mathematics

MDPI

Article

Robust Intrusion Detection for Industrial Control Systems Using Improved Autoencoder and Bayesian Gaussian Mixture Model

Chao Wang [1,2], Hongri Liu [1,3], Chao Li [3], Yunxiao Sun [1,2], Wenting Wang [4] and Bailing Wang [1,2,*]

[1] School of Computer Science and Technology, Harbin Institute of Technology, Weihai 264209, China
[2] School of Cyber Science and Technology, Harbin Institute of Technology, Harbin 150001, China
[3] Weihai Cyberguard Technologies Co., Ltd., Weihai 264209, China
[4] State Grid Shandong Electric Power Company, Electric Power Research Institute, Jinan 250003, China
* Correspondence: wbl@hit.edu.cn

Abstract: Machine learning-based intrusion detection systems are an effective way to cope with the increasing security threats faced by industrial control systems. Considering that it is hard and expensive to obtain attack data, it is more reasonable to develop a model trained with only normal data. However, both high-dimensional data and the presence of outliers in the training set result in efficiency degradation. In this research, we present a hybrid intrusion detection method to overcome these two problems. First, we created an improved autoencoder that incorporates the deep support vector data description (Deep SVDD) loss into the training of the autoencoder. Under the combination of Deep SVDD loss and reconstruction loss, the novel autoencoder learns a more compact latent representation from high-dimensional data. The density-based spatial clustering of applications with noise algorithm is then used to remove potential outliers in the training data. Finally, a Bayesian Gaussian mixture model is used to identify anomalies. It learns the distribution of the filtered training data and uses the probabilities to classify normal and anomalous samples. We conducted a series of experiments on two intrusion detection datasets to assess performance. The proposed model performs better than other baseline methods when dealing with high-dimensional and contaminated data.

Keywords: industrial control systems; network security; intrusion detection; anomaly detection; autoencoder

MSC: 68M25; 68T07

Citation: Chao, W.; Liu, H.; Li, C.; Sun, Y.; Wang, W.; Wang, B. Robust Intrusion Detection for Industrial Control Systems Using Improved Autoencoder and Bayesian Gaussian Mixture Model. *Mathematics* **2023**, *11*, 2048. https://doi.org/10.3390/math11092048

Academic Editors: Daniel Ramotsoela, Adnan M. Abu-Mahfouz, Bruno Silva, Umair Mujtaba Qureshi and Zuneera Umair

Received: 28 March 2023
Revised: 16 April 2023
Accepted: 23 April 2023
Published: 26 April 2023

1. Introduction

An industrial control system (ICS) is a general term that encompasses several types of control systems [1], such as supervisory control and data acquisition systems, and distributed control systems. Typically, these systems are employed in industries such as power plants, water, and wastewater facilities. With advancements in information technology, ICSs have been connected to the internet for increased efficiency. However, this has exposed ICSs to numerous security threats [2]. Attacks on ICSs can disrupt their normal operations, resulting in financial losses and potentially affecting the daily lives of humans. In recent years, several attacks against ICSs have been reported [3–5]. Therefore, it is crucial and urgent to develop security solutions to safeguard ICSs, and intrusion detection systems (IDSs) play a significant role in this regard [2].

Based on the detection technique used, the IDS can be divided into a misuse-based detection method or anomaly-based detection method [2,6]. A misuse-based detection method compares established attack patterns to new events to evaluate if the new events are attacks or not. This approach has a greater detection rate, but it is incapable of detecting unknown threats. The anomaly-based technique detects anomalies by constructing a

normal profile from ICS data. Anomalies are defined as events that deviate from the profile. Anomaly-based detection methods can be divided into three categories [6]: statistics-based, knowledge-based, and machine learning-based. In this paper, we focus on IDS based on machine learning techniques.

