

Intrusion Detection System using Semi-Supervised Learning with Adversarial Auto-encoder

Kazuki Hara

Graduate School of Systems and Information Engineering
University of Tsukuba
Ibaraki, Japan
s1920593@s.tsukuba.ac.jp

Kohei Shiomoto

Faculty of Knowledge Engineering
Tokyo City University
Tokyo, Japan
shiomoto@tcu.ac.jp

Abstract—As computer networks become vulnerable to attacks from intruders, Intrusion Detection Systems (IDS) is a critical component that monitors activities of computer networks and classifies them as either normal or anomalous. Remarkable advancement of machine learning makes us consider to use supervised machine learning to build IDS. Supervised machine learning requires a large amount of training data, leading to costly human-labor operation; it requires the human operator to examine data, classify them, and annotate them with a label. To address this issue, we propose an IDS that employs semi-supervised learning. Semi-supervised learning uses a small number of labeled data in training dataset to reduce costly human-labor tasks and improves the performance with support of unlabeled data in training dataset. The proposed method employs Adversarial Auto-encoder (AAE), a semi-supervised learning algorithm that incorporates the Generative Adversarial Nets (GAN) into the Auto-encoder (AE). We evaluate the effectiveness of the proposed method using NSL-KDD dataset. We confirm that the proposed method that uses only 0.1 percent of labeled data achieves comparable performance with existing IDSs that use machine learning methods.

Index Terms—Intrusion Detection System, Machine learning, Semi-supervised learning, Adversarial Auto-encoder

I. INTRODUCTION

A. Motivation

Computer networks have developed rapidly over the years contributing significantly to social and economic development. This has led to an increasing interest in the security of computer networks. Denial of Service (DoS) attack denies or prevents legitimate users resource on a network by introducing unwanted traffic. Malware is another attack, where attackers use malicious software to disrupt systems. An IDS is a detection system put in place to monitor computer networks. IDS monitors activities of computer and network systems and classifies them as either normal or anomalous. By analyzing patterns of data, IDS helps to detect threats that can be devastating.

Machine learning has been used extensively in computer science for voice, face and image recognition. The natural question is how we can apply machine learning to the IDS. There are two causes that hinder its practical application of machine learning to the IDS. Firstly, the required accuracy is high because false detection risk is very high on IDS [13].

This problem has been gradually improved due to recent advancement in deep learning. Secondly, many supervised machine learning methods need to be trained with training data annotated with the correct labels. The number of training data is critical. In order to build a good supervised machine learning model, we need a large number of training data. Labeling data is, however, a costly task. Human operator examines data, classifies them, and annotates them with an appropriate label. In addition, trends in network traffic that are subject to IDS change from day to day, and new attacks continue to generate. For that reason, labeling work needs to be done many times, and it is difficult to keep a sufficient amount of label data.

To address this issue, we propose an IDS that employs semi-supervised learning based on Adversarial Auto-encoder (AAE). Semi-supervised learning uses a small number of labeled data in training dataset to reduce costly human-labor tasks and improves the performance with support of unlabeled data in training dataset. AAE incorporates the Generative Adversarial Nets (GAN) [3] into the Auto-encoder [7]. The AE reduces the dimension of input data by extracting and maintaining important features as the latent variable vector z in the hidden layer whereas the GAN employs the generator and the discriminator in such a way that the latent variable vector z hold in the hidden layer of the AE follows an arbitrary distribution for regularization. GAN incorporates a generative model and a discriminator to distribute random variables in the middle hidden layer that represents features with reduced dimension. Two types of latent variable vectors z_1 and z_2 are hold in the middle layer of the AE. The encoder $q(z_1, z_2|x)$ generates two latent variable vectors z_1 and z_2 to hold the features of input data; the latent variable vector z_1 holds the features representing the class information (“normal” or “attack” in our method). The latent variable vector z_2 holds the other features. As such, a categorical distribution is used to draw the latent variable vector z_1 to separate and learn only class information whereas a Gaussian distribution is used to draw the latent variable vector z_2 . We evaluate the effectiveness of the proposed method using NSL-KDD dataset and confirm that the proposed method that uses only 0.1 percent of labeled data achieves comparable performance with existing IDSs that use machine learning methods.

Contributions of this paper are the following:

- We propose an IDS that employs a semi-supervised learning with the Adversarial Auto-Encoder (AAE).
- We demonstrate that the proposed semi-supervised learning yields comparable accuracy with the existing machine learning methods and reduces the cost of labeled data since it requires less number of training data. Our approach realizes a detection rate of 82.78% using 1.0% of labeled data compared to other state-of-the-art approaches.

The rest of this paper is organized as follows. Section II describes the related work. Section III explains auto-encoder (AE) and generative adversarial network (GAN) that are core building blocks of AAE. Section IV presents the proposed semi-supervised machine learning method that employs the AAE. Section V presents experimental environment. Section VI presents the detection performance. Section VII presents the computing time. Section VIII concludes the paper.

