Empirical analysis of the fine-tuning for Unsupervised Anomaly Detection in the ICT system

Yoichi Matsuo
NTT Network Service Systems Laboratories, NTT Corporation,
3-9-11 Midori-cho, Musashino-shi, Tokyo 180-8585 Japan
Email:yoichi.matsuo@ntt.com

***Abstract*—Many unsupervised anomaly detection (UAD) methods for ICT systems that use only normal data collected from routers or servers, such as traffic volumes and text logs, have been developed to detect anomalies. Since the normal states of the ICT systems change due to the addition or deletion of devices, configuration changes, and OS updates, system operators need to collect enough normal data to learn various normal states whenever there is a change. Therefore, UAD methods cannot be applied until new data is gathered and a new normal state is identified. Using fine-tuning, one of the transfer learning techniques might reduce the duration of collecting normal state data. However, most of the existing papers focus on transferring knowledge of supervised models on image classification tasks, which are completely different from UAD in ICT systems. This paper analyzes a fine-tuning architecture to improve UAD in an ICT system with a small amount of normal data. By preparing five datasets, comprehensive experiments were conducted, and it was found that the fine-tuning architecture has the possibility to improve the accuracy of anomaly detection with a small amount of data for the ICT system dataset and scenarios.**

***Index Terms*—Unsupervised anomaly detection, Fine-tuning, Deep learning, ICT system**

I. Introduction

Anomaly in an ICT system, such as communication networks or application systems needs to be detected immediately by its providers since stopping the ICT system causes a significant impact on peoples' daily lives [1]. To detect the anomaly in the ICT system, many Unsupervised Anomaly Detection (UAD) methods [2]–[4] that use only normal data such as traffic volumes, Central Processing Unit (CPU) usage, memory usage, and text logs from routers or servers, have been developed since the anomalies rarely occur, and even when they occur, system operators will take action to prevent recurring anomalies. This means that label data of anomalies cannot be collected.

Because the normal state of the ICT system changes with the addition or deletion of devices, configuration changes, and OS updates, it is necessary to re-collect normal data and train an UAD method for each change. Therefore, if the duration of re-collecting data in the normal state of the ICT system increases, UAD methods cannot be used to monitor the system during data collection.

Transfer leading such as fine-tuning or Domain Adaptation (DA) technique can reduce the data collection time for UAD after an ICT system change (i.e., target domain) by using data from before the system change (i.e., source domain), which leads to increasing the available time for anomaly detection. However, the existing DA methods cannot be applied to anomaly detection methods in ICT systems. This is because, the problem setting of DA is that the data in the source domain has a label, while anomaly labels cannot be collected in the ICT system. The fine-tuning technique can be applied to UAD in the ICT system because it can be applied to unsupervised learning or when the number of dimensions differs between domains. However, most of existing papers about fine-tuning technique focus on computer vision research or natural language tasks. Therefore, it is unclear whether fine-tuning technique can reduce the data collection time and improve UAD in the ICT system with small amount of data in normal state.

In this paper, we analyze the performance of the fine-tuning technique for UAD in an ICT system with a small amount of normal data. Especially, we focus on the availability of fine-tuning when the operators need to collect the normal state data due to the addition or deletion of devices, configuration changes, and OS updates, which has not been considered in previous papers. First, we develop the fine-tuning architecture for UAD methods in the ICT system. Then, we evaluated the anomaly detection accuracy of each UAD method for the scenarios requiring fine-tuning using ICT system datasets. The datasets are prepared by using public datasets and developing web service system. Finally, a detailed investigation of the fine-tuning for UAD is provided. The UAD with fine-tuning technique is evaluated using two types of ICT system data which are network security data and web service system. Compared with the baseline methods, experimental results show that fine-tuning can improve the Area Under the ROC Curve (AUC) of UAD in the target domain.

Contributions of this paper are as follows.

- This paper focus on the applicability of the fine-tuning for UAD which is needed to increase the available time of UAD in the ICT system.
- The comprehensive experiments are conducted to evaluate the performance of the fine-tuning for UAD using tow types of ICT system datasets. The results show that the fine-tuning for UAD can improve AUC in the ICT system for a specific scenario. In terms of reducing the data collection time with the proposed architecture, the collection time was at most 1/20 for network security datasets.

978-3-903176-59-1

II. Related Work

A. Anomaly Detection for ICT systems

Various anomaly detection methods have been investigated for managing the ICT systems [2], [4], [8]–[11] using metrics such as CPU usage, memory usage, traffic volume, and/or system logs. USAD [8] combines generative adversarial network (GAN) and AutoEncoder (AE) to generate fake data which is close to normal data for reducing the false anomaly detection when the anomaly data is similar to the normal data. For monitoring application systems with a cloud computing platform, Diamanti et al. [2] collect metrics on host servers and in the cloud computing platform and extract hidden relationships between anomalies and metrics using long short-term memory (LSTM) based AE. LogAnomaly [10] is a framework to model an unstructured log stream using a word2vec approach based on natural language processing and simultaneously detect sequential and quantitative log anomalies. In Zhang et al. [4], the method which extracts temporal dependency and relationship among multivariable is proposed by combining convolutional neural networks (CNN) and LSTM to detect anomalies and diagnose and present the degree of anomalies. In Zhao [11], a graph neural network is used to extract temporal dependency and relationships among multivariable.

