongly affect the performance of the machine learning-based NIDS. Here, if we were to use supervised learning instead of our proposed semi-supervised learning, we would need to select the label data carefully. If by chance we can select "good" or "representative" data samples for training, supervised learning may succeed in constructing a valid model. Conversely, if a "bad" data sample is selected, a valid model cannot be obtained by supervised learning. Therefore, given the constraint that only a small number of labeled data samples can be chosen, supervised learning

requires careful selection of the samples to be labeled. In contrast, our proposed semi-supervised learning is expected to require less careful selection of samples to label.

5.3 Effectiveness of the Proposed Method

To reduce the required number of labeled data samples in the training dataset, we propose a semi-supervised machine-learning-based NIDS that applies an AAE technique. The first question is "*How many labeled data samples are required in the training dataset?*" The AAE-based NIDS uses the structure of the unlabeled data-sample distribution to improve the accuracy of the boundaries between adjacent classes calculated from the labeled samples. We examine how the AAE-based NIDS improves its performance by taking advantage of the unlabeled samples, avoiding the need for costly human labor.

Figure 4 shows the performance of the AAE-based NIDS, including accuracy, precision, recall, and F1 score,

when using 0.1% of the training data of the NSL-KDD dataset as labeled (i.e., p_{sample}= 0.001). In Fig. 4, the horizontal axis represents the number of unlabeled training samples. In Fig. 4, the performance of the MLP is shown for comparison (see the four rightmost bars). Amongst the performance measures, false-negative and false-positive ratios are important because the false detection risk is very high with NIDS [65]. Focusing on the recall in Fig. 4, we observe that the AAE-based NIDS achieves higher recall than the MLP-based NIDS. We confirm that the proposed AAE-based NIDS achieves performance comparable to the MLP-based NIDS with only 0.1% of labeled data samples in the training dataset.

The next question is "*How many unlabeled data samples are required in the training dataset?*". Figure 4 shows that adding unlabeled data samples causes the

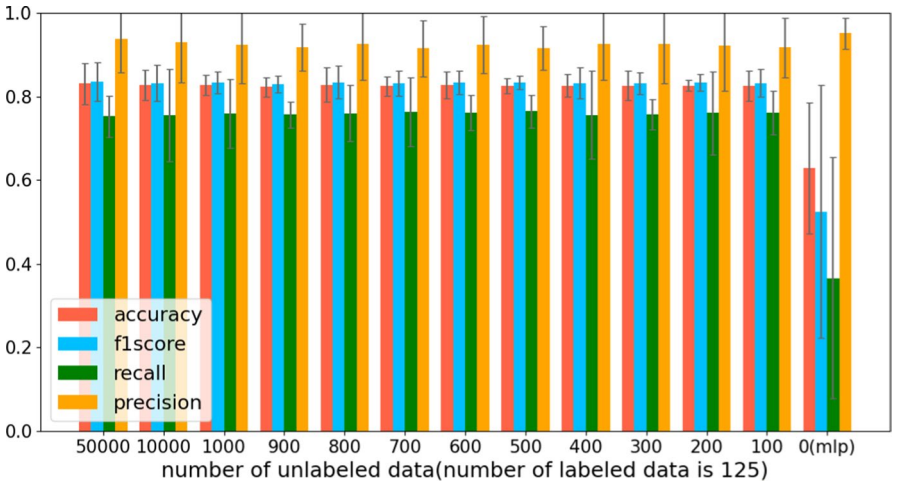

Fig. 4 Performance of the AAE-based NIDS (accuracy, precision, recall, and F1 score) as a function of the fraction of labeled data samples over the total data samples in KDDTrain+ (p_{sample}=0.001)

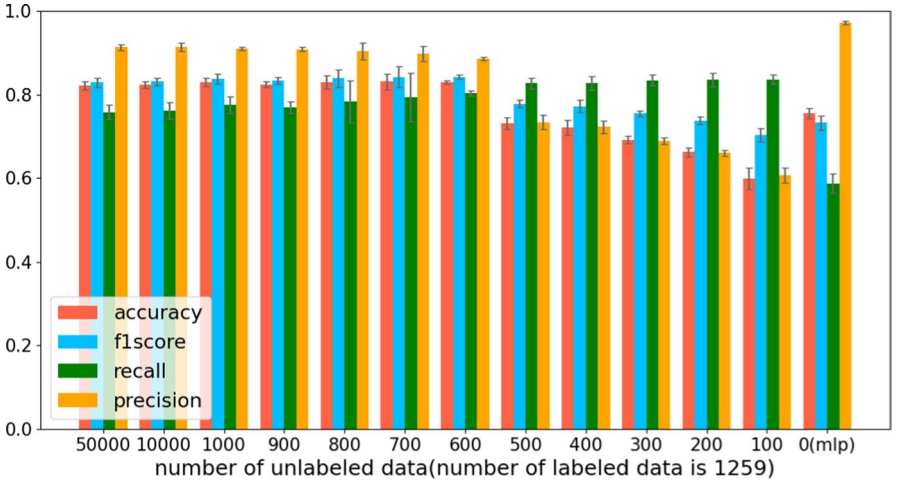

Fig. 5 Performance of the AAE-based NIDS (i.e., accuracy, precision, recall, and F1 score) as a function of the fraction of labeled data samples over the total data samples in KDDTrain+ (p_{sample}=0.01)

