Human-Assisted Online Anomaly Detection with Normal Outlier Retraining

Yasuhiro Ikeda
NTT Network Technology Laboratories, NTT Corporation
yasuhiro.ikeda.sm@hco.ntt.co.jp

Keisuke Ishibashi
NTT Network Technology Laboratories, NTT Corporation
keisuke.ishibashi.zf@hco.ntt.co.jp

Yuusuke Nakano
NTT Network Technology Laboratories, NTT Corporation
yuusuke.nakano.dn@hco.ntt.co.jp

Keishiro Watanabe
NTT Network Technology Laboratories, NTT Corporation
keishiro.watanabe.ry@hco.ntt.co.jp

Kengo Tajiri
NTT Network Technology Laboratories, NTT Corporation
kengo.tajiri.bk@hco.ntt.co.jp

Ryoichi Kawahara
Faculty of Information Networking for Innovation and Design, Toyo University
ryoichi.kawahara@iniad.org

ABSTRACT

Anomaly detection has been essential for information and communications technology (ICT) systems to maintain reliable services. For continually using anomaly detection algorithms for ICT systems, the non-stationarity of such systems should be considered. Although several online anomaly detection algorithms has been investigated for addressing non-stationarity, there has been a fundamental problem of applying the algorithms to real environments, that is, outliers are generally assumed as anomalies with conventional unsupervised algorithms. However, in reality, many normal data are often observed as outliers due to their rarity or emerging new patterns, which we call *normal outliers*. Such *normal outliers* will cause false alarms which lose operator's trust and will result in overlooking of true anomalies. In this paper, we propose an online anomaly detection algorithm that exploits an operator's judgment of detection results to retrain *normal outliers* as normal by learning normal data boundary in the feature space of the data. We exploit elastic weight consolidation (EWC) for learning *normal outliers* as normal in online fashion while remembering the already trained situations for avoiding overfitting without using old training data. We evaluated our algorithm with network benchmark data and real measured data from an enterprise service. We found that our algorithm reduces false alarms due to the *normal outliers* by about 79% compared to a simple sliding window algorithm without degrading the anomaly detection accuracy. [1]

[1]This paper is an extended version of our technical report [8] with no peer review. The difference is the improvement in the proposed algorithm and additional evaluations with real measured data.

ODD v5.0, ACM SIGKDD 2018 Workshop Aug. 20, 2018, London, UK
© 2018 ACM. ISBN 123-4567-24-567/08/06...$15.00
DOI: 10.475/123_4

CCS Concepts

•Computing methodologies → Anomaly detection; •Computer systems organization → Maintainability and maintenance;

Keywords

Online Anomaly Detection; Unsupervised Learning; Normal Outlier; Deep Learning

1. INTRODUCTION

Anomaly detection has been essential for information and communications technology (ICT) systems to maintain reliable services. For continually monitoring ICT systems with anomaly detection algorithms, the non-stationarity of such systems, due to secular changes of these systems or trends in usage change, should be considered as concept drift [11]. Several online anomaly detection algorithms has been investigated for addressing concept drift; however, there has been a fundamental problem of applying such algorithms to real environments: that is, although temporarily outlying observation is generally assumed as an anomaly with conventional unsupervised algorithms, there can be data that are outlying but actually normal.

For monitoring ICT systems, supervised algorithms for classifying the states of such systems into normal or abnormal are not realistic, because not only abnormal states of ICT systems varies and sufficient labeled data of each state will not be available, but also unknown fault situations will emerge in recent ICT systems due to system multiplexing under a virtualization environment. Therefore, unsupervised algorithms that learn "normal states" of ICT systems by using only data in normal situations are promising for detecting varying anomalies without using labeled data that are difficult to obtain.

With most unsupervised anomaly detection algorithms, outlying data, which are considerably different from the majority of normal data, are assumed as anomalies. However, as Sadik et al. [16] also discussed, there are cases in which data are outlying but operators who use the algorithms assume them as normal, called *normal outliers* in this paper, due to their rarity or emerging new patterns after training (see Fig. 1). For example, in the case of network monitoring, newly appearing types of connections due to the emergence of a new application may be assumed as outliers in the feature space of monitoring data. Another example is the case of server monitoring. In ICT services, since several server processes are required and the frequency of occurrence of requirements varies, there can be many processes that do not often occur and their states are considerably different from usual state, such as regular backups. Such situations become outliers in the feature spaces of data and are assumed as anomalies with the conventional algorithms. Although operators may ignore alarms during regular backups, for example, this can result in overlooking of true anomalies in backup processes. Therefore, learning *normal outliers* as normal is essential for avoiding such misdetection and overlooking, that is, false positives (FPs) and false negatives (FNs).

In this paper, we propose an online unsupervised anomaly detection algorithm for retraining *normal outliers* as normal. With the algorithm, once a FP due to a *normal outlier* is observed by the operator, the anomaly detection model is retrained to learn the outlying data as normal, preventing a recurrence of similar FPs. The algorithm also learns *incremental drift*, which expresses gradual changes in normal status and is one of the major issues of concept drift, by using recent data for retraining. The *incremental drift* is automatically retrained as the recent data for retraining is stored, on the other hand, the *normal outlier* is retrained at a timing when the operator confirmed the detection is a FP. Our algorithm is based on autoencoders [23, 17, 3, 26], a class of classification algorithms with which outliers are not simply assumed as anomalies and that learns a boundary that distinguishes normal and abnormal data in feature space. The learning of boundaries is biased by the majority of training data and the outliers in training data tend to be ignored in a naive autoencoder; therefore, our algorithm uses an incremental learning model to learn boundaries that can include both majority and outliers in training data. For incremental learning, we exploit elastic weight consolidation (EWC) [10] for learning *normal outliers* as normal while remembering the already trained situations without old training data. Although one might also be concerned with *normal outliers* and not want to neglect, the retraining algorithm will be concurrently utilized with a conventional anomaly detection algorithm for false-alarm-sensitive situations.

Our contributions are as follows:

- Our algorithm retrains the anomaly detection model to learn *normal outliers*, which have been just assumed as anomalies, as normal once observed.
- Our algorithm requires only recent data for following *incremental drift*, while not forgetting past training.