However, these methods require a massive number of normal data to be trained in the normal state of their systems.

B. Domain Adaptation and Fine-tuning

Papers using Domain adaption for application classification, network performance prediction, and anomaly detection are published [12]–[17]. Authors [12], [13] proposed DA methods that treat network traffic in the real world with no labels as the target domain and network traffic in open-dataset with labels as the source domain since collecting the label which indicates applications type is complex in real-world due to the privacy, encryptions of the traffic. The proposed method in paper [14] predicts network performance such as RTT by extending the method [6] using the labels in the source and target domains. Since this paper assumes using DA based method when the network system dynamics are changed due to the re-configurations, the dimension of used data in both domains is the same. Although papers [15], [16] about anomaly detection of traffic data do not use the DA based methods, authors propose techniques to accommodate differences in data distribution between domains. Anomaly detection method using DA for network logs is also proposed [17] by embedding the log data with transformer [18], in which the anomaly log data in the source domain are used.

Although DA based methods for tasks in ICT systems' operations are proposed, these methods assume that data in the source domain data have labels and/or dimensions of data in both domains are the same. However, for anomaly detection in the ICT system, collecting labels is hard, and the dimension of data in the target domain differs from that in the source domain.

Anomaly detection methods using fine-tuning are also proposed in various research fields [19]–[23]. Papers [19], [20] propose a feature selection method for anomaly detection in images and the survey of fine-tuning for video surveillance, respectively. H. Liu et al. [21] proposes an anomaly detection method for transportation systems using sensing data. In this paper, although there is enough data, fine-tuning is used to reduce the training time of Deep Neural Network (DNN). The anomaly detection in ICT systems is also proposed in papers [22], [23]. In paper [22], fine-tuning is used to adapt the pre-trained large language model to log text for detecting anomalies in system logs. M. Sun et al. [23] pre-trains a UAD model for embedding the high dimensional monitoring data in the ICT system into the latent space. Then, embedded data are clustered to reduce the dimensions of monitoring data. Finally, the pre-trained model is fine-tuned for anomaly detection in clustered monitoring data.

III. Fine-tuning Architecture for UAD

We describe the fine-tuning architecture and implemented DNN based UAD methods for our performance analysis. This analysis aims to clarify whether the fine-tuning technique can reduce the data collection time and improve UAD in the ICT system with a small amount of data in normal state. Therefore, this section first explains the implemented DNN based and other UAD methods. Then, we develop the fine-tuning architecture for UAD methods.

A. UAD Methods

To analyze the fine-tuning performance for UAD methods, DNN based anomaly detection methods are implemented. We explain each method briefly in this section.

Let $\boldsymbol{x}_i$ be a l-dimensional vector whose elements are metrics at time i, where l is the number of kinds of metrics collected from the ICT system and y_t is a label of the ICT system at time i, which takes 1 if the data is normal and -1 otherwise. Let $\mathcal{D}_{\text{train}}$ be a set of tuples, $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$, which is training dataset and N is the number of training data. Similarly, let $\mathcal{D}_{\text{test}}$ be a set of tuples, $\{(\acute{\boldsymbol{x}}_i \acute{y}_i)\}_{i=1}^{\acute{N}}\}$ which are test dataset and $\acute{N}$ is the number of test data. Here $\acute{}$ denotes data in the test dataset, y_i in the training dataset $\mathcal{D}_{\text{train}}$ always takes 1, and $\acute{y}_i$ in the test dataset $\mathcal{D}_{\text{test}}$ takes 1 or -1 depending on the application system state. In the training phase, weights of UAD methods are trained using a training dataset $\mathcal{D}_{\text{train}}$, and in the test phase, the anomaly score of $\acute{x}_i$ is calculated to estimate whether $\acute{y}_i$ is a normal or abnormal state.

1) AE: AutoEncoder (AE) is an unsupervised encoder-decoder-based method in which parameters are trained by a normal state dataset so that input data and output data by AE are the same. In AE, for the encoder, weight parameters W_{AE}^{j} in the j-th layer are trained by the following equation.

$$z^j = \sigma(W_{\text{AE}}^{j} z^{j-1} + b^j), \tag{1}$$

where z^0 is x_t, and σ is activation function, and b^j are bias functions. For the decoder, weight parameters $\tilde{W}_{\text{AE}}^{j}$ in the j-th layer are trained by the following equation.

$$\tilde{z}^j = \sigma(\tilde{W}_{\text{AE}}^{j} \tilde{z}^{j-1} + \tilde{b}^j), \tag{2}$$

where $\tilde{z}^0$ is z^J, and J is the number of AE layers. Then, AE is trained so that the reconstructed vector $\hat{x}_t$ equals x_t as follows.

$$L(x_1, \ldots, x_N) = \min_{W,\tilde{W}} \sum_{i=1}^{N} \|x_i - \hat{x}_i\|_2. \tag{3}$$

where $\hat{x}_i$ is calculated by encoding x_i and decoding z^J. To calculate the anomaly score for test data $\acute{x}_{\acute{i}}$, the difference between $\acute{x}_{\acute{i}}$ and predicted $\hat{x}_{\acute{i}}$ is calculated by using Equation (3). More details can be found in Sakurada and Yairi. [24].