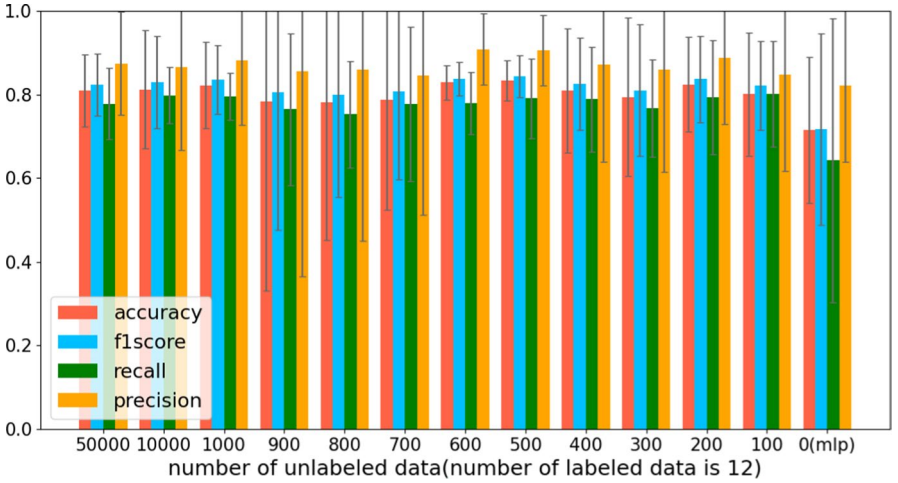

Fig. 6 Performance of the AAE-based NIDS (i.e., accuracy, precision, recall, and F1 score) as a function of the fraction of labeled data samples over the total data samples in KDDTrain+ (p_{sample}=0.0001)

performance of the AAE-based NIDS to improve. Thus, we confirm that the AAE-based NIDS takes advantage of the structure of the unlabeled data sample distribution to improve the accuracy of the boundaries between adjacent classes calculated from the labeled data samples. We note that the addition of a small number of unlabeled data samples effectively improves performance.

We investigated the effect of $p_{samples}$ on the AAE-based NIDS, to determine the number of labeled data samples required to improve performance. Figures 5 and 6 show the results with $p_{samples}$ = 0.01 and 0.0001 (i.e., using 1% and 0.01% of the

training data of NSL-KDD dataset as labeled samples). The horizontal axis represents the number of unlabeled data samples in Figs. 5 and 6. As observed, we confirm that the AAE-based NIDS improves its performance by increasing the number of unlabeled data samples. By comparing Figs. 5 and 6, we observe that both the AAE- and MLP-based NIDS yield better variable results with wider confidence intervals with $p_{samples} = 0.01$ than with $p_{samples}$=0.0001. We believe that the reason for this is as follows. When $p_{samples}$=0.0001, the number of labeled data sample is so small that the way the selected data samples are distributed has a strong influence on the performance of machine learning-based NIDS. That is, the number of labeled data samples per class is only 12. If we coincidentally select and label representative training samples, we might succeed in building a good model, whereas if we fail to select and label representative data samples, we might not. However, when $p_{samples}$ =0.01, the number of labeled data sample is sufficient to select and label representative data samples to avoid such variance. That is, the number of labeled data samples per class is 1,259. Thus, AAE- and MLP-based NIDS both yield higher variable results with wider confidence intervals at $p_{samples}$=0.0001 than those with $p_{samples}$ = 0.01.

5.4 Dimensionality of the Latent Variable

We are interested in how the proposed AAE successfully extracts feature representations of normal and anomaly traffic. Hence, we evaluated the effect of the dimensionality of the latent variable. Figure 7 shows the performance of the proposed AAE-based NIDS as a function of dimension sizes of $z_2 = 2$, 10, and 50. As observed in Fig. 7, $z_2 = 10$ achieves the best performance (i.e., recall and F1 score).