Through evaluations with network benchmark data and real measured data from an enterprise system, we argue that the proposed algorithm reduces FPs caused by *normal outliers* and *incremental drift* without degrading the anomaly detection accuracy compared to state-of-the-art algorithms.

The paper is organized as follows. We first summarize related work in Section 2 and explain the principle of autoencoders in Section 3. We describe our proposed online anomaly detection algorithm in Section 4. After evaluating the algorithm in Section 5, we conclude our paper in Section 6.

2. RELATED WORK

As Chandola et al. stated [4], there are several types of unsupervised techniques for anomaly detection. Although many types of unsupervised techniques just define outliers as anomalies, classification techniques define regions that distinguish between normal and abnormal data in feature space; therefore, they are promising for learning *normal outliers*. Autoencoders, which is based on nonlinear dimensionality reduction using deep neural networks, are attracting much attention [23, 17, 3, 26]. Although autoencoders are promising for anomaly detection in ICT systems in which monitored data tend to be high-dimensional, online adaptation of autoencoders for streaming data has not been extensively investigated. Yue and Nathalie [5], who only addressed online anomaly detection with autoencoders to the best of our knowledge, proposed an ensemble approach of autoencoders to address concept drift. However, they did not consider *normal outliers*, which we focus on.

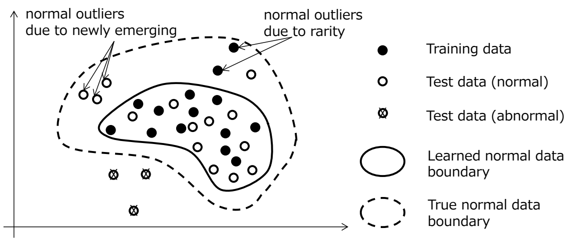

normal outliers due to newly emerging
normal outliers due to rarity
Training data
Test data (normal)
Test data (abnormal)
Learned normal data boundary
True normal data boundary

Figure 1: Image of *normal outliers*

Online learning for classification has been extensively investigated with consideration of concept drift. Since fault situations of ICT systems vary and rarely occur, labeled data will be difficult to obtain and therefore unsupervised techniques for online anomaly detection are important. Using principal component analysis (PCA) is a representative unsupervised technique [21, 25]. Sun et al. [21] proposed an adaptive PCA algorithm with a sliding window to follow concept drift. They use neural networks for extracting principal component eigenvectors of a large matrix. Instead of using sliding windows, Xie and Krishnan [25] proposed a non-overlapping moving-window technique for dynamic PCA to improve detection accuracy. Different from these PCA based approaches, Siffer et al. [18] proposed an online outlier detection algorithm based on extreme value theory while also considering concept drift. However, the algorithms are insufficient for following concept evolution [15], which is the appearance of a novel class of data, and therefore is often equivalent to *normal outliers*.

Some studies have addressed unsupervised techniques considering concept evolution [19, 14, 22]. Spinosa et al. [19] proposed the Online Novelty and Drift Detection Algorithm (OLINDDA) for not only detecting anomalies but also learning newly emerged concepts using the k-means clustering algorithm. OLINDDA stores outliers to a short-time memory, and if outliers form a new cluster, the outliers are assumed as the newly emerged concept. Masud et al. [14] also used k-means for outlier detection and distinguished anomalies and outliers due to concept evolution according to the degree of cohesion among the outliers. Tharkan and Toshniwal [22] proposed an algorithm using density-based spatial clustering of applications with noise (DBSCAN) and weighted k-means clustering. With the algorithm, the outlying data of DBSCAN are considered candidate outliers, and if they fall in a new cluster of weighted k-means, they are assumed as concept evolution. Although these algorithms also take into account *normal outliers* and address retraining for judging them as normal, sparsely-occurring *normal outliers* cannot be retrained as normal since it is assumed that the *normal outliers* form a new cluster in several time windows. However, our algorithm retrains the normal data boundary once a *normal outlier* is observed; therefore, it decreases the number of not only similar FPs but also other types of FPs.

3. AUTOENCODER

An autoencoders is an artificial neural network composed of input, output, and one or more hidden layers and is originally used for dimensionality reduction. By expressing input data as N-dimensional vector $\boldsymbol{x}^{(1)}$, the number of layers as L, output values in each layer as $\boldsymbol{x}^{(l)}$, $l = 2, ..., L$, the output values are calculated using the following recursive formula:

$$\boldsymbol{x}^{(l)} = \phi^{(l)}(W^{(l)}\boldsymbol{x}^{(l-1)} + \boldsymbol{b}^{(l)}),\ l = 2, ..., L, \tag{1}$$

where $W^{(l)}$ is the weight matrix between the $(l-1)$th and lth layer, $\boldsymbol{b}^{(l)}$ is the bias in the lth layer, and $\phi^{(l)}$ is the activation function. From this, we omit subscript (1) of $\boldsymbol{x}^{(1)}$ and simply denote it as $\boldsymbol{x}$. Activation functions perform element-wise transformation and enable autoencoders to learn non-linear dimensional reduction. The objective of autoencoders is to reconstruct the input data in the output layer, which corresponds to dimensional reduction in the hidden layer since the size of the hidden layer is set to lower than that of the input data. In using autoencoders as anomaly detection, it is assumed that normal data has normal correlations among the dimensions and distributes around low-dimensional manifold. Therefore, learning dimensionality reduction using normal data corresponds to learning the manifold. By training the autoencoder using normal data, the anomaly score of test data $\boldsymbol{x}$ is defined by the following mean squared error (MSE) [3]:

$$MSE(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N}(x_i^{(L)} - x_i)^2, \tag{2}$$

where x_i is the i-th dimensional value of $\boldsymbol{x}$. As in [23, 17, 3, 26], we assume the normal correlations among the dimensions are collapsed in anomalous data and the data deviates from the learned low-dimensional manifold. Since the MSE is the difference between test data and reconstructed data from the manifold, it is assumed as the anomaly degree of the test data.