2) VAE: Variational AutoEncoder (VAE) [25] is a method to model the probability distribution $P(X|Z)$ using the latent variable Z by estimating the posterior distribution $P(Z|X)$, where X is a random variable and $x_i, \acute{x}_{\acute{i}}$ is the observation of X. In the VAE, the encoder $q_\varphi(Z|X)$ and the decoder $p_\theta(X|Z)$ which are DNN and the φ and θ are the trained parameters, are trained to estimate the probability distribution $P(Z|X)$ and $P(X|Z)$, respectively. This training process is done by minimizing the reconstruction and regularization errors so that the encoded data through $q_\varphi(z|x)$ obey the normal distribution as follows.

$$\begin{aligned} L(x_1, \ldots, x_N) = &\min_{\varphi,\theta} \sum_{i}^{N} \|x_i - \hat{x}_i\|_2 \\ &+ KL[q_\varphi(Z|X)||p_\theta(X)], \end{aligned} \tag{4}$$

where, KL is a Kullback–Leibler divergence [26]. To calculate the anomaly score for test data $\acute{x}_{\acute{i}}$, the difference between $\acute{x}_{\acute{i}}$ and predicted $\hat{x}_{\acute{i}}$ is calculated by using Equation (4). More details can be found in Kingma and Welling [25].

3) DeepSVDD: Deep Support Vector Data Description (DeepSVDD) [27], which is a one-class classification using DNN. DeepSVDD is trained using normal state data so that these normal data are embedded into the hypersphere with center c as follows.

$$\begin{aligned} L(x_1, \ldots, x_N) = &\min_{W} \frac{1}{N} \sum_{i=1}^{N} ||\phi(\boldsymbol{x}_i, W) - c||^2 \\ &+ \frac{\lambda}{2} \sum_{l=1}^{L} ||W^l||_F^2, \end{aligned} \tag{5}$$

where the ϕ is the DNN and W is the trained weight parameters of ϕ. The anomaly score of test data $\tilde{x}_{\tilde{i}}$ is calculated as the distance between the hypersphere center c and embed test data $\tilde{x}_{\tilde{i}}$ using Equation 5. More details can be found in L. Ruff et al. [27].

Additionally, Local Outlier Factor (LOF) and Isolation Forest (IF) are implemented for the comparison. LOF is a density-based method in which the anomaly score of a certain data point is calculated using the average distance of the k-nearest neighbor. IF is a machine learning method to find the outlier in the dataset by constructing the trees. More details of LOF and IF can be found in Breunig et al. [28] and F. T. Liu et al. [29], respectively.

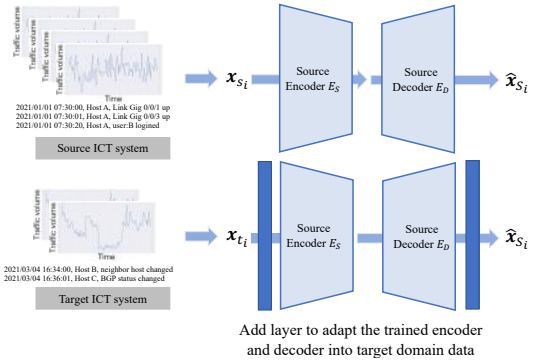

Fig. 1. Fine-tuning for AE and VAE

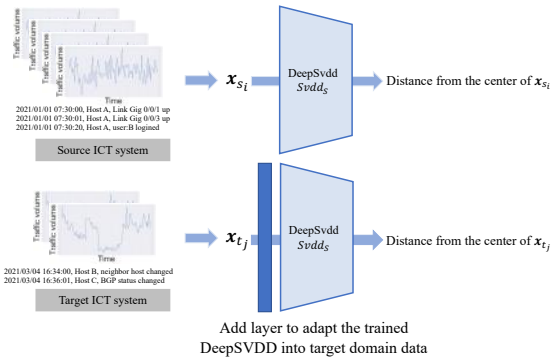

Fig. 2. Fine-tuning for DeepSVDD

B. Fine-tuning for UAD

The motivation of this paper is to know if it is possible to transfer the trained model from the source domain, in which there are enough training data, to the target domain, in which there are small amounts of data, by using fine-tuning. These situations occur when the data dimension changes due to the addition or deletion of the device or the characteristic of the normal state change due to an update of configurations, which requires the collection of the data in normal state again.