To investigate the mechanism behind the AAE successfully classifying data, we visualize how well the latent variable represents a distinctive set of features

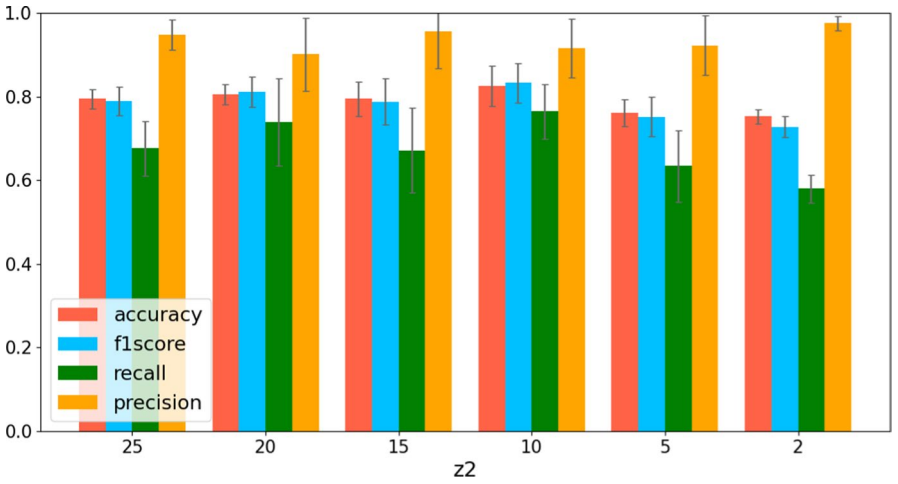

Fig. 7 Performance of the AAE-based NIDS (i.e., accuracy, precision, recall, and F1 score) as a function of the dimensionality of the AE, z_2

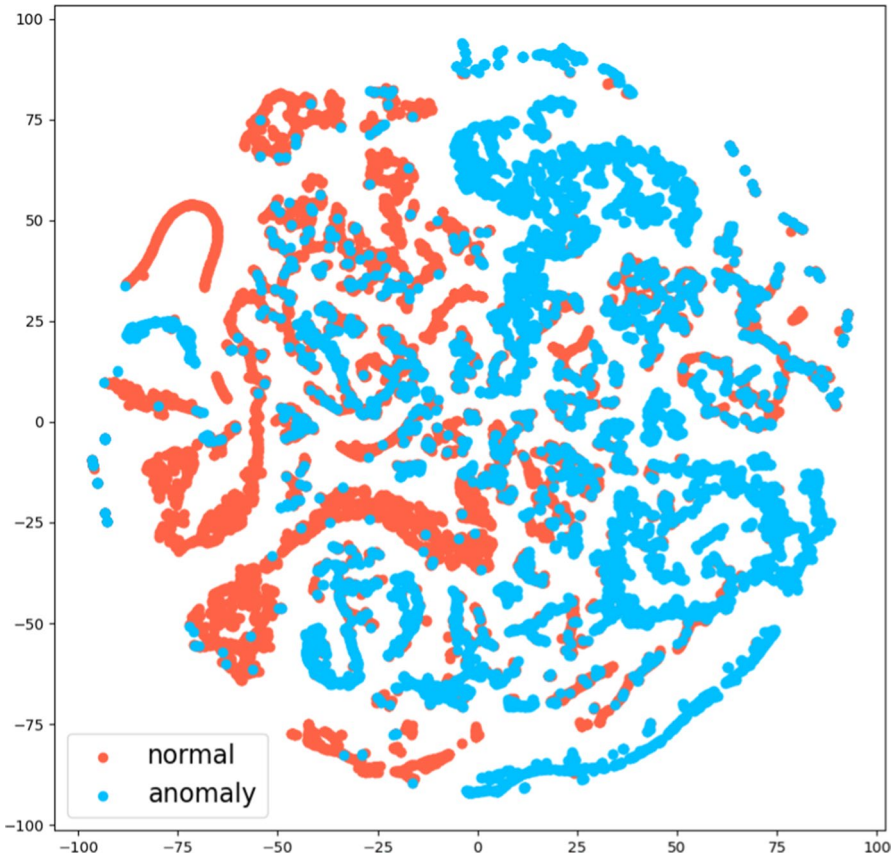

Fig. 8 Visualization of latent variable by t-SNE (z_2=2)

of input traffic. To visualize the multidimensional latent variable, we employed T-distributed stochastic neighbor embedding (t-SNE) [66] to reduce the dimensionality of the data. t-SNE is well suited for embedding high-dimensional data for visualization in a low-dimensional space. Specifically, it models each high-dimensional object by a 2- or 3-dimensional point in such a way that similar objects are modeled by nearby points, and dissimilar objects are modeled by distant points having high probability. We used a t-SNE with a perplexity of 50 and random state of zero. We used dimension sizes $z_2 = 2$, 10, and 50, with p_{sample}=0.001. Figures 8, 9, and 10 show how the latent variable z_2 represents a distinctive set of features from the input traffic when the dimensionality of z_2 is 2, 10, and 50. As observed in Fig. 9, $z_2 = 10$ produces the clearest separation between normal and attack designations. Figures 8 and 10 show that $z_2 = 2$ and 50 produce separations that are less clear.