Training autoencoders is to optimize parameters: $W^{(l)}, \boldsymbol{b}^{(l)}, l = 2, ..., L$. By using training data set $\boldsymbol{x}_1, ..., \boldsymbol{x}_T$, which contains the data in normal situations, the parameters are optimized so that the mean of MSE with the training data is minimized by solving the following problem, which is called batch learning.

$$\min_{W^{(l)},\boldsymbol{b}^{(l)}} \frac{1}{T}\frac{1}{N}\sum_{t=1}^{T}\sum_{i=1}^{N}(x_{t,i}^{(L)} - x_{t,i})^2\ l = 2, ..., L. \tag{3}$$

Note that input data to autoencoders should be preprocessed for normalizing or standardizing values in each dimension since the difference in scale among the dimensions of input data will affect both parameter optimization and calculation of MSE. The typical preprocessing is normalization [3] or standardization [17] by using the statistics of each dimension in normal data.

4. PROPOSED ALGORITHM

In this section, we describe our proposed algorithm. A simple image of the algorithm is depicted in Fig. 2 and its pseudo code is in Algorithm 1. If online test data is determined as normal with the algorithm or, the operator confirmed that the detection is a FP even though the data was determined as anomaly, the data is used for retraining. For the confirmation of a FP, the algorithm for identifying contributing features to the anomaly [7], which suggests the cause of the detection, may be exploited. Note that since our autoencoder-based algorithm does not depend on temporal information, the timing of confirming if the detection is a FP or not and using it for retraining needs not to be real time.

The algorithm involves not only retraining of the autoencoder but also the updating of preprocessing parameters and thresholds. Updating preprocessing parameters corresponds to the online update of minimum and maximum or mean and standard deviation values for each dimension for normalizing or standardizing the input data, respectively. We now explain the retraining model and updating threshold, and then describe the sequence of the algorithm.

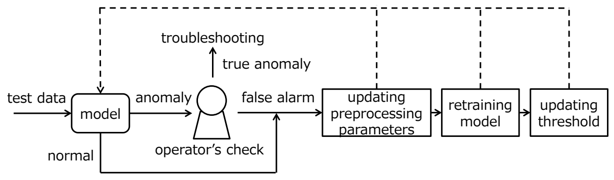


Figure 2: Image of proposed algorithm.

4.1 Retraining Model

For adopting autoencoders to online anomaly detection, parameters $W^{(l)}, \boldsymbol{b}^{(l)}, l = 2, ..., L$ should be adequately retrained. In the case of autoencoders, the *incremental drift* can also be retrained by using batch learning (Eq. (3)) with the recent normal data similar to the conventional sliding window methods since the changes in the data are moderate and using the recent data for retraining is sufficient for following the *incremental drift*.

Normal outliers, however, are difficult to retrain as normal by using conventional batch learning since batch learning is executed to minimize the mean MSE over all the training data; therefore, minimizing the MSE for outliers tends to be ignored. Although replicating the outliers in the training data seems to be effective to overcome this problem, as confirmed from numerical evaluation discussed in Section 5.1, it fails since the best amount of replicating depends on both the degree of outlying and distribution of other normal data. In addition, the batch learning approach requires past training data, which is not promising for large ICT systems. Another simple solution is to retrain the model to minimize the MSE only for the outlier data by using Eq. (2); however, this will cause catastrophic forgetting [6] and lead to other FPs that were properly determined as normal in past trainings.

Therefore, we propose an online anomaly detection algorithm which retrains both *incremental drift* and *normal outliers* as normal. With the algorithm, a *normal outlier* is retrained to minimize its MSE while avoiding catastrophic forgetting with EWC [10]. In EWC, parameters $\boldsymbol{\theta} = [\theta_1, ..., \theta_K]^T$ of neural networks are trained by adding a penalty function whose coefficients are the diagonal of the Fisher information matrix. The Fisher information matrix is a $K \times K$ matrix in which each element is calculated as $E\left[\frac{\partial}{\partial\theta_i}\log f(X;\boldsymbol{\theta})\frac{\partial}{\partial\theta_j}\log f(X;\boldsymbol{\theta})|\boldsymbol{\theta}\right]$, where $f(X;\boldsymbol{\theta})$ is a probability density function of the model and X is an observable random variable, and Fisher information corresponds to the amount of information X has for parameter $\boldsymbol{\theta}$. The principle of EWC is that by assuming the negative of the loss function for the problem as log probability of the data $\log p(\mathcal{D}|\boldsymbol{\theta})$ and the data are split into two independent parts $\mathcal{D}_A$ and $\mathcal{D}_B$, the probability can be expressed as $\log p(\mathcal{D}|\boldsymbol{\theta}) = \log p(\mathcal{D}_B|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}|\mathcal{D}_A) - \log p(\mathcal{D}_B)$ and the maximizing of $\log p(\mathcal{D}|\boldsymbol{\theta})$ corresponds to that of $\log p(\mathcal{D}_B|\boldsymbol{\theta})$ with a quadratic penalty whose coefficients are the diagonal of the Fisher information matrix calculated with $\mathcal{D}_A$ under the Laplace approximation.

Since our purpose is not just minimizing MSEs of training data but learning a normal data boundary, which includes *normal outliers*, we assume the loss function as $MSE(D) + g(D)$, where $MSE(D)$ is the mean of the MSEs over D and $g(D)$ is 0 if all

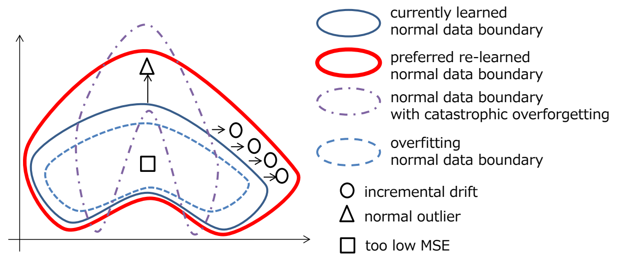

currently learned normal data boundary
preferred re-learned normal data boundary
normal data boundary with catastrophic overforgetting
overfitting normal data boundary
incremental drift
normal outlier
too low MSE

Figure 3: Image of retraining. Boundaries express region where MSE becomes below threshold $\gamma(\alpha)$.