To enable the fine-tuning for UAD, we propose to add the layer into DNN based methods described above to adapt the data dimension changes between source and target domains. In the following, we describe the problem setting of the analysis and the fine-tuning architecture.

Let $\mathcal{D}^S_{\text{train}} = \{(\boldsymbol{x}^S_i, y^S_i)\}_{i=1}^{N^S}$ and $\mathcal{D}^T_{\text{train}} = \{(\boldsymbol{x}^T_i, y^T_i)\}_{i=1}^{N^T}$ be the training dataset in source domain S and target domain T. Here, we assume N^S is much larger than N^T. Using these datasets, first, AE, VAE, and DeepSVDD models are trained using enough amount of data $\mathcal{D}^S_{\text{train}}$. Then, to adapt the trained model in the source domain to the data characteristics and data dimension in the target domain, we add another layer to the first layer of the encoder and the last layer of the decoder in AE and VAE. The illustration of the fine-tuning architecture for AE and VAE is shown in Figure 1. Similar to AE and VAE, another layer is added into the first layer of the trained DeepSVDD model as in Figure 2. This added layer is trained using the data in the target domain to adapt the data

TABLE I
DATASETS SUMMARY

Name	D	N_{train}	N_{test}	R_A
NSL-KDD	114	67343	22544	0.57
CICIDS	71	50000	50000	0.40
Web system 1	638	119	116	0.32
Web system 2	659	119	118	0.31
Web system 3	660	119	117	0.31

distribution and data dimension in the target domain.

IV. EVALUATIONS

In this section, first, we describe dataset information and the preprocessing procedure for experiments. Then, the experimental setting is explained. Finally, we evaluate the fine-tuning performance for UAD by comparing it with the baseline method in which only the target domain data are used for the training.

A. Datasets and Scenarios

For our analysis, we prepared two data types in the ICT system, i.e., network security data and web service system. The summary of datasets is described in Table II. N_{train} and N_{test} represents the number of the samples in the dataset used for N^S or N^T, and the number of the dataset used for N^T_{test}, respectively. D represents the number of dimensions of the datasets. Each data point in the datasets is preprocessed as described in the following subsections and normalized by min-max normalization. R_A represents the ratio of anomaly data in the test data.

1) Network Security Datasets: We prepared two network intrusion datasets, NSL-KDD [30] and CIC (Canadian Institute of Cybersecurity) IDS 2017 [31]. All datasets contain information about packed size, source IP addresses, destination IP addresses, port numbers, and other information. However, the number of kinds of collected data and the duration of the collection differ. Each data at each time is preprocessed to the vector. Then, the vectors are input to UAD methods to train or evaluate UAD methods using normal or abnormal labels.

2) Web system Datasets: We built two web systems with several routers and servers to collect normal and abnormal data for the evaluations. The details of the web system topology and collected datasets in our experiments are shown in Figures. 3 and 4, and Table II, respectively. In these web systems, we collect the data in normal state for two hours to train UAD methods as an offline process. Then, test data are collected for two hours as an online process. During the online process, anomalies such as high CPU usage, high memory usage, or packet drop are injected once every five minutes at each router or server.

The Web system 1 dataset contains the data collected from tree topology in Figure 3, and the Web system 2 and 3 datasets contain the data collected from full mesh topology in Figure 4. In the experiments, we conduct all patterns of source and target datasets using Web systems 1, 2, and 3, and each of the experiments represents a case where fine-tuning for UAD is required in the operation of an web system. For instance, the experiment with the Web system 1 dataset used as source dataset and Web system 2 or 3 datasets used as target dataset represent the scenarios when the device is added to the web system and configurations are updated or when the new full mesh web system is built. The case with the Web system 2 dataset as the source dataset and the Web system 3 dataset as the target dataset represents when the data characteristics are changed due to configuration updates.

Since it is possible to measure the performance of fine-tuning in scenarios where fine-tuning is required in system operation using the prepared dataset, we can clear the availability of fine-tuning.