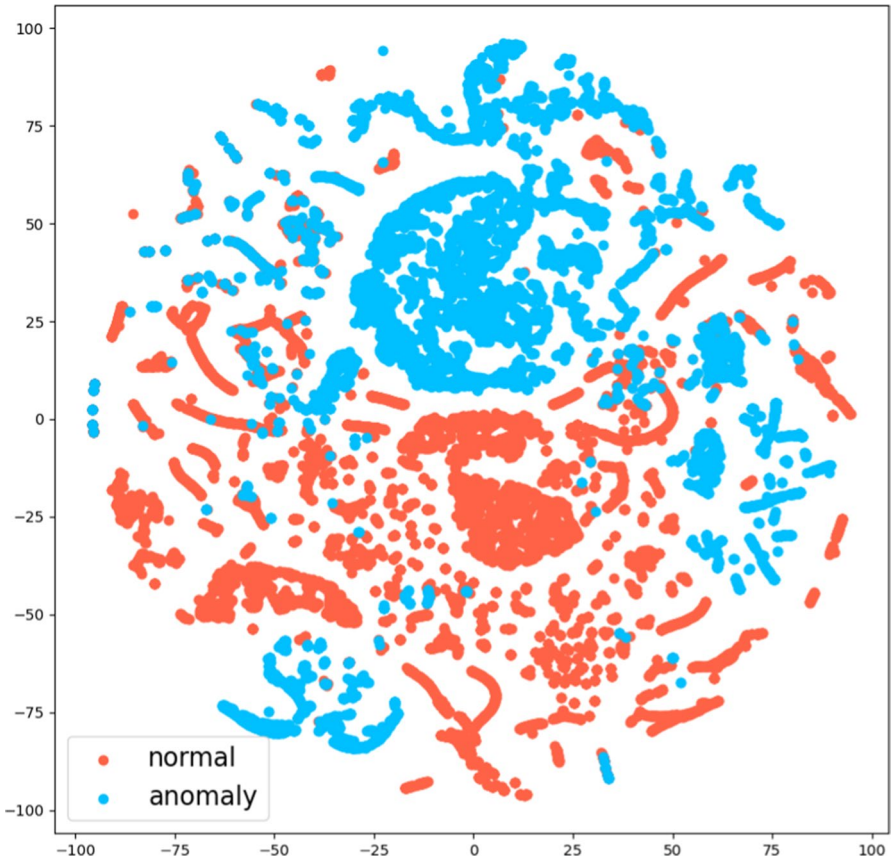

Fig. 9 Visualization of latent variable by t-SNE (z_2=10)

5.5 Computation Time

We next discuss the computation time required to train the AAE and MLP. Figure 11 shows the time required to train the AAE and MLP for various ratios of labeled data. The computing environment used in the experiment is shown in Table 2.

The learning rate and number of epochs were as shown in Table 3.

If the AAE is used for supervised learning, only the AE parameters need to be trained. Hence, as the ratio of labeled data increases, fewer training epochs are required and less computation time is needed to train the AAE. However, when the AAE uses unlabeled data, the two discriminators must be trained in addition to the AE training. As such, as the ratio of unlabeled data increases, more training epochs are required and more computation time is needed to train the AAE. As shown in Fig. 11, the ratio of labeled data is less than 5.0%, and the AAE requires more training time than the MLP. This is because the MLP trains one neural network, whereas the AAE trains one neural network for the AE and two for the discriminators.

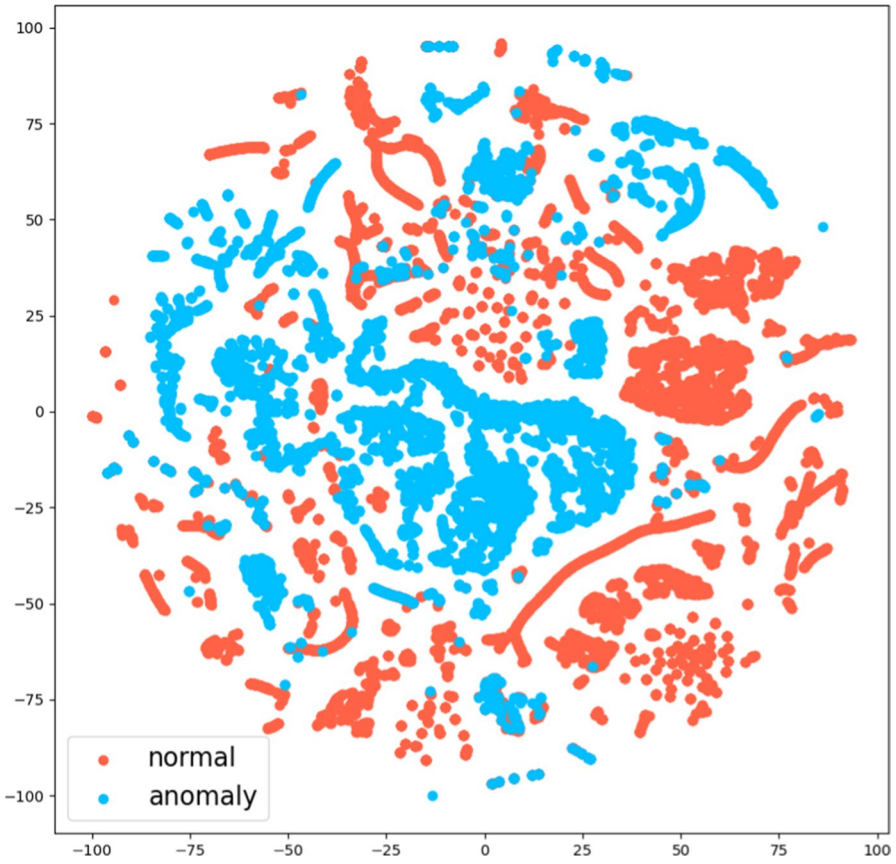

Fig. 10 Visualization of latent variable by t-SNE (z_2=50)