the data in D are inside the normal data boundary; otherwise, ∞. By assuming $\mathcal{D}_A$ and $\mathcal{D}_B$ as data already trained as normal and as outlying normal data $\boldsymbol{x}_t$, respectively, and approximating that a *normal outlier* occurs independent from $\mathcal{D}_A$, minimizing the loss function of EWC becomes the learning of the parameters that determine $\boldsymbol{x}_t$ as normal with the penalty function; thus, the problem can be expressed as follows:

$$\min_{\boldsymbol{\theta}} \frac{1}{N}\sum_{i=1}^{N}(x_{t,i}^{(L)} - x_{t,i})^2 + \sum_{k=1}^{K}\frac{\lambda}{2}F_k(\theta_k - \theta_k^*)^2, \tag{4}$$

$$\text{sbj.to } \frac{1}{N}\sum_{i=1}^{N}(x_{t,i}^{(L)} - x_{t,i})^2 \leq \text{threshold}, \tag{5}$$

where $\boldsymbol{\theta}$ is all the parameters in $\{W^{(l)}, \boldsymbol{b}^{(l)}\}$, $l = 2, ..., L$, F_k is the kth element of diagonal of Fisher information matrix F, K is the total number of the parameters, θ_k^* is the value of the parameter before retraining, and λ is a weight to determine the strength of the penalty. Note that F can be computed by assuming the loss function as $MSE(\mathcal{D}_A)$ since $\mathcal{D}_A$ has already been trained as normal; therefore, $g(\mathcal{D}_A) = 0$. To ensure that $\mathcal{D}_B$ and similar *normal outliers* are retrained as normal with Eq. (5), we set the threshold to the Q3 value of MSEs of normal data, which is discussed in Section 4.2.

Once the model is retrained, the Fisher information is updated according to $\boldsymbol{x}_t$ and the retrained model. Although the original EWC just adds new Fisher information to the previous Fisher information, it does not work with our algorithm due to the deviation from the strong assumption of independence between past training data and outliers. For example, if one retraining with a *normal outlier* is not sufficient to reduce all similar outliers and another similar *normal outlier* was observed again, the Fisher information with the outlier should resemble that of the already retrained *normal outlier* and immediately adding that could cause bias. Therefore, we update the Fisher information by adding new information with weight according to the distance from the previous information as $F_k \leftarrow F_k + ||\hat{F} - \hat{F}_{new,k}|| \cdot F_{new,k}$, where $\hat{F}$ is the normalized diagonal of Fisher information by its maximum. The value range of F varies in the online retraining; therefore, we determine the weight of penalty function λ as inversely proportional to the current mean value of F as $\lambda = \frac{\lambda_0}{F_{mean}}$, where λ_0 is a hyperparameter. Since the problem of Eqs. (4) and (5) may be infeasible if λ was too high, λ_0 should be decreased when a solution could not be obtained. In this paper, we decrease λ_0 at a rate of 0.9 if solving the problem of Eqs. (4) and (5) failed. We evaluated our algorithm with several initial λ_0 in Section 5.1.

4.2 Updating Threshold

Since the MSE of autoencoders is a continuous value, we should determine the threshold values of the MSE for determining if the input data are abnormal. In this paper, we calculate the distribution of the MSEs of normal data with a trained autoencoder and define the threshold as the outlier degree of the MSE from the distribution. Since the distribution of MSEs may not follow a normal distribution due to the presence of *normal outliers*, we use quartiles to express the distribution and determine the threshold according to the deviation from Q3 normalized by the interquartile range (IQR). The threshold value of the MSE is then expressed as $\gamma(\alpha, \rho_{mse}, \tau_{mse}) = \rho_{mse} + \alpha\tau_{mse}$, where ρ_{mse} and τ_{mse} are the Q3 and IQR values of the MSE with training data, respectively, and α is a hyperparameter to set the threshold value. Afterwards, we

Algorithm 1 Pseudo code of the proposed algorithm

```
 1: function $RET_{out}(\boldsymbol{x})$
 2:     Retrain by Eqs. (4) and (5) with $\boldsymbol{x}$
 3:     Update $F$
 4: end function

 5: function $RET_{inc}(X)$
 6:     Retrain by Eq. (3) with $X$ while adding EWC penalty
 7:     Update $F$
 8: end function

 9: function $UpdateThreshold(X)$
10:     Update $\rho_{mse}$ and $\tau_{mse}$ with current model and $X$
11: end function

12: Offline Retraining Phase
13: Train by Eq. (3) with $X_{train}$
14: for all $\boldsymbol{x}_{train,l}$ in $X_{train}$ do
15:     Calculate $MSE(\boldsymbol{x}_{train,l})$
16:     if $\gamma(\alpha_H) \leq MSE(\boldsymbol{x}_{train,l})$ then
17:         $RET_{out}(\boldsymbol{x}_{train,l})$
18:         $UpdateThreshold(X_{train})$
19:     end if
20: end for

21: Online Retraining Phase
22: $X_{normal} \leftarrow$ some recent data
23: $X_{ret} \leftarrow X_{train}$
24: $retrain\_count \leftarrow 0$
25: for online test data $\boldsymbol{x}_t$ do
26:     if $MSE(\boldsymbol{x}_t) < \gamma(\alpha_{ano})$ or operator judged $\boldsymbol{x}_t$ as normal then
27:         Update preprocessing parameters
28:         Add $\boldsymbol{x}_t$ to $X_{normal}$ and delete oldest
29:         if $\gamma(\alpha_L) \leq MSE(\boldsymbol{x}_t) < \gamma(\alpha_H)$ then
30:             $retrain\_count \leftarrow retrain\_count + 1$
31:             Add $\boldsymbol{x}_t$ to $X_{ret}$ and delete oldest
32:             if $retrain\_count = T_{ret}$ then
33:                 $RET_{inc}(X_{ret})$
34:                 $retrain\_count \leftarrow 0$
35:             end if
36:         else if $\gamma(\alpha_H) \leq MSE(\boldsymbol{x}_t)$ then
37:             $RET_{out}(\boldsymbol{x}_t)$
38:         end if
39:         $UpdateThreshold(X_{normal})$
40:     else
41:         Troubleshooting for the anomaly
42:     end if
43: end for
```

simply denote thresholds as $\gamma(\alpha)$. Since ρ_{mse} and τ_{mse} depend on both the normal data and model, we store recent normal data and update these parameters with the data and current model in the online training.