To address the detection of anomalies using labeled datasets, some supervised machine learning techniques have been developed [7,8], such as random forest and decision tree. However, gathering attack data to create a labeled dataset is difficult and expensive [9]. As a result, training the detection model with only normal data makes sense. One frequently employed approach is the one-class support vector machine (OCSVM) [10,11]. However, these methods are affected by the curse of dimensionality, which could reduce the detection rate. Before training the detection model, some researchers suggested using the autoencoder (AE) to reduce the dimension first [9,12]. By rebuilding the input as much as possible, the AE learns a compressed latent representation. Furthermore, AE could detect anomalies by using the reconstruction error, which has a higher value for anomalous samples [13].

In addition to the issue of high-dimensional data, the presence of outliers in the training set also has a negative impact on performance. When outliers are included, models such as OCSVM [14] or AE [15] may also learn the distribution of outliers, which degrades performance. Motivated by the work of the deep support vector data description (Deep SVDD) neural network [16], which transforms data into a hypersphere where outliers are outside, an improved autoencoder (IAE) [17] was proposed. By incorporating the Deep SVDD loss during the training phase of AE, the IAE learns a compact representation. However, utilizing the distance calculated in the latent space makes it challenging to identify some outliers that may not be entirely excluded from the hypersphere. The Deep SVDD neural network can apply to the samples directly. But this approach suffers from the problem of hypersphere collapse [16].

In this paper, we propose a hybrid anomaly detection method to address the problems of high-dimensional data and the existence of outliers in the training set. In general, the novel model introduces two steps before anomaly detection: feature extraction and outlier removal. Motivated by the work [9,18], which uses the AE to extract features first, we apply the IAE to obtain latent representations. To further decrease the effect of outliers within the training set, we employ the density-based spatial clustering of applications with noise (DBSCAN) algorithm to remove some potential outliers. Finally, the distribution of the filtered dataset is modeled using a Bayesian Gaussian mixture model (BGMM). The following is a summary of the contributions:

- With the aim of extracting useful features from high-dimensional data, IAE is used to produce more compact latent representations. In detail, the Deep SVDD loss is combined with the reconstruction loss during the training of the IAE. The new latent representation improves the effectiveness of the anomaly detection model.
- With the improved latent representation, we first utilize DBSCAN to remove the potential outliers within the training set in order to decrease its impact on detection performance. BGMM is trained using the filtered training set, which uses probability to identify abnormalities. The outlier removal step benefits for the BGMM and improves detection performance.
- To show the effectiveness of the suggested methodology, we run a series of experiments on two intrusion detection datasets. Our method demonstrates more promising detection performance than some other baselines in terms of effectiveness and robustness. Additionally, an ablation study proves the improvement provided by both steps mentioned above.

The remainder of the paper is organized as follows. In Section 2, we review related works involving anomaly detection. We provide a full introduction to the proposed methods in Section 3. Then, the effectiveness of the suggested approach is discussed in Section 4. Finally, the relevant conclusions are presented in Section 5, together with directions for future research.

2. Related Work

In recent years, ICS security issues have received a lot of attention. Attacks against ICSs, such as Stuxnet [19] and BlackEnergy3 [20], can cause devastating consequences when compared to traditional information systems. IDS plays an important role in providing security. In particular, with the development of machine learning and deep learning, many research studies [8] have been proposed. Supervised machine learning methods can be used to classify the normal data and attacks when there are labeled datasets. However, gathering attack data is challenging [9], in contrast, collecting normal data is simple. This paper primarily reviews anomaly detection methods that use normal data only for training. In an anomaly-based intrusion detection method, the model is trained using normal data in order to learn a normal profile. Samples will be classified as abnormal if they differ from the established profile.

As a one-classification method, OCSVM is frequently used in the field of anomaly detection. It finds a hyperplane that separates normal data from the origin with a maximum margin. In [10], an improved OCSVM was developed. Modbus traffic packets were used to train the OCSVM model to generate a normal communication pattern, and a particle swarm optimization method was used to optimize the parameters of OCSVM. Ensemble learning and OCSVM were combined to create the intrusion detection technique named IT-OCSVM [11]. The authors combined the predictions of several OCSVM models using mean majority voting. The performance of detection using the suggested method is promising. However, OCSVM suffers from the issue of high-dimensional data [16].