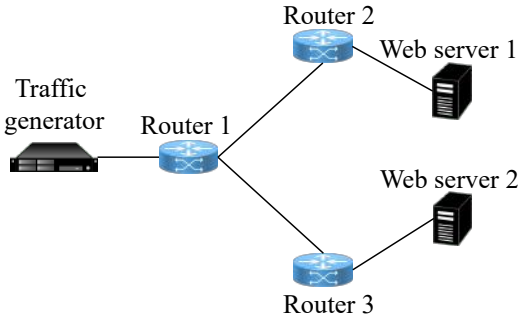

Fig. 3. Web system topology 1 in experiments

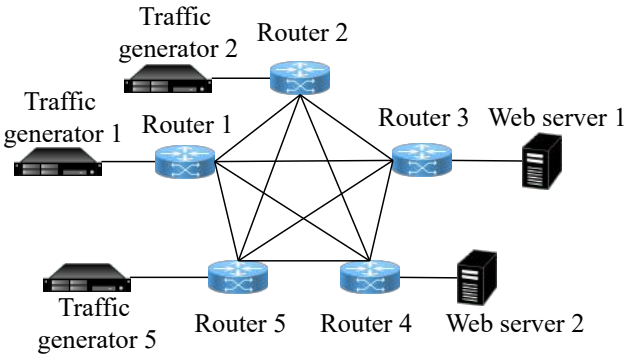

Fig. 4. Web system topology 2 in experiments

TABLE II
WEB SYSTEM DATASET INFORMATION

Experimental environment	Condition
Topology (# Equipment)	Tree (6), Full mesh (10)
Collected data	CPU usage, memory usage, syslog, traffic data
Injected anomalies	high CPU usage, high memory usage, or packet drops
Degree of injections	low (20%), middle (60%), or high (80%)
Anomaly frequency	Once every five minutes
Duration	Training data: two hours Test data: four hours

TABLE III
AUC AND STANDARD DEVIATION OF UAD METHODS WITH SOURCE DATASET NSL-KDD AND TARGET DATASET CICIDS

Method	Target data ratio					
	1%	5%	10%	25%	50%	100%
LoF	0.496 (0.023)	0.521 (0.040)	0.481 (0.014)	0.508 (0.028)	0.530 (0.006)	0.678 (0.000)
IF	0.678 (0.000)	0.678 (0.000)	0.678 (0.000)	0.678 (0.000)	0.678 (0.000)	0.678 (0.000)
AE	1.000 (0.000)	0.995 (0.006)	0.994 (0.007)	0.994 (0.011)	0.997 (0.005)	0.998 (0.003)
AE-FT	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)
VAE	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)
VAE-FT	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)
DeepSVDD	0.606 (0.299)	0.566 (0.371)	0.434 (0.494)	0.778 (0.441)	0.445 (0.527)	0.751 (0.405)
DeepSVDD-FT	0.630 (0.358)	0.267 (0.275)	0.363 (0.127)	0.401 (0.098)	0.395 (0.088)	0.319 (0.025)

TABLE IV
AUC AND STANDARD DEVIATION OF UAD METHODS WITH SOURCE DATASET CICIDS AND TARGET DATASET NSL-KDD

Method	Target data ratio					
	1%	5%	10%	25%	50%	100%
LoF	0.849 (0.007)	0.785 (0.016)	0.757 (0.010)	0.778 (0.006)	0.754 (0.025)	0.740 (0.000)
IF	0.678 (0.000)	0.678 (0.000)	0.678 (0.000)	0.678 (0.000)	0.678 (0.000)	0.678 (0.000)
AE	0.550 (0.122)	0.577 (0.100)	0.604 (0.090)	0.885 (0.055)	0.947 (0.013)	0.948 (0.003)
AE-FT	0.912 (0.041)	0.953 (0.018)	0.948 (0.012)	0.947 (0.041)	0.950 (0.015)	0.954 (0.016)
VAE	0.773 (0.054)	nan (nan)	0.942 (0.001)	0.964 (0.003)	0.964 (0.000)	nan (nan)
VAE-FT	0.120 (0.022)	0.122 (0.019)	0.128 (0.025)	0.115 (0.000)	0.128 (0.025)	0.122 (0.019)
DeepSVDD	0.934 (0.021)	0.937 (0.011)	0.946 (0.014)	0.929 (0.017)	0.942 (0.019)	0.936 (0.025)
DeepSVDD-FT	0.500 (0.212)	0.453 (0.229)	0.412 (0.223)	0.393 (0.240)	0.470 (0.230)	0.453 (0.226)

B. Experimental Setting

Hyperparameters of UAD methods are described here. In AE, VAE, and DeepSVDD, a full connected neural network is used for each layer, and the number of layers is determined using grid search with a search range [2,3,4,5]. Rectified linear unit (ReLU) function and batch normalization are used as the function between layers. The number of dimensions of the neural networks was also set using grid search with range [20%, 30%, ..., 70%] of the dimension of the input vector in each domain, respectively. As learning rate and weight decay, we prepared from 0.01 to 0.0001 and 0.001 to 0.000001, respectively. Then, a grid search is conducted and hyperparameters of each UAD are set to achieve the highest AUC. Every model was optimized with Adam [32], and the epoch is set to 500.

The number of neighboring points for LOF was set to 1, and The number of base estimators in the ensemble in IF is set to 100, since it recorded the highest AUC in the grid search.

To investigate the characteristics of the fine-tuning in terms of the number of data, we select the data from target dataset with following ratio to 1%, 5%, 10%, 25%, 50%, and 100% for network security dataset, and 10%, 25%, 50%, and 100% for web system dataset, which represents the number of used target data. This is because only 1% of the data for the web system dataset is too few. All of the train data in the source domain is used for UAD methods. We randomly select the target data three times with set ratio and for each of sampled data experiment was executed three times. As an evaluation metric of anomaly detection methods, AUC is selected and the average and variance were calculated.