Table 2 Computing Environment

Hardware	
OS	Ubuntu 16.04
CPU	Intel Pentium@3.50 GHz
GPU	NVIDIA GeForce GTX1060 (6 GB)
RAM	8 GB
Software	
Programming language	Python (v.3.6)
Deep learning framework	Pytorch (v.0.4.1)
CUDA	version: 9.0

6 Conclusions

To reduce the required number of labeled training samples, we proposed a semi-supervised machine-learning-based NIDS that applies an AAE technique. We

Table 3 Hyperparameter values used for AAE and MLP training

Hyperparameter	Value
AAE	–
learning rate	1.0e-7
batch size	128
epochs (90%,10%)	30
epochs (1%)	300
epoch (0.1%,0.01%)	3000
dimensionality of the latent variable	z_1:2, z_2:50
MLP	
learning rate	1.0e-5
batch size	128
epochs (90%,10%,1%,0.1%,0.01%)	300

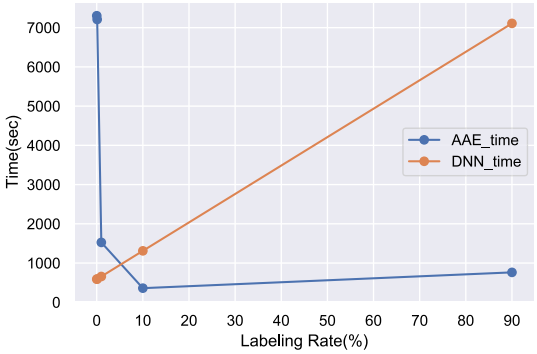

Fig. 11 Computing time required to train the AAE and MLP

evaluated the proposed method with a series of experiments and obtained the following results:

- The proposed AAE-based NIDS achieved performance comparable to that of an MLP-based NIDS with only 0.1% of the labeled data samples and the addition of a small number of unlabeled data samples.
- When the number of labeled data samples was small, the accuracy did not change, irrespective of the samples selected to be labeled; hence, the selection of data samples for annotation does not affect the performance of the proposed AAE-based NIDS.
- We demonstrated that the data structure dimensionality of z_2=10 successfully extracted the essential features from the data samples and produced the clearest separation between normal and attack classifications, resulting in the best performance in terms of recall and F1 score.

This study demonstrates promising results obtained by our novel semi-supervised learning method, which reduces the number of labeled training samples and greatly offsets the operational costs of a machine-learning-based NIDS.

Regarding future research directions, we should point out the applicability to real-world environments. In recent years, several publicly available datasets have been published [67–69]. These datasets may help us assess applicability to real-world environments even though efforts to assess the applicability of public datasets for NIDS evaluation is still ongoing [70]. More importantly, a prototype system will be deployed in a real-world environment.

Acknowledgements The author is grateful to Kazuki Hara for their contribution to the architectural design of the AAE-based NIDS during the early stage of this work. The author is grateful to Yuta Ichino for their assistance with the numerical simulations. This work was partially supported by the Grant-in-Aid for Scientific Research (Grant No. J19K11950) from the Japan Society for the Promotion of Science.

Funding This work is partially supported by the Grant-in-Aid for Scientific Research (Grant No. J19K11950) from the Japan Society for the Promotion of Science.

References

1. Landes, D., Otto, F., Schumann, S., Schlottke, F.: Identifying Suspicious Activities in Company Networks Through Data Mining and Visualization, pp. 75–90. Springer, London (2013)
2. Ring, M.., Wunderlich, S.., Grüdl, D., Landes, D., Hotho, A.: A Toolset for Intrusion and Insider Threat Detection, pp. 3–31. Springer International Publishing, Cham (2017)
3. Mishra, P., Varadharajan, V., Tupakula, U., Pilli, E.S.: A detailed investigation and analysis of using machine learning techniques for intrusion detection. IEEE Commun. Surv. Tutor. **21**(1), 686–728 (2019)
4. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative Adversarial Networks. ArXiv e-prints (2014)
5. Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., Frey, B.: Adversarial Autoencoders. ArXiv e-prints (2015)
6. Hara, K., Shiomoto, K.: Intrusion detection system using semi-supervised learning with adversarial auto-encoder. In: NOMS 2020–2020 IEEE/IFIP Network Operations and Management Symposium, pp. 1–8 (2020)
7. Bhuyan, M.H., Bhattacharyya, D.K., Kalita, J.K.: Network anomaly detection: methods, systems and tools. IEEE Commun. Surv. Tutor. **16**(1), 303–336 (2014)
8. Buczak, A.L., Guven, E.: A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Commun. Surv. Tutor. **18**(2), 1153–1176 (2016)
9. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the kdd cup 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp. 1–6 (2009)