4.3 Sequence of algorithm

We now describe the sequence of our algorithm (see Algorithm 1). The algorithm is composed of two retraining methods; one is batch learning for the *incremental drift* (RET_{inc}) and the other is learning with EWC for *normal outliers* (RET_{out}). To determine which retraining is executed for a test data, we introduce MSE threshold $\gamma(\alpha_H)$. This is because the MSEs with the *incremental drift* tend to be lower since the changes in data are moderate; however, those with *normal outliers* tend to be higher since the data are quite different from already trained data (see Fig. 3). We also introduce lower threshold $\gamma(\alpha_L)$ for RET_{inc} since retraining with data that are not much different from already trained data will cause overfitting, as the evaluation discussed in Section 5.1 shows.

Our algorithm is divided into two retraining phases: offline and online. In the offline retraining phase, an autoencoder is normally trained with initial training data X_{train} first, after that, the algorithm is retrained with all data in X_{train} if the MSE with the data is larger than $\gamma(\alpha_H)$, which is defined as the *normal outlier* in the training data. After the offline retraining phase, the algorithm moves to the online retraining phase for anomaly detection and retraining with the online test data. In this phase, the MSE with test data $\boldsymbol{x}_t$ is first calculated and compared with threshold $\gamma(\alpha_{ano})$ as anomaly detection. The test data, which are determined as normal according to the threshold or as abnormal but operators confirmed them as an FP, are used for retraining. Note that confirmation by operators is not necessarily required instantly, as discussed above. Our algorithm determines which retraining should be executed for the test data. If the MSE of the data is below $\gamma(\alpha_L)$, the data are not used for retraining to avoid overfitting. If the MSE is between $\gamma(\alpha_L)$ and $\gamma(\alpha_H)$, the data are regarded as the *incremental drift* and stored for RET_{inc}. If it is larger than $\gamma(\alpha_H)$, the data are regarded as *normal outliers* and used for RET_{out}. For initial training and RET_{inc}, we used mini-batch learning with Adam [9], and for minimizing Eq (4) in RET_{out}, we used a simple gradient descent (GD) method.

5. EVALUATION

We evaluated our algorithm with network benchmark data and real measured data from an enterprise system. We used a five-layered autoencoder as the anomaly detection model. The hyper-parameters used in both evaluations are described in Table 1. We executed the calculations of the autoencoder using deep learning framework, Chainer v3.0 [24].

5.1 Evaluation with Network Benchmark Data

We evaluated the algorithm with the NSL-KDD dataset [2] to examine its effectiveness in reducing FPs due to outlying normal communications and changes in generalization performance. We first evaluate only RET_{out} for retraining *normal outliers* as normal and then evaluate both RET_{inc} and RET_{out} with data that include pseudo *incremental drift*.

5.1.1 Data Description

The NSL-KDD dataset is the amended version of the KDD Cup 1999 dataset [1], which is a representative network benchmark used for the data mining competition in KDD'99. The dataset is composed of about five million connection records, which are classified into five main classes: one normal class and four attack classes, and includes training data (NSL-KDD-train) and test data (NSL-KDD-test). One record consists of 34 continuous and 7 discrete feature values. By expressing the discrete values as one-hot representation, we converted the data as a set of 122 feature vectors. Since many binary features are included, we used normalization as preprocessing.

5.1.2 Evaluation Method

Although the proposed algorithm is for online anomaly detection, the NSL-KDD dataset does not contain time-series data. Therefore, we assumed the data as time-series data in ascending order of record ids. We first extracted 67,343 normal records in NSL-KDD-train and used the first 5,000 records as initial training data X_{train} in the offline retraining phase and the remaining 62,343 records as the test data in the online retraining phase. Since all test data in the online retraining phase were normal, all the detections were FPs and the objective was to reduce them. We also evaluated the generalization performance by using NSL-KDD-test as validation data D_{val}. Since D_{val} includes both normal and attack connections, we could evaluate the change in the generalization performance as the detection accuracy of the attack connections through the online retraining phase. We set α_{ano} to 3 and evaluated the algorithm with varying parameters α_L, α_H, and λ_0.

5.1.3 Evaluation Results

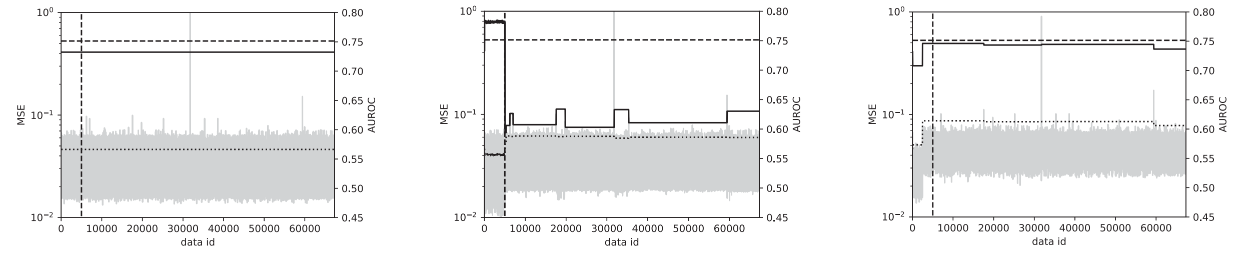
We first evaluated the performance of RET_{out} for retraining *normal outliers* as normal against two naive methods: only updating normalization parameters and thresholds (*parameters*) and batch training with past training data and inflated *normal outlier* (*batch_inflate*). With *batch_inflate*, we replicated the *normal outlier* to the same amount of the past training data and combined them as the new training data. Note that the batch training method is not suitable for large ICT systems since it should maintain all past training data. Figure 4 shows the results of each method. As shown in the figure, RET_{out} outperformed in terms of both decreasing the number of FPs and generalization performance compared to the other naive methods. The average MSE tended to increase because RET_{out} continued retraining for the outlying data and the parameters of the autoencoder slightly departed from the original ones, which minimized the MSEs of the majority of normal data. However, the increase in the MSEs of normal data also increased the threshold value of the MSEs. Retraining for outlying data will enlarge the area of normal data boundary (see Fig. 3). Although it likely produces FNs, that is, judging abnormal data as normal, the improvement of AUROC for validation data suggests that we can reduce FPs without increasing FNs with a proper threshold. How to determine the threshold should be further discussed in future work.