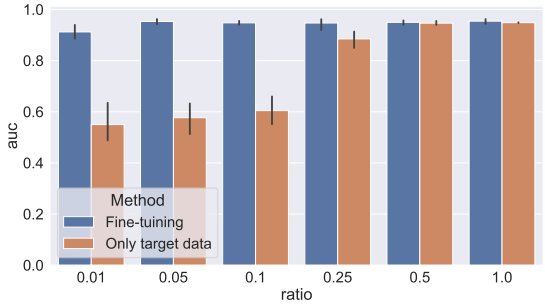

Fig. 5. Results of Fine-tuned AE with S:CICIDS and T:NSL-KDD

C. Experimental Results

The performance of the fine-tuning was evaluated using prepared datasets. In Tables III and IV, AUC and standard deviation of implemented UAD methods with/without fine-tuning are illustrated, where -FT represents the UAD methods with fine-tuning using source dataset and other methods only use the target dataset and the number in the () is standard deviation.

1) Results of Network Security Datasets: In Table III, each DNN based method achieved high AUC with only a small amount of data such as 1% or 5%. Since the ports used for normal and abnormal data in the CICIDS dataset are so different, the trained model, with a small amount of data, can easily classify the test data. Regarding the performance of the fine-tuning, AUC of the UAD methods does not deteriorate, and the AUC and standard deviation for AE increases.

TABLE V
AUC AND STANDARD DEVIATION OF UAD METHODS FOR WEB SYSTEM 1 DATASET

Method	Source data: Web system 2, Target data: Web system 1, ratio				Soruce data: Web system 3, Target data: Web system 1, ratio			
	10%	25%	50%	100%	10%	25%	50%	100%
LoF	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)
IF	0.678 (0.000)	0.678 (0.000)	0.678 (0.000)	0.678 (0.000)	0.678 (0.000)	0.678 (0.000)	0.678 (0.000)	0.678 (0.000)
AE	0.595 (0.065)	0.587 (0.093)	0.599 (0.064)	0.522 (0.065)	0.595 (0.065)	0.587 (0.093)	0.599 (0.064)	0.522 (0.065)
AE-FT	0.520 (0.042)	0.484 (0.044)	0.478 (0.018)	0.476 (0.042)	0.524 (0.048)	0.556 (0.097)	0.518 (0.072)	0.466 (0.051)
VAE	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)
VAE-FT	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)
DeepSVDD	0.485 (0.053)	0.501 (0.007)	0.490 (0.026)	0.491 (0.024)	0.485 (0.053)	0.501 (0.007)	0.490 (0.026)	0.491 (0.024)
DeepSVDD-FT	0.513 (0.086)	0.461 (0.068)	0.559 (0.060)	0.538 (0.085)	0.426 (0.074)	0.503 (0.030)	0.486 (0.028)	0.493 (0.019)

TABLE VI
AUC AND STANDARD DEVIATION OF UAD METHODS FOR WEB SYSTEM 2 DATASET

Method	Soruce data: Web system 1, Target data: Web system 2, ratio				Soruce data: Web system 3, Target data: Web system 2, ratio			
	10%	25%	50%	100%	10%	25%	50%	100%
LoF	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)
IF	0.678 (0.000)	0.678 (0.000)	0.678 (0.000)	0.678 (0.000)	0.678 (0.000)	0.678 (0.000)	0.678 (0.000)	0.678 (0.000)
AE	0.502 (0.008)	0.501 (0.007)	0.495 (0.017)	0.497 (0.009)	0.502 (0.008)	0.501 (0.007)	0.495 (0.017)	0.497 (0.009)
AE-FT	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.498 (0.018)	0.477 (0.055)	0.514 (0.064)	0.483 (0.029)
VAE	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)
VAE-FT	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)	0.494 (0.000)
DeepSVDD	0.503 (0.046)	0.493 (0.010)	0.497 (0.005)	0.470 (0.046)	0.503 (0.046)	0.493 (0.010)	0.497 (0.005)	0.470 (0.046)
DeepSVDD-FT	0.469 (0.097)	0.457 (0.086)	0.492 (0.037)	0.471 (0.029)	0.470 (0.038)	0.516 (0.088)	0.515 (0.046)	0.486 (0.022)