10. Najafabadi, M. M., Khoshgoftaar, T. M., Kemp, C., Seliya, N., Zuech, R.: Machine learning for detecting brute force attacks at the network level. In 2014 IEEE International Conference on Bioinformatics and Bioengineering, pp. 379–385 (2014)
11. Shyu, M.L., Chen, S.C., Sarinnapakorn, K., Chang, L.: A novel anomaly detection scheme based on principal component classifier. In: Proceedings of the IEEE Foundations and New Directions of Data Mining Workshop, in conjunction with the Third IEEE International Conference on Data Mining (ICDM'03), pp. 172–179 (2003)
12. Pervez, M. S., Farid, D. M.: Feature selection and intrusion classification in nsl-kdd cup 99 dataset employing svms. In: The 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2014), pp. 1–6 (2014)
13. Kenaza, T., Bennaceur, K., Labed, A.: An efficient hybrid svdd/clustering approach for anomaly-based intrusion detection. In: Proceedings of the 33rd Annual ACM Symposium on Applied Computing, SAC '18, pp. 435–443. Association for Computing Machinery, New York, NY (2018)
14. Shahadat, N., Hossain, I., Rohman, A., Matin, N.: Experimental analysis of data mining application for intrusion detection with feature reduction. In: 2017 International Conference on Electrical, Computer and Communication Engineering (ECCE), pp. 209–216 (2017)
15. Andresini, G., Appice, A., Di Mauro, N., Loglisci, C., Malerba, D.: Exploiting the auto-encoder residual error for intrusion detection. In: 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS PW), pp. 281–290 (2019)
16. Tan, Z., Jamdagni, A., He, X., Nanda, P., Liu, R.P., Hu, J.: Detection of denial-of-service attacks based on computer vision techniques. IEEE Trans. Comput. **64**(9), 2519–2533 (2015)
17. Hemmer, A., Abderrahim, M., Badonnel, R., Chrisment, I.: An ensemble learning-based architecture for security detection in iot infrastructures. In: 2021 17th International Conference on Network and Service Management (CNSM), pp. 180–186 (2021)
18. Vanerio, J., Casas, P.: Ensemble-learning approaches for network security and anomaly detection. In: Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks, Big-DAMA '17, pp. 1–6. Association for Computing Machinery, New York, NY (2017)
19. Zoppi, T., Gharib, M., Atif, M., Bondavalli, A.: Meta-learning to improve unsupervised intrusion detection in cyber-physical systems. *ACM Trans. Cyber-Phys. Syst.*, 5(4) (2021)
20. Sundqvist, T., Bhuyan, M.H., Forsman, J., Elmroth, E.: Boosted ensemble learning for anomaly detection in 5g ran. In :Ilias. M., Lazaros, I., Elias P. (eds.) Artificial Intelligence Applications and Innovations, pp. 15–30. Springer International Publishing, Cham (2020)
21. Khare, S., Totaro, M.: Ensemble learning for detecting attacks and anomalies in iot smart home. In: 2020 3rd International Conference on Data Intelligence and Security (ICDIS), pp. 56–63 (2020)
22. Ma, T., Wang, F., Cheng, J., Yu, Y., Chen, X.: A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks. Sensors (Basel, Switzerland) **15**(10), 1701 (2016)
23. Tang, T.A., Mhamdi, L., McLernon, D., Zaidi, S.A.R., Ghogho, M.: Deep learning approach for network intrusion detection in software defined networking. In: 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM), pp. 258–263 (2016)
24. Tang, T.A., Mhamdi, L., McLernon, D., Zaidi, S.A.R., Ghogho, M.: Deep recurrent neural network for intrusion detection in sdn-based networks. In: 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), pp. 202–206 (2018)
25. Yin, C., Zhu, Y., Fei, J., He, X.: A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access **5**, 21954–21961 (2017)
26. Loukas, G., Vuong, T., Heartfield, R., Sakellari, G., Yoon, Y., Gan, D.: Cloud-based cyber-physical intrusion detection for vehicles using deep learning. IEEE Access **6**, 3491–3508 (2018)
27. Li, Z., Qin, Z., Huang, K., Yang, X., Ye, S.: Intrusion detection using convolutional neural networks for representation learning. In: Liu, D., Xie, S., Li, Y., Zhao, D., El-Alfy, E.-S.M. (eds.) Neural Information Processing, pp. 858–866. Springer International Publishing, Cham (2017)
28. Wang, W., Sheng, Y., Wang, J., Zeng, X., Ye, X., Huang, Y., Zhu, M.: Hast-ids: learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. IEEE Access **6**, 1792–1806 (2018)
29. Truong-Huu, T., Dheenadhayalan, N., Pratim Kundu, P., Ramnath, V., Liao, J., Teo, S.G., Praveen Kadiyala, S.: An empirical study on unsupervised network anomaly detection using generative adversarial networks. In: Proceedings of the 1st ACM Workshop on Security and Privacy on