II. RELATED WORKS

IDS can be classified as either signature-based or anomaly-based. Signature-based IDS analyzes known attacks to extract feature and patterns and make a database based on this. The advantages of signature-based IDS are that it has a high detection rate and a low misdetection for known attacks. On the other hand, the disadvantage is that it cannot detect unknown and new attacks. Anomaly-based IDS compares new data with a model of normal user behavior and marks a significant deviation from this model as an anomaly. Anomaly-based IDS can use machine learning to build a classifier model based on network traffic features. The advantage of anomaly-based detection is that it can detect unknown and new attacks that have never been seen before. Various machine learning methods are proposed for IDS [10], [11], [14]–[19].

Tavallaee et al. [16] proposed a new data set called NSL-KDD dataset to address the inherent issues in KDDCUP 99 dataset by removing huge number of redundant records and creating the new train and test data which consist of selected records of the complete KDDCUP 99 dataset. They used this to experiment and presented a benchmark with seven machine learning techniques, namely, J48 decision tree learning, Naive Bayes, NBTree, Random Forest, Random Tree, Multilayer Perceptron and Support Vector Machine(SVM). Support Vector Machine (SVM)-based IDS is proposed [10]. Feature selection and classification of multiple class using NSL-KDD dataset is discussed.

Research using the ensemble method has been conducted, and high accuracy has been achieved. Verma et al. used both clustering and boosting techniques [17]. Dividing the data into several clusters using k -means clustering, they classify by XGBoost for data in each cluster. As a result, 84.25 % accuracy was achieved. Pham et al. used AdaBoost and Bagging with various decision tree methods [11]. They also conducted feature selection. 25 features are selected from 41 features by using the *leave-one-out* techniques and Naive Bayes classifiers [8], and 35 features are selected by using Gain Ratio (GR) technique [4]. As a result, they achieved the

best performance (84.25%) by bagging model that used J48 as the base classifier and worked on 35 features.

Deep learning has been attracting attention in IDS research in recent years. Convolutional neural networks (CNNs) and long short term memory (LSTM) networks for feature extraction to capture spatio-temporal features is employed in [18]. They used DARPA 1998 [6] dataset and ISCX (Information Security Center for Excellence of the University of New Brunswick) 2012 dataset [12] to demonstrate that the proposed method achieves the efficient feature learning and reduces the false alarm ratio. In [19], the authors proposed a recurrent neural network (RNN) based IDS and evaluated the accuracy of the proposed method with a range of number of nodes in hidden layer and learning rates for binary and multiclass classification tasks using NSL-KDD dataset. In [15] and [14], the authors applied Deep Neural Network (DNN) and Gated Recurrent Unit Recurrent Neural Network (GRU-RNN) under the context of SDNs with the NSL-KDD dataset. They demonstrated that the proposed method achieves the accuracy of 89% with just six basic features with GRU-RNN.

The IDS based on deep learning is powerful to improve the accuracy of IDS, and we consider to apply deep learning to the IDS. Because many methods use supervised learning that requires a lot of labeled data to realize high accuracy. We therefore aim not only to improve the accuracy but also to reduce the cost of labeled data which is required to train supervised learning classifiers. We propose an IDS that employs a semi-supervised learning with the Adversarial Auto-Encoder (AAE). The recent work applies semi-supervised learning for the IDS using Variational Auto-Encoder (VAE) [9]. VAE assumes that the underlying distribution for the latent variables is a Gaussian while the AAE assumes that an arbitrary distribution is used the underlying distribution for the latent variables.

III. MACHINE LEARNING

The Adversarial Auto-encoder (AAE) is the core building block of our proposed method that incorporates the Generative Adversarial Nets (GAN) [3] into the Auto-encoder [7]. We explain the AE and GAN in this section.

A. Auto-encoder (AE)

Auto-encoder (AE) is an unsupervised learning that uses a neural network. AE is trained such that output data (X') becomes equal to input data (X) (See Fig. 1). The purpose of AE is to reduce dimension of features; features of input is compressed and preserved in a latent variable vector (z) in the hidden middle layer of the neural network.

This method consists of two neural network: an encoder and a decoder. Firstly, the encoder reduces the dimension of the input data. Compressed dimension is preserved in a latent variable vector (z) in the hidden middle layer. Secondly, the decoder expands the dimension of the latent variable vector z to restore the original input data. In order to minimize the loss between input and output data, the parameters of the neural networks associated with the encoder and decoder are trained.