Many deep learning-based anomaly detection works have been proposed [21–23]. One way to apply deep learning is by using it as a feature extraction module [9,24]. AE is a special kind of deep learning model that works to reconstruct the input as accurately as possible. To extract discriminative features for IDS, a stacked sparse AE is used [25]. Experiments have shown that it is effective in feature extraction, and it alleviates the performance degradation caused by high-dimensional data [9,12]. The reconstruction error can also be utilized as the anomaly score to detect anomalies directly. By utilizing the stacked denoising AE [26], malicious packets within ICSs can be detected.

These approaches, however, work under the presumption that the training set contains only normal data. However, it is hard to maintain in a practice environment. The anomaly detection model may also learn the anomalous samples well when the training dataset is contaminated by outlier samples. The performance of anomaly detection can be affected as a result, as shown in [17]. For example, in AE-based methods, the reconstruction error of anomalous samples may decrease to a level similar to that of normal data as the number of training epochs increases [15]. The Deep SVDD neural network [16] aims to learn a mapping that transforms normal data into a hypersphere while minimizing the volume of the hypersphere. Anomalies are identified as samples that are located outside of this hypersphere. However, the Deep SVDD neural network suffers from the problem of hypersphere collapse [16]. By adding a constraint supervised by the Deep SVDD loss for the latent space, Zhen Chen et al. [17] proposed IAE, which aims to position the anomalous samples outside the hypersphere in the latent space. As for the anomaly score, it uses the distance between the center and each sample in the latent space. Distance may not be a useful metric to distinguish between normal and anomalous data as some outliers might be inside the hypersphere. However, this method is effective for learning latent representations. In general, these two methods try to create a novel loss function for the neural network while considering the existence of outliers.

To reduce the effect of anomalous samples in the training dataset, ref. [15] proposed the use of an adversarial AE. They introduced an outlier labeling procedure during training using OCSVM. The identified outlier would not affect the training of normal samples. However, the adoption of the OCSVM may cause an increase in training time.

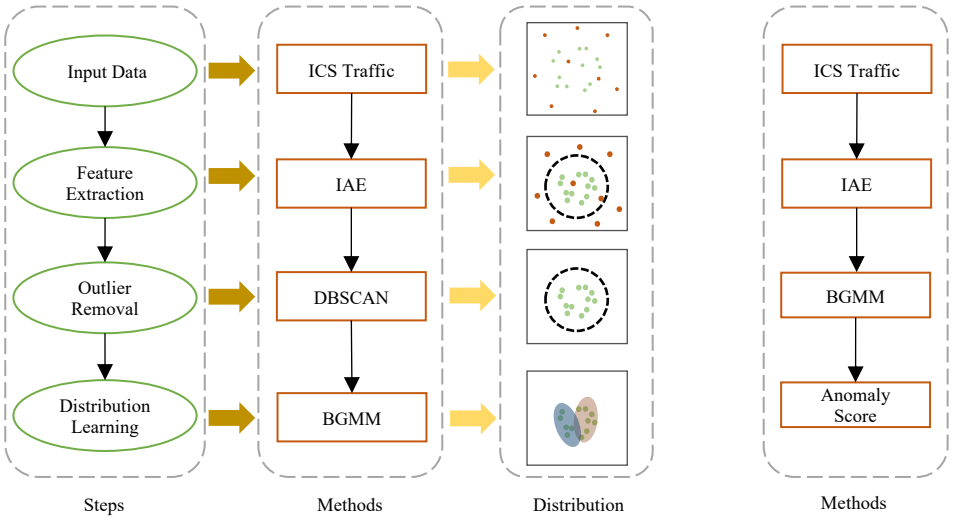
Based on the studies mentioned above, there are two views on dealing with high-dimensional and contaminated datasets. The first is the powerful capability of deep learning for feature extraction, which facilitates anomaly detection through representation learning. The other is an operation to minimize the influence of outliers in the training set. Our proposed method takes into account both of these factors.