Artificial Intelligence, SPAI '20, page 20–29. Association for Computing Machinery, New York, NY (2020)
30. Su, T., Sun, H., Zhu, J., Wang, S., Li, Y.: Bat: deep learning methods on network intrusion detection using nsl-kdd dataset. IEEE Access **8**, 29575–29585 (2020)
31. Madani, P., Vlajic, N.: Robustness of deep autoencoder in intrusion detection under adversarial contamination. In: Proceedings of the 5th Annual Symposium and Bootcamp on Hot Topics in the Science of Security, HoTSoS '18. Association for Computing Machinery, New York, NY (2018)
32. Zhou, C., Paffenroth, R.C.: Anomaly detection with robust deep autoencoders. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '17, pp. 665–674. Association for Computing Machinery, New York, NY (2017)
33. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.-A.: Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th International Conference on Machine Learning, ICML '08, pp. 1096–1103. Association for Computing Machinery, New York, NY (2008)
34. Gehring, J., Miao, Y., Metze, F., Waibel, A.: Extracting deep bottleneck features using stacked auto-encoders. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 3377–3381 (2013)
35. Qi, Y., Wang, Y., Zheng, X., Wu, Z.: Robust feature learning by stacked autoencoder with maximum correntropy criterion. In: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6716–6720 (2014)
36. Vartouni, A.M., Kashi, S.S., Teshnehlab, M.: An anomaly detection method to detect web attacks using stacked auto-encoder. In: 2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS), pp. 131–134 (2018)
37. Shone, N., Ngoc, T.N., Phai, V.D., Shi, Q.: A deep learning approach to network intrusion detection. IEEE Trans. Emerg. Topics Comput. Intell. **2**(1), 41–50 (2018)
38. Khan, F.A., Gumaei, A., Derhab, A., Hussain, A.: A novel two-stage deep learning model for efficient network intrusion detection. IEEE Access **7**, 30373–30385 (2019)
39. Al-Qatf, M., Lasheng, Y., Al-Habib, M., Al-Sabahi, K.: Deep learning approach combining sparse autoencoder with svm for network intrusion detection. IEEE Access **6**, 52843–52856 (2018)
40. Ahmad, J., Niyaz, Q., Sun, W., Alam, M.: A deep learning approach for network intrusion detection system. In: Proceedings of the 9th EAI International Conference on Bio-Inspired Information and Communications Technologies (Formerly BIONETICS), BICT'15, pp. 21-26. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Brussels, BEL (2016)
41. Niyaz, Q., Sun, W., Javaid, A.Y.: A deep learning based ddos detection system in software-defined networking (SDN). CoRR, abs/1611.07400 (2016)
42. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.-A.: Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. J. Mach. Learn. Res. **11**, 3371–3408 (2010)
43. Aygun, R.C., Yavuz, A.G.: Network anomaly detection with stochastically improved autoencoder based models. In: 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud), pp. 193–198 (2017)
44. Yu, Y., Long, J., Cai, Z.: Session-based network intrusion detection using a deep learning architecture. In: Vicenç, T., Yasuo, N., Aoi, H., Sozo, I., (eds.), Modeling Decisions for Artificial Intelligence, pp. 144–155. Springer International Publishing, Cham (2017)
45. Yu, Y., Long, J., Cai, Z.: Network intrusion detection through stacking dilated convolutional autoencoders. Secur. Commun. Netw. (2017)
46. Rifai, S., Vincent, P., Muller, X., Glorot, X., Bengio, Y.: Contractive auto-encoders: Explicit invariance during feature extraction. In: Proceedings of the 28th International Conference on International Conference on Machine Learning, ICML'11, pp. 833–840. Omnipress, Madison, WI (2011)
47. Wang, W., Du, X., Shan, D., Qin, R., Wang, N.: Cloud intrusion detection method based on stacked contractive auto-encoder and support vector machine. IEEE Trans. Cloud Comput., p. 1 (2020)
48. Salama, M.A., Eid, H.F., Ramadan, R.A., Darwish, A., Hassanien, A.E.: Hybrid intelligent intrusion detection scheme. In: António, G.-C., Ricardo, T., Gerald, S., Lino, C., (eds.) Soft Computing in Industrial Applications, pp. 293–303. Springer, Berlin, Heidelberg (2011)