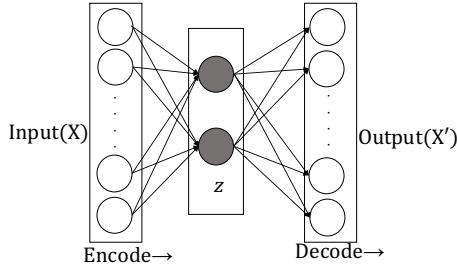


Fig. 1. Architecture of an Auto-encoder

The purpose of this method is to reduce the dimension of input data while maintaining the important feature of data. In other words, only the important features and structures of data is extracted and held in low dimensions as a latent variable vector z in the hidden middle layer. The encoder and decoder are defined as follows:

$$\text{encoder} : z = f_{\theta}(x) = s(Wx + b) \quad (1)$$

$$\text{decoder} : x^r = g_{\theta}(z) = s_r(W'z + c) \quad (2)$$

x is input data and x^r is output data restored from the latent variable. s and s_r are activation functions. The reconstruction layer x^r is trained to be equal to the input layer x . That is, it is trained to optimize the parameters weights (W and W') and biases (b and c) to minimize reconstruction errors.

The middle layer z in the neural network is a latent variable, and it captures the essential features of the input data with a lower dimension than that of the input data. Using this latent variable, we classify normal and anomaly communications. Furthermore, we classify them by assuming an arbitrary prior distribution for this latent variable z .

B. Generative Adversarial Networks (GAN)

Fig. 2 shows Generative Adversarial Networks (GAN) [3]. GAN automatically learns the regularities or patterns in input data in such a way that the model can be used to generate new examples that plausibly could have been drawn from the original dataset. GAN is used to make the latent variables in the AE follow an arbitrary distribution in Adversarial Auto-encoder (AAE) as explained later.

GAN consists of two neural network models: Generator and Discriminator. The generator G tries to capture the training data distribution. It is trained to generate data (X_{fake}) similar to training data (X_{real}). In other words, the training procedure for the generator G is to maximize the probability that the discriminator D makes a mistake. On the other hand, the discriminator D tries to correctly judge whether the object data is training data (X_{real}) or data (X_{fake}) generated by the generator G . It is trained by maximizing the probability that it can correctly discriminate whether a sample comes from the training data rather or the generator G .

The purpose of GAN is to train two models alternately to adversarially compete for generation and discrimination. Eventually, the generator G becomes able to generate data

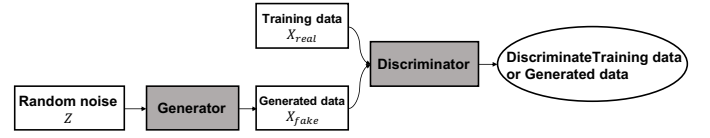


Fig. 2. Architecture of the Generative Adversarial Networks

(X_{fake}) with features similar to training data (X_{real}). Therefore, it is desirable that the probability that is the output of the discriminator D is 50%, and it is hoped for that the generator G generates data (X_{fake}) that matches the distribution of learning data (X_{real}). The technique of GAN is defined as follows:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}} [\log D(x)] + \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))] \quad (3)$$

In the original generative adversarial network (GAN) paper [3], the authors define a prior on input noise variables $p_z(z)$, then employ a mapping to data space as $G(z; \theta_g)$, where G is a differentiable function represented by a multilayer perceptron (MLP) with parameters θ_g . They also define a second MLP $D(x; \theta_d)$ that outputs a single scalar, where $D(x)$ represents the probability that x comes from the data rather than p_g . The discriminator D is trained to maximize the probability of assigning the correct label to both training examples and samples from the generator G . In addition, the generator G is trained simultaneously to minimize $\log(1 - D(G(z)))$.

IV. PROPOSED METHOD

A. Semi-supervised learning

Supervised learning requires *big* data to build a good classifier. The more training data we use, the better performance we get for the supervised machine learning classifier. Unfortunately obtaining large amount of training data is a costly process. Semi-supervised learning requires just a small set of labeled data in training dataset. If we use only a small set of labeled data, we may obtain poor performance for supervised learning classifier. In order to improve the performance, we consider to use unlabeled data in training data set in semi-supervised learning. Unlabeled data does not require costly human-labor tasks. We hope that we could improve the performance of the classifier in support of unlabeled data in training dataset. This is a fundamental motivation of our use of semi-supervised learning.

Fig. 3 illustrates a comparative example of semi-supervised learning and supervised learning when there are a small amount of labeled data. We are trying to solve a two-class classification task, where each data is represented by a set of two features. We have a set of labeled data in training dataset as shown in Fig. 3. Since data associated with label are scattered, we may want to draw the straight line (in red) as shown in Fig. 3. Then since we have large amount of “unlabeled data” in training dataset as shown in Fig. 3, we will draw a curve (in green) that nicely separates entire training

dataset into two clusters as shown in Fig. 3. Note that semi-supervised learning does NOT require *big* labeled data that requires costly human-labor tasks.