TABLE VII
AUC AND STANDARD DEVIATION OF UAD METHODS FOR WEB SYSTEM 3 DATASET

Method	Soruce data: Web system 1, Target data: Web system 3, ratio				Soruce data: Web system 2, Target data: Web system 3, ratio			
	10%	25%	50%	100%	10%	25%	50%	100%
LoF	0.479 (0.000)	0.479 (0.000)	0.479 (0.000)	0.479 (0.000)	0.479 (0.000)	0.479 (0.000)	0.479 (0.000)	0.479 (0.000)
IF	0.678 (0.000)	0.678 (0.000)	0.678 (0.000)	0.678 (0.000)	0.678 (0.000)	0.678 (0.000)	0.678 (0.000)	0.678 (0.000)
AE	0.471 (0.066)	0.448 (0.047)	0.485 (0.033)	0.456 (0.051)	0.471 (0.066)	0.448 (0.047)	0.485 (0.033)	0.456 (0.051)
AE-FT	0.479 (0.000)	0.479 (0.000)	0.479 (0.000)	0.479 (0.000)	0.486 (0.017)	0.483 (0.024)	0.474 (0.028)	0.460 (0.034)
VAE	0.479 (0.000)	0.479 (0.000)	0.479 (0.000)	0.479 (0.000)	0.479 (0.000)	0.479 (0.000)	0.479 (0.000)	0.479 (0.000)
VAE-FT	0.479 (0.000)	0.479 (0.000)	0.479 (0.000)	0.479 (0.000)	0.479 (0.000)	0.479 (0.000)	0.479 (0.000)	0.479 (0.000)
DeepSVDD	0.469 (0.018)	0.459 (0.058)	0.471 (0.014)	0.463 (0.019)	0.469 (0.018)	0.459 (0.058)	0.471 (0.014)	0.463 (0.019)
DeepSVDD-FT	0.490 (0.055)	0.459 (0.041)	0.466 (0.054)	0.471 (0.025)	0.397 (0.077)	0.504 (0.090)	0.519 (0.073)	0.478 (0.072)

Here, although DeepSVDD-FT decreases the AUC compared with DeepSVDD, considering the standard deviation of each method, the performance of DeepSVDD-FT is maintained.

Table IV represents the results with CICIDS used as the source dataset and NSL-KDD used as the target dataset. Comparing with the AUC for the case 100% data are used, the AUC for 1%, 5%, and 10% of each UAD method are lower. In this case, the results show that the fine-tuning for DNN based UAD improves the AUC. Figure 5 shows the AUC of the AE with and without fine-tuning. In this figure, when the ratio is 1%, fine-tuning improves the AUC by approximately 0.4 points. In terms of shortening the data collection time, the data collection time can be reduced to 1/20 because the AUC is about the same when using 5% of the data with fine-tuning as when using all the data.

However, in VAE, the AUC decreases much with fine-tuning. In AE, the added layers in the encoder and the encoder could learn the characteristics of the KDD data. However, since VAE estimates the input data's probability distribution, only adding one layer for the encoder and decoder might not be enough to adapt the trained model when calculating the KL divergence. This deterioration might be solved by adding more layers.

2) Results of Web system datasets: Tables V, VI, and VII show the AUC and standard deviation of each method, with two results are given in a table for each case where one dataset is used as the target dataset, and the other two datasets are used as the source datasets.

First, the AUC values are lower than the AUC for the network security dataset. This is because when an anomaly is inserted into the web system, the time of insertion is labeled as the anomaly. However, the effect of injected anomaly propagates not only to the device that inserted the anomaly but also to other adjacent devices. Therefore, returning to the data in normal state takes time after the anomaly insertion is stopped. Also, the number of samples in each dataset might be small for DNN based UAD methods even if all of the samples are used in N_{train} since each dataset includes only

119 samples. As a result, it makes it difficult for UAD methods to estimate the anomaly in web system datasets. In contrast, IF can detect the anomaly if the anomaly data is far from the data in normal state, even if the number of samples is small.

Table V shows the results of the scenario when the device is removed and the routing of the network changed from full mesh type to tree type. In this case, DeepSVDD improved the AUC by approximately 0.07 points at most except 25% with Web system 2 used as the source dataset and 10% Web system dataset used as the source dataset. When the Web system 1 dataset is used as the source data and the Web system 3 dataset is used as the target data in Table VII, AUC for 1% and 100% were improved by approximately 0.02 points by fine-tuning. For the VAE, the AUC is maintained. However, the AUC for AE decrease by approximately 0.1 at most. When the number of devices or configuration, such as routing, changes, the encoder-decoder-based methods, such as AE, may have difficulty handling significant changes in input data. On the other hand, embedding in a given hypersphere, as in DeepSVDD, is more straightforward than reconstructing the input data, increasing AUC.

Right-hand side in Tables VI and VII include the results for the case when the data characteristics are changed due to configuration update. The AUC of DeepSVDD is improved by fine-tuning except for 1% case in Tables VI and VII. The AUC of AE is improved in Table VII except the 50% case.

The results show that fine-tuning has the ability to improve the AUC in the scenarios in web system operations.

V. Conclusion

In this paper, we analyzed a fine-tuning architecture to improve UAD in an ICT system with a small amount of normal data. The performance of the fine-tuning architecture is evaluated using two types of ICT system data, which are network security data and web service system. By preparing five datasets, comprehensive experiments were conducted, and it was found that the fine-tuning architecture has the possibility to improve the accuracy of anomaly detection with a small amount of data for the ICT system scenarios.