3. Method

In this section, we present the proposed IDS model. We begin by introducing the entire detection framework. The IAE is then presented. In the end, a robust anomaly detection approach is developed using the latent representations derived from the IAE.

3.1. Overview

Before delving into the details of each part, we provide an overview of the proposed method, which is depicted in Figure 1. In general, there are two phases, training and testing, as shown in the figure.



(**a**) Training process. (**b**) Testing process.

Figure 1. The overview of our proposed method.

There are three steps during the training phase. Given the ICS traffic features, we developed an IAE, which derives the compressed representations of original features. The combination of reconstruction loss and Deep SVDD loss is used to train the IAE. We used DBSCAN to remove the potential outlier points because their presence would negatively affect the performance of the anomaly detection model. Finally, a BGMM was trained to learn the data distribution using the filtered training set.

We applied the learned model to identify the anomalies in the testing set after training. Unlike the training process, the BGMM inspects the latent representation of the testing dataset directly during testing. The BGMM generates higher anomaly scores for anomalous samples compared to normal data. Each component of the proposed methodology for anomaly detection is thoroughly described in the subsequent subsections.

3.2. Improved Autoencoder

Compared with the traditional AE, the IAE incorporates the Deep SVDD as part of the loss function to learn a compact data representation with minimized volume. The AE and Deep SVDD are described further below, and the IAE is introduced at the end.

3.2.1. Autoencoder

The AE is a special type of neural network. The input data go through several hidden layers in the AE. It aims at reconstructing the input as much as possible. It has been applied in the field of feature extraction widely [27].

In detail, AE can be divided into two parts: the encoder and the decoder. Considering a set of input data $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}$, where N is the number of samples in the dataset, the encoder learns a map function ϕ for each sample $\boldsymbol{x}_i$ and outputs compressed latent representation $\boldsymbol{z}_i$. The decoder then attempts to reconstruct the input data from the compressed latent representation, $\boldsymbol{z}_i$, and outputs the reconstructed data, $\hat{\boldsymbol{x}}_i$. The training process of the AE aims to find the parameters that minimize the reconstruction loss. In this study, the mean squared error is used to calculate the reconstruction loss, which is given in (1).

$$L_{rec} = \|\hat{\boldsymbol{x}}_i - \boldsymbol{x}_i\|^2 \tag{1}$$

When employing an AE to extract features from a dataset, the decoder is abandoned after training, only the encoder is maintained.

3.2.2. Improved Autoencoder

Support vector data description (SVDD) [28], like OCSVM, is a one-class classification method. When carrying out tasks for anomaly detection, the SVDD seeks to find the smallest hypersphere that encloses most of the normal samples. Lukas Ruff et al. [16] proposed the Deep SVDD neural network, which combines the SVDD with the neural network. The Deep SVDD introduces the SVDD as the loss function for the training of the neural network. The trained neural network learns a mapping function, and within the output space of the mapping function, most normal samples are enclosed by a hypersphere.

Let $\Phi(\cdot; \boldsymbol{W})$ be the transform function of the neural network, with $\boldsymbol{W}$ denoting the set of hidden layer weights. The $\Phi(\boldsymbol{x}_i; \boldsymbol{W})$ is the output of the neural network for sample $\boldsymbol{x}_i$. The loss function of the Deep SVDD neural network is defined in (2).

$$L_{ds} = R^2 + \frac{1}{\nu N} \sum_{i=1}^{N} \max\{0, \|\Phi(\boldsymbol{x}_i; \boldsymbol{W}) - \boldsymbol{c}\|^2 - R^2\} \tag{2}$$

The first term refers to the volume of the hypersphere with radius R. The second term is a penalty for points whose distance from the center $\boldsymbol{c}$ is greater than R. The hyperparameter ν controls the trade-off between the volume of the hypersphere and the number of points outside it. Deep SVDD neural networks aim to map normal samples into a hypersphere with minimal volume based on the loss function. When using Deep SVDD for anomaly detection, the distances to the center $\boldsymbol{c}$ can be calculated as anomaly scores.