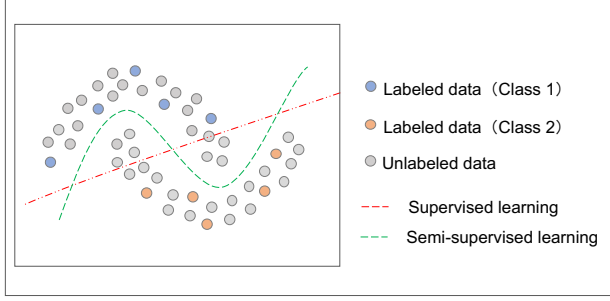


Fig. 3. Description of the behavior of the semi-supervised learning approaches

B. Adversarial Auto-encoder (AAE)

Fig. 4 shows an architecture of an Adversarial Auto-encoder (AAE). AAE employs AE and GAN as a key building block. The AE reduces the dimension of input data by extracting and maintaining important features as the latent variable vector z whereas the GAN employs the generator and the discriminator in such a way that the latent variable vector z of the AE follows an arbitrary distribution for regularization. Latent variable vector z 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 latent variable z . Let x be the input and z be the latent variable of the AE. Let $p(z)$ be the prior distribution we want to impose on the latent variable, $q(z|x)$ be an encoding distribution and $p(x|z)$ be a decoding distribution. The encoding function of the AE $q(z|x)$ defines an aggregated posterior distribution of $q(z)$ on the latent variable of the AE as follows:

$$q(z) = \int_x q(z|x)p_d(x)dx \quad (4)$$

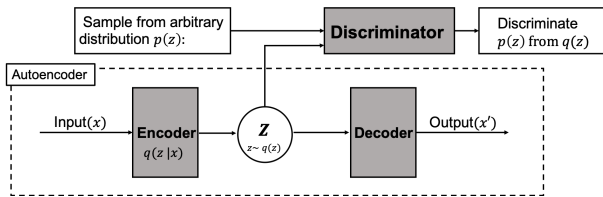


Fig. 4. Architecture of an adversarial auto-encoder.

As the learning progresses, the AAE can match an arbitrary prior $p(z)$ to the aggregated posterior $q(z)$ of latent variable vector z ; the latent variable vector z follows the prior distribution. In other words, the encoder of AE is regarded as the generator of GAN. As such the latent variable vector z of AE is regarded as the generated data by the generator. The discriminator discriminates the latent variable vector z generated by the AE that follows the distribution of $q(z)$ from a sample generated by the generator that follows the distribution of $p(z)$. It yields the probability d that input data

comes from the sample of arbitrary prior $p(z)$. The adversarial network and the AE are trained jointly with stochastic gradient descent (SGD) method.

C. Semi-Supervised AAE

The AAE is used as semi-supervised learning. Fig. 5 shows an architecture of an AAE for semi-supervised learning. The encoder $q(z_1, z_2|x)$ generates two latent variable vectors z_1 and z_2 to hold the features of input data; the latent variable vector z_1 holds the features representing the class information (“normal” or “attack” in our method) whereas the latent variable vector z_2 holds the other features. As such, a categorical distribution $cat(z_1)$ is used to draw the latent variable vector z_1 to separate and learn only class information whereas a Gaussian distribution $N(z_2|0, I)$ is used to draw the latent variable vector z_2 . The categorical distribution is a distribution that takes the same number of one-hot values as the number of the classes; two in our method “normal” or “attack”. The latent variable (z_1) is learned by the Discriminator to hold a one-hot vector (if it is not perfect, one value is close to 1 and the other values are close to 0). The latent variable (z_1) holds the class information. Therefore, classification can be performed by referring to the value of z_1 estimated by the encoder. The latent variable vector (z_2) imposes a Gaussian distribution. We assume that there should be complex latent expressions behind normal and anomaly communications. The NSL-KDD dataset includes four attacks types as anomaly communication: Probe, Dos, U2R, and R2L. They are classified into 40 detailed attack types. Similarly, normal communication also has different latent features depending on the service and protocol used. It is difficult to accurately represent the latent features of these attack types using only the latent variable vector z_1 . As such, we use the latent variable vector (z_2) that imposes the Gaussian distribution in order to preserve detailed features other than class information. We assumed that the latent variable vector (z_2) is based on Gaussian distribution with the mean of 0, the standard deviation is 1, and the number of dimensions is 50.

The proposed AAE employs two pairs of generative model and discriminator to regularize those two latent variables z_1 and z_2 : one follows a categorical distribution and the other a Gaussian distribution. 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 place clusters apart from one another. Therefore, it is assumed that input data is 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. We train the AAE using unlabeled data. When a labeled data is available, we train the AAE by using the label instead of the categorical generative model. Once the AAE is trained, it is used to classify new coming 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 is normal or anomaly.