Future work includes changing the number of fine-tuning layers, applying fine-tuning to other datasets and analyzing their performance, and investigating other effective transfer methods when the shape of the data changes significantly.

References

[1] "slack," accessed on 19th, April, 2021. [Online]. Available: https://slack.engineering/slacks-outage-on-january-4th-2021/

[2] A. Diamanti, J. M. S. Vilchez, and S. Secci, "Lstm-based radiography for anomaly detection in softwarized infrastructures," in *32nd ITC*, 2020, pp. 28–36.

[3] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proc. ACM SIGSAC CCS*, ser. CCS '17, 2017, p. 1285–1298.

[4] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, and N. V. Chawla, "A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data," in *Proc. AAAI*, vol. 33, no. 01, 2019, pp. 1409–1416.

[5] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, "Usad: Unsupervised anomaly detection on multivariate time series," in *Proc. ACM SIGKDD*, 2020, p. 3395–3404.

[6] R. C. Aygun and A. G. Yavuz, "Network anomaly detection with stochastically improved autoencoder based models," in *4th Int. Conf. CSCloud*, 2017, pp. 193–198.

[7] W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei, Y. Liu, Y. Chen, R. Zhang, S. Tao, P. Sun, and R. Zhou, "Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs," in *Proc. 28th IJCAI*, 2019, pp. 4739–4745.

[8] H. Zhao, Y. Wang, J. Duan, C. Huang, D. Cao, Y. Tong, B. Xu, J. Bai, J. Tong, and Q. Zhang, "Multivariate time-series anomaly detection via graph attention network," in *ICDM*, 2020, pp. 841–850.

[9] D. Li, Q. Yuan, T. Li, S. Chen, and J. Yang, "Cross-domain network traffic classification using unsupervised domain adaptation," in *Proc. ICOIN*, 2020, pp. 245–250.

[10] S. Tobiyama, B. Hu, K. Kamiya, and K. Takahashi, "Large-scale network-traffic-identification method with domain adaptation," in *Proc. WWW*, 2020, p. 109–110.

[11] H. Larsson, F. Moradi, J. Taghia, X. Lan, and A. Johnsson, "Domain adaptation for network performance modeling with and without labeled data," in *Proc. NOMS*, 2023, pp. 1–9.

[12] S. Saha and A. Haque, "Wavelet-based hybrid machine learning model for out-of-distribution internet traffic prediction," in *Proc. NOMS*, 2023, pp. 1–8.

[13] Y. Katz and D. Raz, "Cold start for cloud anomaly detection," in *Proc. NOMS*, 2023, pp. 1–8.

[14] S. Nedelkoski, J. Bogatinovski, A. Acker, J. Cardoso, and O. Kao, "Self-attentive classification-based anomaly detection in unstructured logs," in *Proc. ICDM*, 2020, pp. 1196–1201.

[15] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *JMLR*, vol. 17, no. 1, pp. 2096–2030, 2016.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[17] J. Lin, S. Chen, E. Lin, and Y. Yang, "Deep feature selection for anomaly detection based on pretrained network and gaussian discriminative analysis," *IEEE Trans Instrum Meas*, vol. 1, pp. 1–11, 2022.

[18] L. Borawar and R. Kaur, "Anomaly detection methods in surveillance videos: A survey," in *Proc. SMART GENCON*, 2022, pp. 1–7.

[19] H. Liu and L. Li, "Anomaly detection of high-frequency sensing data in transportation infrastructure monitoring system based on fine-tuned model," *IEEE Sens. J.*, vol. 23, no. 8, pp. 8630–8638, 2023.

[20] Y. Zhao, R. Yang, N. Yang, T. Lin, Q. Fu, and Y. Ma, "Robust log-based anomaly detection with hierarchical contrastive learning," in *Proc. ICASSP*, 2023, pp. 1–5.

[21] M. Sun, Y. Su, S. Zhang, Y. Cao, Y. Liu, D. Pei, W. Wu, Y. Zhang, X. Liu, and J. Tang, "CTF: Anomaly detection in high-dimensional time series with coarse-to-fine model transfer," in *Proc. INFOCOM*, 2021, pp. 1–10.

[22] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Proc. MLSDA*, ser. MLSDA'14, 2014, p. 4–11.

[23] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. ICLR*, Y. Bengio and Y. LeCun, Eds., 2014. [Online]. Available: http://arxiv.org/abs/1312.6114

[24] S. Kullback, *Information theory and statistics*. Courier Corporation, 1997.

[25] L. Ruff, R. A. Vandermeulen, N. Görnitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *Proc. of the 35th ICML*, vol. 80, 2018, pp. 4393–4402.

[26] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: Identifying density-based local outliers," *SIGMOD Rec.*, vol. 29, no. 2, p. 93–104, 2000.

[27] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. ICDM*, 2008, pp. 413–422.

[28] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *Proc. CISDA*, 2009, pp. 1–6.

[29] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." *ICISSp*, vol. 1, pp. 108–116, 2018.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.