We also evaluated the dependencies on parameters λ_0 and α_H in Fig. 5a. Note that λ_0 was not decreased throughout the retraining since RET_{out} did not failed even once in this evaluation . Retraining without considering Fisher information ($\lambda = 0$) resulted in less generalization performance compared to when $\lambda > 0$. This is because the retraining without EWC drastically changes the normal data boundary and other normal data inside the previous boundary may protrude, which will cause other FPs. In addition, the result with $\alpha_H = 3$ was even worse since it caused overfitting by continuing fitting to all the slightly outlying data.

Next, we evaluated our algorithm with data that include *incremental drift*. By assuming the NSL-KDD dataset as the time-series data with *incremental drift*, we renumbered the record ids in ascending order of feature values *src_byte*, which express the number of bytes sent in each connection, for providing pseudo *incremental drift* in the dataset. As shown in Fig. 5b, the number of FPs continually increased since feature value *src_byte* continually in-

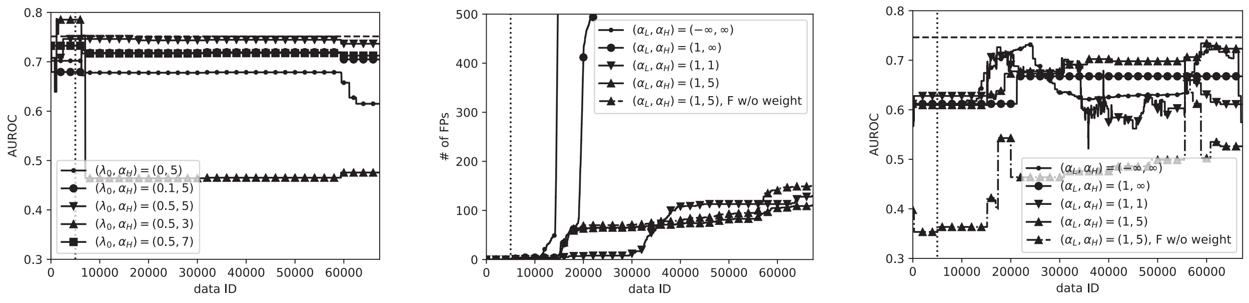
Table 1: Hyperparameters of autoencoder

Hyperparameter		Benchmark data	Real measured data
Autoencoder	size of second, third, and fourth layers	(61,31,61)	(86,43,86)
	activation function (only in second and fourth layers)	ReLU	
	L2 weight decay in initial training	0.0005	0.0001
	# of epochs in initial training and RET_{inc}	200	
	mini-batch size in Adam	100	10
	step size in Adam and GD	(0.001, 0.01)	(0.001, 0.001)
	dropout rate in initial training and RET_{inc}	0.2 for input layer and 0.5 for all hidden layers [20]	
Online retraining	$(T_{ret}, \|X_{retrain}\|, \|X_{normal}\|)$	(500,5000,100)	(1008,4320,144)



(a) *parameters*($\#_{FP}$ = 2,708) (b) *batch_inflate*($\#_{FP}$ = 645) (c) RET_{out}($\#_{FP}$ = 11)

Figure 4: Results of retraining of *normal outliers*. Gray lines show MSE, dotted lines show $\gamma(\alpha_{ano})$, solid lines show area under a receiver operating characteristic curve (AUROC) for D_{val}, horizontal broken lines show AUROC for D_{val} with offline-trained model with NSL-KDD-train, and vertical broken lines show boundary of offline and online retraining. $\#_{FP}$ is the number of FPs and that without any retraining was 2,748.



(a) Dependencies on λ_0 and α_H (b) $\#_{FP}$ with *incremental drift* (λ_0 = 0.5) (c) AUROC for D_{val} with *incremental drift* (λ_0 = 0.5)

Figure 5: Evaluation on parameters. "F w/o weight" is result with updating F_k without weighting. Note that eventual $\#_{FP}$ in *no retraining*, *parameters*, $(\alpha_L, \alpha_H) = (-\infty, \infty)$, and $(\alpha_L, \alpha_H) = (1, \infty)$ were 57,428, 583, 4,603, and 1,213, respectively.

creased after the training. The two lines with $\alpha_H = \infty$ in Fig. 5c are the results of evaluation for only RET_{inc} with respect to threshold α_L introduced for avoiding overfitting. The comparison clarifies that retraining without α_L causes unsustainability of the model and many FPs due to overfitting. The line with $(\alpha_L, \alpha_H) = (1, 1)$ is, in contrast, the result in which RET_{out} was executed for all the data over the threshold and shows that retraining by only RET_{out} also causes unsustainability of generalization performance due to overfitting. The results with $(\alpha_L, \alpha_H) = (1, 5)$, which is our proposed algorithm using both RET_{inc} and RET_{out}, not only reduced the number of FPs but also improved generalization performance. Note that the AUROC of the algorithm in which Fisher information F_k is updated without weighting is quite lower since it ignores the dependencies between a retraining *normal outlier* and the already retrained ones. In conclusion, the evaluations showed that our algorithm could reduce FPs due to both *incremental drift* and *normal outliers* without degrading the generalization performance in the network benchmark data.

5.2 Evaluation with Real Measured Data

We evaluated our algorithm with real measured data from an enterprise system to confirm the retraining effect considering both *incremental drift* and *normal outliers* occurring in a real environment and if RET_{out} can reduce similar FPs.