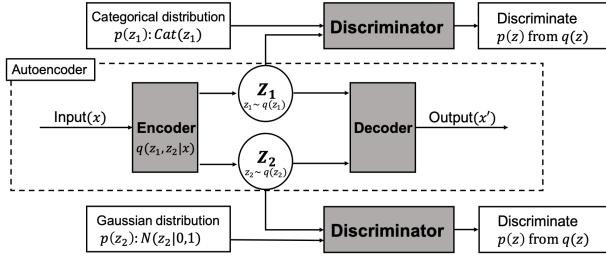


Fig. 5. Architecture of an adversarial auto-encoder (semi-supervised learning).

The middle of Fig. 5 is the neural network of AE that reduces dimension of input data ($Input(X)$) and generates latent variable vectors (z_1, z_2). The top of Fig. 5 is the neural network of discriminator that imposes a categorical distribution to a latent class variable vector z_1 as a prior distribution. Therefore, the distribution of a latent class variable vector z_1 is ensured to match a categorical distribution. Hence only class information that is required to detect attack is extracted from two latent variable vectors. The bottom of Fig. 5 is the neural network of discriminator that imposes a Gaussian distribution to a latent class variable z_2 as a prior distribution. Therefore, the distribution of a latent variable vector z_2 is ensured to match a Gaussian distribution. The semi-supervised AAE is trained with the stochastic gradient descent (SGD) in three phases as follows:

- 1) Reconstruction phase: Updates the encoder and the decoder. Optimizes the parameters of the AE in order to minimize reconstruction error of input data and output data. Using unlabeled data only 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 in order to discriminate the latent variable vector z_1 or sample of the categorical distribution, and to discriminate the latent variable vector z_2 or sample of the Gaussian distribution. The AE is optimized based on the determination result of discriminator. This training based on Eq.(4).
- 3) Semi-supervised classification phase: The AE is updated to minimize the cross-entropy error on labeled data. In this phase, realize the semi-supervised learning using labeled data.

After the AAE is trained, the latent variable z_1 is used to classify the input data is a normal one or an attack.

V. EXPERIMENTAL ENVIRONMENT

A. Dataset

We have used NSL-KDD dataset [16] that is widely used in performance evaluation other IDS methods. Although it is quite old and not a perfect representative of existing real networks, it is still a good reference. It has been used in the past to evaluate the performance of IDS by many researchers, so there are significant performance measurement results for comparison. This dataset consists of records of traffic sent and

received between the source and destination IP address, and is divided into KDDTrain+ (125,973 data records) for train and KDDTest+ (22,544 data records) for evaluation. Each traffic sample has 41 features that are categorized into three types of features: basic features, content-based features, traffic-based features. Among 41 features, some of them are categorical such as protocol type that takes three values (tcp, udp, icmp), flag that takes 11 values (SF, S1, REJ, etc.), and service that takes 70 values (http, telnet, ftp, etc.). Instead of coding each categorical data into a scalar value, we adopt an one-hot vector representation, resulting in 122 features.

All data are labeled for normal or anomaly. The KDDTrain+ have 22 types of data, and the KDDTest+ have 38 types of data. KDDTest+ have 17 types of attack data that are not in KDDTrain+, and KDDTrain+ have 2 types of attack data that are not in KDDTest+. Therefore, the KDDTest+ dataset is a reliable indicator of the performance to Zero-day attack and new attacks that are variants of existing attacks as well.

B. Evaluation Metrics

In order to evaluate the proposed method, Accuracy (ACC), True Positive Rate (TPR), and False Positive Rate (FPR) are used. These performance measure are calculated from four different metrics, true positive (TP), true negative (TN), false positive (FP), and false negative (FN):

- TP: the number of anomaly records correctly classified.
- TN: the number of normal records correctly classified.
- FP: the number of normal records incorrectly classified.
- FN: the number of anomaly record incorrectly classified.

Accuracy (AC): the percentage of true detection over total traffic records,

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (5)$$

False Negative Rate (FNR): as the equivalent of the Detection Rate (DR), it shows the percentage of the number of records incorrectly over the total number of anomaly records.

$$FNR = \frac{FN}{TP + FN} \quad (6)$$

False Positive Rate (FPR): the percentage of the number of records incorrectly is divided by the total number of normal records

$$FPR = \frac{FP}{FP + TN} \quad (7)$$

C. Parameters

We present the detection performance of the proposed method based on AAE in terms of Accuracy. We also compare the performance of the AAE with DNN. We computed the mean accuracy from ten times experiments. For the AAE training, experiments were conducted using the defined percentage samples from KDDTrain+ (125,973 data records) as labeled data and the rest as unlabeled data. On the other hand, DNN was used as a comparison, all KDDTrain+ were used as labeled data for training. KDDTest+ (22,544 data records) was used for evaluation in both methods.