49. Raina, R., Battle, A., Lee, H., Packer, B., Ng, A.Y..: Self-taught learning: transfer learning from unlabeled data. In: Proceedings of the 24th International Conference on Machine Learning, ICML '07, pp. 759–766. Association for Computing Machinery, New York, NY (2007)
50. Papamartzivanos, D., Mármol, F.G., Kambourakis, G.: Introducing deep learning self-adaptive misuse network intrusion detection systems. IEEE Access **7**, 13546–13560 (2019)
51. Lake, B.M., Salakhutdinov, R., Gross, J., Tenenbaum, J.B.: One shot learning of simple visual concepts. Cognit. Sci., 33 (2011)
52. Koch, G.., Zemel, R., Salakhutdinov, R. et al.: Siamese neural networks for one-shot image recognition. In: ICML Deep Learning Workshop, vol. 2. Lille (2015)
53. Hsiao, S.-C.., Kao, D.-Y., Liu, Z.-Y., Tso, R.: Malware image classification using one-shot learning with siamese networks. In: Procedia Comput. Sci., **159**, 1863–1871 (2019). (Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 23rd International Conference KES2019)
54. Tran, T.K., Sato, H., Kubo, M.: Image-based unknown malware classification with few-shot learning models. In 2019 Seventh International Symposium on Computing and Networking Workshops (CANDARW), pp. 401–407 (2019)
55. Tang, Z., Wang, P., Wang, J.: Convprotonet: deep prototype induction towards better class representation for few-shot malware classification. Appl. Sci., 10(8) (2020)
56. Hindy, H., Tachtatzis, C., Atkinson, R., Bayne, E., Bellekens, X.: Developing a siamese network for intrusion detection systems. In: Proceedings of the 1st Workshop on Machine Learning and Systems, EuroMLSys '21, pp. 120–126. Association for Computing Machinery, New York, NY (2021)
57. Osada, G., Omote, K., Nishide, T.: Network intrusion detection based on semi-supervised variational auto-encoder. In: Simon, N.F., Dieter, G., Einar, S. (eds.) Computer Security—ESORICS 2017, pp. 344–361. Springer International Publishing, Cham (2017)
58. Said Elsayed, M., Le-Khac, N.-A., Dev, S., Jurcut, A.D.: Network anomaly detection using lstm based autoencoder. In: Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks, Q2SWinet '20, pp. 37–45. Association for Computing Machinery, New York, NY (2020)
59. Aouedi, O., Piamrat, K., Bagadthey, D.: A semi-supervised stacked autoencoder approach for network traffic classification. In: 2020 IEEE 28th International Conference on Network Protocols (ICNP), pp. 1–6 (2020)
60. Aouedi, O., Piamrat, K., Muller, G., Singh, K.: Fluids: federated learning with semi-supervised approach for intrusion detection system. In: 2022 IEEE 19th Annual Consumer Communications Networking Conference (CCNC), pp. 523–524 (2022)
61. Kayacik, H.G., Zincir-Heywood, A.N., Heywood, M.I.: Selecting features for intrusion detection: a feature relevance analysis on kdd 99. In: PST (2005). Accessed 13 April 2020
62. Kayacik, H.G., Zincir-Heywood, A.N., Heywood, M.I.: Selecting features for intrusion detection: a feature relevance analysis on kdd 99. In: PST (2005)
63. Pytorch.: https://github.com/pytorch/pytorch (2022)
64. Kingma, D.P., Ba, J.A.: A method for stochastic optimization. arXiv:1412.6980 (2014)
65. Sommer, R., Paxson, V.: Outside the closed world: On using machine learning for network intrusion detection. In: 2010 IEEE Symposium on Security and Privacy, pp. 305–316 (2010)
66. van der Maaten, L., Hinton, G.: Visualizing data using t-sne. J. Mach. Learn. Res. **9**, 2579–2605 (2008)
67. Song, J., Takakura, H., Okabe, Y., Eto, M., Inoue, D., Nakao, K.: Statistical analysis of honeypot data and building of kyoto 2006+ dataset for nids evaluation. In: Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, BADGERS'11, pp. 29–36. Association for Computing Machinery, New York, NY (2011)
68. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In Proceedings of the 4th International Conference on Information Systems Security and Privacy—Volume 1: ICISSP, pp. 108–116. INSTICC, SciTePress (2018)
69. Gharib, A., Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: An evaluation framework for intrusion detection dataset. In: 2016 International Conference on Information Science and Security (ICISS), pp. 1–6 (2016)
70. Ring, M., Wunderlich, S., Scheuring, D., Landes, D., Hotho, A.: A survey of network-based intrusion detection data sets. CoRR, abs/1903.02460 (2019)

Kohei Shiomoto **Kohei Shiomoto** is a Professor, Tokyo City University, Tokyo Japan. Since joining NTT Laboratories in 1989, he has been engaged in research and development in the data communications industry on high-speed computer network architecture, traffic management, and network analysis. From 1996 to 1997, he was a visiting scholar at Washington University in St. Louis, MO, USA. In 2017, he joined Tokyo City University to engage in research and education on data science and computer networking. Current research interests include data mining for network management, human flow analysis, cloud computing and blockchain. He is a Fellow of the Institute of Electronics, Information and Communication Engineers (IEICE), a Senior Member of the IEEE, and a member of the ACM and the Information Processing Society of Japan (IPSJ).