5.2.1 Data Description

The system for which we evaluated our algorithm manages the account information of a few hundred thousand users of a service

Table 2: Categories of sar data

Categories	Description	# of Features
CPU (× 40)	Usage of CPU	9 (× 3 statistics)
DISK (× 9)	Device activity	8 (× 3 statistics)
IFACE (× 36)	Network condition	16 (× 3 statistics)
PROCESS	Process condition	1
DISK-IO	Disk I/O condition	5
SWITCH	Switch activity	1
SWAP	Swap activity	2
PAGE	Paging activity	9
MEMORY	Memory assigning activity	3
MEMORY and SWAP	Memory and swap conditions	12
INODE	Inode condition	4
QUEUE	CPU wait condition	5
TTY	TTY device activity	7
NFS	NFS client activiy	6
NFSD	NFS server activity	11
SOCK	Socket condition	6

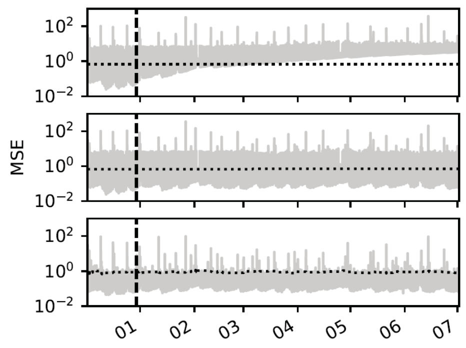


Figure 6: Transition of MSE. (top) *conventional*, (middle) *inc_only*, and (bottom) *inc_and_out*

Table 3: $\#_{FP}$ and average AUROC.

	conventional	[22]	[5]	*inc_only*	*inc_and_out*
$\#_{FP}$	20,676	635	10,657	1,878	394
AUROC	1.0	0.3	1.0	1.0	1.0

and uses the Unix operating system. We used system activity report (sar) data monitored every ten minutes as time-series measured data. The categories of sar data are listed in Table 2. Since the features of CPU, DISK, and IFACE are described for each CPU, device, and interface, we extracted the mean, max, and standard deviation values over all CPUs, devices, or interfaces as feature values. This is for making the anomaly detector robust against frequent system switching. Consequently, 171 feature values were extracted from the sar data. Since all the values are continuous and the range can become very wide, we used standardization as preprocessing. Due to several procedures in the system, such as data backup, the data included many *normal outliers*. In addition, since a temporally increasing feature was included, there is also *incremental drift*. We evaluated if the proposed algorithm reduces FPs caused by the *normal outliers* and the *incremental drift*.

5.2.2 Evaluation Method

The data were measured over seven months. Since there was no anomaly event in the first seven months, the first month's data were used for initial training and the following six months data were for online testing. For evaluating generalization performance, we used nine days of data measured after online test data in which the system update is executed as validation data. Since unusual processes are enforced during the update, such as switching to redundant systems and backup of an enormous amount of data, we labeled the data of the period in which update was executed over six hours as anomaly data and the data before the update as normal data. Hyperparameters λ_0, α_L, and α_H are set to 0.1, 2, and 5, respectively.

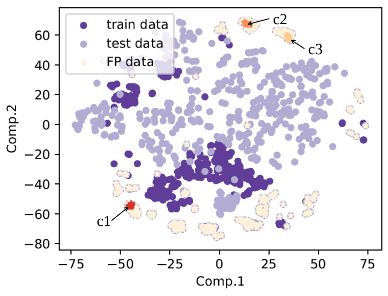
5.2.3 Evaluation Results

We first evaluated the effect on reducing FPs. Figure 6 compares the MSE transitions with three algorithms: an algorithm without retraining (*conventional*), the proposed algorithm that only executes RET_{inc} without EWC penalty term (*inc_only*), and that executes both RET_{inc} and RET_{out} (*inc_and_out*). The number of FPs and average AUROC are listed in Table 3, in which the algorithm of Tharkan and Toshniwal [22] is also compared. The comparison between *inc_only* and *inc_and_out* indicates that RET_{out} reduces the number of FPs caused by *normal outliers* by about 79% compared to the algorithm using only RET_{inc}, which is a simple sliding window method, without degrading detection accuracy for the validation data.

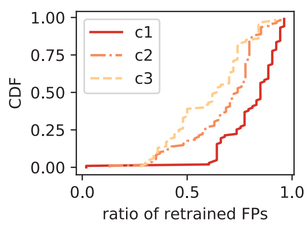
5.2.4 Evaluation on effectiveness in reducing number of similar FPs

We further examined if RET_{out} actually reduces similar FPs. We first classified the *normal outliers* by clustering (Fig. 7a), and for representative three clusters, we confirmed how many FPs caused by similar *normal outliers* can be reduced in one retraining (Fig. 7b) and if the effect of retraining can be maintained throughout the retraining (Fig. 7c). The classification of the *normal outliers* was executed as follows. We first compressed the high-dimensional data into 2-dimensions by using t-distributed stochastic neighbor embedding (t-SNE) [13] then classified the data by using DBSCAN [22]. This approach is promising for clustering high-dimensional data under a situation in which the number of clusters is unknown [12]. Note that t-SNE is an offline algorithm; therefore, cannot be applied for online anomaly detection. Figure 7a shows the 2-dimensional embedding of the data by using t-SNE. For extracting *normal outliers*, we distinguished the data with which the MSE was over $\gamma(\alpha_H)$ in the algorithm with only RET_{inc} (center of Fig. 6) for eliminating *incremental drift* and classified using DBSCAN with $minPts = 3$ and $\epsilon = 0.3$. After that, we chose the largest 3 clusters as the target of the analysis.

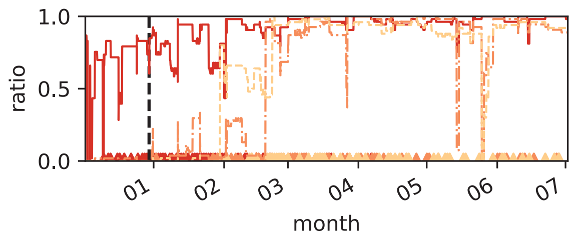
Table 4 describes each cluster. To characterize each cluster, we confirmed the highest feature value in absolute value of the deviation from the distribution of the training data. Although all the clusters were the data in the backup process, the characteristics of each cluster differed. We first evaluated how many similar FPs can be reduced by one retraining. We retrained the original model with one *normal outlier* in a cluster, then evaluated the amount of data with which the MSE was less than $\gamma(\alpha_{ano})$ with the retrained model, that is, the number of reduced FPs. To purely evaluate the effectiveness of retraining the model for reducing the number of FPs, we did not update preprocessing parameters and thresholds.



(a) t-SNE of input data



(b) Distributions of reduced FP ratios in same cluster



(c) Ratios of reduced FPs throughout retraining. Bottom triangles show the times the FP occured.

Figure 7: Evaluation of the effectiveness of reducing similar FPs by RET_{out}.