We implemented the proposed method based on AAE and the method based on DNN using Pytorch [2]. We use a ADAM optimizer [5] for both models. In addition, we added dropout and batch normalization to prevent over-fitting during the training phase. The detail of our models that AAE and DNN are shown in Fig. 6. Regarding the AAE, the encoder has 122 inputs, compresses them into the important features, and yields 52 ($=2+50$) outputs (2 for z_1 and 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 the size of 1000×1000 . They also have a dropout and batch normalization between layers. The discriminator for the categorical distribution (z_1) receives two inputs and yields one outputs (“Fake” or “Real”). The discriminator for the Gaussian distribution (z_2) receives 50 inputs and yields one outputs (“Fake” or “Real”). Both the discriminators for the categorical distribution (z_1) and the Gaussian distribution (z_2) have a middle fully connected layer with the size of 1000×1000 . They also have batch normalization between layers.

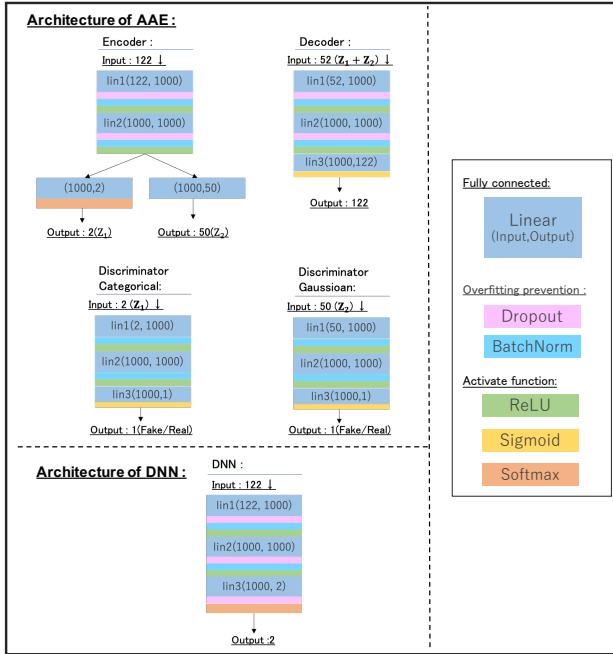


Fig. 6. The detail of AAE and deep neural network that we used

In this study, we used k -fold cross-validation with $k = 5$. k -fold cross-validation is a technique that divides data into k pieces, trains $k - 1$ pieces as training data, and do performance evaluation using remain data as validation data in order to adjust parameters. The performance measure reported by k -fold cross-validation is then the average of the values computed in the loop. This way, we can prevent that knowledge about the test dataset leak into the training model, and it is possible to select useful parameters. Therefore, we can perform an evaluation close to the real environment. This is especially important in this study because the attacks

included in KDDTrain+ and KDDTest+ are different. In other words, the data distribution is different. For example, by using the information of KDDTest + and setting parameters, it is possible to create a classifier suitable for KDDTest + and achieve high accuracy. However, this is not the desirable experimental method because it has used the information on unknown attacks that cannot be obtained in training phase in a practical environment. For this reason, KDDTest+ should be used only for final evaluation. Therefore, we use k -fold cross-validation. In this study, we divided KDDTrain+ into five pieces. The training and validation were performed 5 times, and the test with KDDTest+ was performed using the model with the highest accuracy of validation in each fold, and the average value of the classification results was the final Test Accuracy.

The hyper-parameter of AAE and DNN is described in Tables I.

TABLE I
HYPERPARAMETER VALUES USED FOR AAE AND DNN TRAINING

Hyperparameter	Value
Adversarial Autoencoder(AAE)	
learning rate	1.0e-7
batch size	128
the epoch (90%,10%)	30
the epoch (1%)	300
the epoch (0.1%,0.01%)	3000
dimensionality of the latent variable	$z_1:2, z_2:50$
Deep neural networks(DNN)	
learning rate	1.0e-5
batch size	128
the epoch (90%,10%,1%,0.1%,0.01%)	300

D. Computing environment

The computing environment used in the experiment is shown in Table II.

TABLE II
COMPUTING ENVIRONMENT

Hardware	
OS	Ubuntu 16.04
CPU	Intel Pentium@3.50GHz
GPU	NVIDIA GeForce GTX1060(6GB)
RAM	8GB
Software	
Programming Language	Python(3.6)
Deep Learning Framework	Pytorch(version: 0.4.1)
CUDA	version: 9.0

VI. EXPERIMENTAL RESULTS

A. Accuracy comparison

The AAE was learned by semi-supervised learning, and 2-class classification (normal and anomaly) was conducted. Table III shows the comparison results of accuracy between

AAE by semi-supervised learning which is the proposed method and deep neural network(DNN) which is one of the supervised learning methods.

The purpose of this section is verifying the effectiveness of support with unlabeled data by comparing two methods of semi-supervised learning and supervised learning. The results of Accuracy comparison can be seen in Table III.

TABLE III
RESULTS OF ACCURACY COMPARISON(AAEDNN)

Labeling Rate	90%	10%	1%	0.10%	0.01%
Labeled Data	113,375	12,597	1,259	125	11
Unlabeled Data	12,598	113,376	124,714	125,848	125,962
Adversarial Autoencoder	83.20	83.11	82.78	81.37	53.66
Deep neural network	81.07	77.33	75.88	76.87	71.55

When the labeling rate was 0.1% or more, the accuracies of AAE that is better than those of DNN. On the other hand, when the labeling rate is extremely low, such as 0.01 %, the accuracy is significantly low, and the AAE is inferior to DNN. From the above results, we argue that the proposed method based on AAE is effective since it detects high accuracy except in the environment where only extremely few labeled data is available.

B. Misdetction comparison

Next, we focus on the false detection rate and consider each sample is classified which target. A high FN is that anomaly communication is regarded as normal communication. In other words, there are many missed attack. The experiment was performed using 1.0% labeled data with AAE, same as the environment of Accuracy comparison. The result of the misdetction comparison that AAE and DNN is shown in Tables IV. Samples predicted in the number of actual samples.

TABLE IV
MISDETECTION RESULTS OF AAE AND DNN

AAE	DNN	Predicted Class			
Actual Class		Normal		Anomaly	
		Normal	Anomaly	Normal	Anomaly
		86.5%	92.2%	13.5%	7.8%
Anomaly		20.0%	36.5%	80.0%	63.5%

The proposed AAE method reduced the FNR from 20.0% to 36.5% compared to DNN, and succeeded in realizing a better model. However, we observe that the FNR (20.0 %) is higher than the FPR (13.5 %) even the results of AAE that could suppress FN. As mentioned earlier, one of the reasons why machine learning is still difficult to be used in a practical environment is high misdetection rate. In particular, high FNR is fatal. We think it is one of the topic of discussion that should be improved in future research. A model with FNR lower than FPR is desirable since there exists tradeoff between FN and FP.

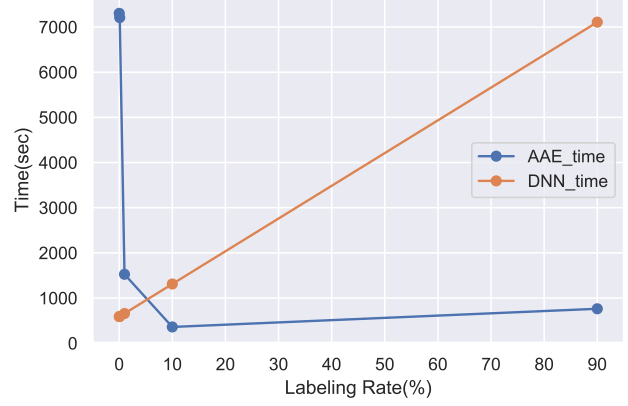


Fig. 7. Computing time required to train AAE and DNN.

C. Comparison with previous researches

Table compares the accuracy of the proposed method. We used the data published in the past literatures that evaluates various machine learning algorithms. XGboost with K-Means, Bagging(Base classifier -J48), recurrent neural networks, support vector machine, NB Tree, random forest, J48, multi-layer perceptron. The proposed method yields the comparable accuracy with the previous methods. It uses small number of labeled data, that is 10 % or 1.0 % of the entire data. We argue that the proposed semi-supervised machine learning method avoids fairly amount of costly human-labor tasks while it achieves the comparable performance. However, the accuracy results were slightly inferior to ensemble methods such as XGboost with K-Means and Bagging(Base classifier -J48).

TABLE V
RESULT OF COMPARISON WITH PREVIOUS RESEARCHES

Method	Accuracy	Labeled	Unlabeled
XGboost with K-Means [17]	84.25	125,973	0
Bagging(Base classifier -J48) [11]	84.25	125,973	0
RNN [19]	83.28	125,973	0
AAE (10%labeled)	83.11	12,597	113,376
AAE (1%labeled)	82.78	1,259	124,714
Support Vector Machine [10]	82.37	125,973	0
NBTree [16]	82.02	125,973	0
Random Tree [16]	81.59	125,973	0
J48 [16]	81.05	125,973	0
Random Forest [16]	80.68	125,973	0
Multilayer Perceptron [16]	77.41	125,973	0

VII. COMPUTING TIME

We present the computation time required to train the AAE and the DNN. Fig. 7 shows the time required to train the AAE and DNN for various ratio of labeled data. We used the parameters of the number of epochs and the learning rate as show in Table. I. If AAE is used as supervised learning, only the AE parameters need to be trained. As such when the ratio of labeled data is higher, it requires the shorter computing time

to train the AAE. When the ratio of labeled data is high, it requires less number of epochs to train the AAE. On the other hand, when the AAE uses unlabeled data, the training of two discriminators is needed in addition to the training of AE. As such when the ratio of unlabeled data is higher, it requires the longer computing time to train the AAE. When the ratio of unlabeled data is higher, it requires more epochs to train the AAE. As shown in Fig. 7, the ratio of labeled data is less than 5.0%, the AAE requires the longer training time than the DNN. This is because the DNN trains one neural network whereas the AAE trains one neural networks for the AE and two neural networks for Discriminators.