Table 4: Description of each cluster. Right two columns show mean values over the data in the cluster.

cluster (size)	characteristics	MSE (deviation)	reduced FPs in same / other cluster(s)
c1 (53)	high disk usage	13.5	42.6/31.1
c2 (54)	conflict in CPU	35.6	36.9/36.2
c3 (50)	high packet receiving rate	19.9	30.5/85.2

Therefore, this evaluation was conservative compared to that of the proposed algorithm. The evaluation was conducted for all the data in each cluster, and the distributions of the reduced FP ratios is depicted in Fig. 7b (mean values are described in Table 4). The figure shows that all the data in c1 reduced the number of FPs in the same cluster by more than about 60%. However, that ratio of c2 and c3 is less than c1. It can be said that the effectiveness of retraining varies among the clusters, that is, the types of anomalies. As shown in Table 4, the retraining not only reduced the number of FPs in the same cluster but also that in other clusters. This is because the retraining increases the region of the normal data boundary and will include other FPs in the previous boundary. This is one of the strengths of the proposed algorithm compared to the conventional clustering algorithm discussed in Section 2.

Figure 7c shows the transition in the ratio of the number of FPs reduced for each cluster to confirm if a new retraining does not degrade the previous training. Each time in the online retraining phase, the number of FPs reduced in each cluster was evaluated with the algorithm. From the figure, we confirm that the ratio of the number of FPs reduced was generally maintained throughout the retraining, that is, our algorithm retrains *incremental drift* and *normal outliers* without degrading the previous trainings.

6. CONCLUSION

We proposed an online anomaly detection algorithm for retraining *normal outliers*, which cause FPs and have been major problems in practical use, as normal by using EWC with autoencoders. The evaluation results showed that our proposed algorithm could retrain the *normal outliers* as normal so that decreasing similar FPs and the total number of FPs due to the *normal outliers* is reduced by about 79% compared to a simple sliding window algorithm with real measured data. Determining the adequate hyperparameters and the threshold, and retraining of FNs should be further discussed.

7. ACKNOWLEDGMENTS

The authors would like to thank to Satoshi Imai, Takayuki Nakao, and Taichiro Sugiyama for many helpful discussions.

8. REFERENCES

[1] KDD Cup 1999 Data. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. Last accessed 15 May 2018.

[2] NSL-KDD dataset. http://www.unb.ca/cic/datasets/index.html. Last accessed 15 May 2018.

[3] D. B. Araya, K. Grolinger, H. F. ElYamany, M. A. Capretz, and G. Bitsuamlak. Collective contextual anomaly detection framework for smart buildings. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 511–518. IEEE, 2016.

[4] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[5] Y. Dong and N. Japkowicz. Threaded ensembles of supervised and unsupervised neural networks for stream learning. In *Canadian Conference on Artificial Intelligence*, pages 304–315. Springer, 2016.

[6] R. M. French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[7] Y. Ikeda, K. Ishibashi, Y. Nakano, K. Watanabe, and R. Kawahara. Inferring causal parameters of anomalies detected by autoencoder using sparse optimization. In *IEICE Tech. Rep. IN2017-18*, volume 117, pages 61–66. IEICE, June 2017. (in japanese).

[8] Y. Ikeda, K. Ishibashi, Y. Nakano, K. Watanabe, and R. Kawahara. Retraining of anomaly detection model using autoencoder. In *IEICE Tech. Rep. IN2017-84*, volume 117, pages 77–82. IEICE, Jan. 2018. (in japanese).

[9] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *ArXiv e-prints*, Dec. 2014.

[10] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, page 201611835, 2017.

[11] T. Lane and C. E. Brodley. Approaches to online learning and

concept drift for user identification in computer security. In *KDD*, pages 259–263, 1998.

[12] P. Lewin, L. Petrov, and L. Hao. A feature based method for partial discharge source classification. In *Electrical Insulation (ISEI), Conference Record of the 2012 IEEE International Symposium on*, pages 443–448. IEEE, 2012.

[13] L. v. d. Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[14] M. M. Masud, J. Gao, L. Khan, J. Han, and B. Thuraisingham. Integrating novel class detection with classification for concept-drifting data streams. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 79–94. Springer, 2009.

[15] M. M. Masud, C. Woolam, J. Gao, L. Khan, J. Han, K. W. Hamlen, and N. C. Oza. Facing the reality of data stream classification: coping with scarcity of labeled data. *Knowledge and information systems*, 33(1):213–244, 2012.

[16] S. Sadik and L. Gruenwald. Research issues in outlier detection for data streams. *ACM SIGKDD Explorations Newsletter*, 15(1):33–40, 2014.

[17] M. Sakurada and T. Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, page 4. ACM, 2014.

[18] A. Siffer, P.-A. Fouque, A. Termier, and C. Largouet. Anomaly detection in streams with extreme value theory. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1067–1075. ACM, 2017.

[19] E. J. Spinosa, F. de Leon, A. Ponce, and J. Gama. Novelty detection with application to data streams. *Intelligent Data Analysis*, 13(3):405–422, 2009.

[20] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[21] J. Sun, Y. Chai, M. Guo, and H. Li. On-line adaptive principal component extraction algorithms using iteration approach. *Signal Processing*, 92(4):1044–1068, 2012.

[22] Y. Thakran and D. Toshniwal. Unsupervised outlier detection in streaming data using weighted clustering. In *Intelligent Systems Design and Applications (ISDA), 2012 12th International Conference on*, pages 947–952. IEEE, 2012.

[23] B. B. Thompson, R. J. Marks, J. J. Choi, M. A. El-Sharkawi, M.-Y. Huang, and C. Bunje. Implicit learning in autoencoder novelty assessment. In *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, volume 3, pages 2878–2883. IEEE, 2002.

[24] S. Tokui, K. Oono, S. Hido, and J. Clayton. Chainer: a next-generation open source framework for deep learning. In *Proceedings of workshop on machine learning systems (LearningSys) in the twenty-ninth annual conference on neural information processing systems (NIPS)*, volume 5, 2015.

[25] S. Xie and S. Krishnan. Dynamic principal component analysis with nonoverlapping moving window and its applications to epileptic eeg classification. *The Scientific World Journal*, 2014, 2014.

[26] C. Zhou and R. C. Paffenroth. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 665–674. ACM, 2017.