VIII. CLOSING REMARKS

We presented IDS using the AAE as semi-supervised learning, demonstrated that it achieves high accuracy compared to DNN in most cases. In addition, our scheme uses a minimum number of labeled data compared to other state-of-the-art approaches. We speculate that this is because assist the data structure with labeled data after getting the data structure from unlabeled data. As a result, it is possible to realize higher accuracy than supervised learning that uses only the same number of labeled data. We confirmed that unlabeled data could be used effectively by using AAE.

The following three dimensions requires future works. Firstly, the recent work applies semi-supervised learning for the IDS using Variational Auto-Encoder (VAE) [9]. VAE assumes that the underlying distribution for the latent variables is a Gaussian while the AAE assumes that an arbitrary distribution is used the underlying distribution for the latent variables. Even though we used a mixture of Gaussian and categorical distribution in this paper, study on applying other types of distribution is left for future work, which we will work for the next step. We aim to realize the higher detection rate and lower misdetection. Optimization of the distribution imposed on the latent variable as well as the dimensionality of the latent variable, which is an important parameter in using AAE need to be investigated. Secondly, we succeeded in reducing FNR compared with DNN, but it is not sufficient when we take practical use into account. It is necessary to capture the features of each attack and improve our method by focusing on each attack. Thirdly, we used NSL-KDD dataset. Although it is still used as a reference in recent study, it is old and not a perfect representative of existing real networks. We plant to use more recent dataset [1] to consider the latest attacks.

REFERENCES

- [1] Canadian Institute for Cybersecurity. <https://www.unb.ca/cic/datasets/index.html>.
- [2] Pytorch. <https://github.com/pytorch/pytorch>.
- [3] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. *ArXiv e-prints*, June 2014.
- [4] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.
- [5] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [6] Richard Lippmann, Robert K Cunningham, David J Fried, Isaac Graf, Kris R Kendall, Seth E Webster, and Marc A Zissman. Results of the darpa 1998 offline intrusion detection evaluation. In *Recent advances in intrusion detection*, volume 99, pages 829–835, 1999.
- [7] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. Adversarial Autoencoders. *ArXiv e-prints*, November 2015.
- [8] Saurabh Mukherjee and Neelam Sharma. Intrusion detection using naive bayes classifier with feature reduction. *Procedia Technology*, 4:119–128, 2012.
- [9] Genki Osada, Kazumasa Omote, and Takashi Nishide. Network intrusion detection based on semi-supervised variational auto-encoder. In Simon N. Foley, Dieter Gollmann, and Einar Snekkenes, editors, *Computer Security – ESORICS 2017*, pages 344–361, Cham, 2017. Springer International Publishing.
- [10] M. S. Pervez and D. M. Farid. 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)*, pages 1–6, Dec 2014.
- [11] Ngoc Tu Pham, Ernest Foo, Suriadi Suriadi, Helen Jeffrey, and Hassan Fareed M Lahza. Improving performance of intrusion detection system using ensemble methods and feature selection. In *Proceedings of the Australasian Computer Science Week Multiconference, ACSW '18*, pages 2:1–2:6, New York, NY, USA, 2018. ACM.
- [12] Ali Shiravi, Hadi Shiravi, Mahbod Tavallae, and Ali A Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *computers & security*, 31(3):357–374, 2012.
- [13] R. Sommer and V. Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE Symposium on Security and Privacy*, pages 305–316, May 2010.
- [14] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho. Deep recurrent neural network for intrusion detection in sdn-based networks. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pages 202–206, June 2018.
- [15] Tuan A. Tang, Lotfi Mhamdi, Des McLernon, Syed Ali Raza Zaidi, and Mounir Ghogho. Deep learning approach for Network Intrusion Detection in Software Defined Networking. *Proceedings - 2016 International Conference on Wireless Networks and Mobile Communications, WINCOM 2016: Green Communications and Networking*, pages 258–263, 2016.
- [16] M. Tavallae, E. Bagheri, W. Lu, and A. A. Ghorbani. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pages 1–6, July 2009.
- [17] P. Verma, S. Anwar, S. Khan, and S. B. Mane. Network intrusion detection using clustering and gradient boosting. In *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–7, July 2018.
- [18] Wei Wang, Yiqiang Sheng, Jinlin Wang, Xuwen Zeng, Xiaozhou Ye, Yongzhong Huang, and Ming Zhu. Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. *IEEE Access*, 6:1792–1806, 2018.
- [19] C. Yin, Y. Zhu, J. Fei, and X. He. A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access*, 5:21954–21961, 2017.