But the Deep SVDD neural network suffers from the problem of hypersphere collapse [16]. When this happens, the output of the neural network for every input is constant, and the hypersphere radius R collapses to zero. However, we can use it as a regularizer to constrain the latent space of AE [17]. In this way, the encoder produces compact latent representations with minimized volume. Combining the two loss functions, the architecture of IAE is displayed in Figure 2. As seen in the figure, the Deep SVDD loss is applied to the output of the encoder only. Considering the map function ϕ of the encoder, we can rewrite the joint loss function

$$L = L_{rec} + \lambda L_{ds} = \|\hat{\boldsymbol{x}}_i - \boldsymbol{x}_i\|^2 + \lambda\{R^2 + \frac{1}{\nu N} \sum_{i=1}^{N} \max\{0, \|\phi(\boldsymbol{x}_i; \boldsymbol{W}) - \boldsymbol{c}\|^2 - R^2\}\} \tag{3}$$

where λ is the weight controlling the relationship between reconstruction loss and Deep SVDD loss. It is worth noting that $\boldsymbol{W}$ represents the layer weights within the encoder. Next, we describe the training algorithm in more detail, as shown in Algorithm 1. We start with k warm-up epochs during the training phase, where the AE is trained solely on the reconstruction loss. This allows the encoder to generate a stable latent representation. After that, the center value, $\boldsymbol{c}$, is calculated as the mean value of the encoder output for all training data. The IAE is then trained with both Deep SVDD loss and reconstruction loss simultaneously. During training, we update the parameters of the neural network and update R. After calculating the distances $\boldsymbol{S}$ between the encoder output and the center $\boldsymbol{c}$, we calculate the new R using the distance $\boldsymbol{S}$, following the approach in the original paper [16].

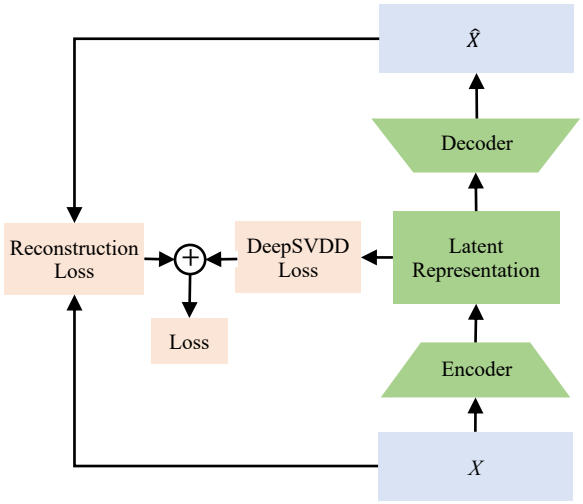


Figure 2. The framework of IAE.

Algorithm 1: Training process of IAE.

Data: Training data $\boldsymbol{x}_i$. Initialized parameters $\boldsymbol{\theta}$ of IAE. The number of warm epochs k, hyperparameter λ, hyperparameter ν, and the number of iterations t.

Result: The Parameters $\boldsymbol{\theta}$.

`// Training the IAE using reconstruction loss only`

for $i = 1, 2, \ldots, k$ **do**
  Compute the reconstruction loss by (1)
  Update the parameters $\boldsymbol{\theta}$ in the IAE by the backpropagation algorithm
end
Compute the center $\boldsymbol{c}$ by the mean of the output of the encoder for all training samples

`// Training the IAE using joint loss`

$t \leftarrow 0$;
while not converge, **do**
  $t \leftarrow t + 1$;
  Compute the joint loss by (3)
  Update the parameters $\boldsymbol{\theta}$ in the IAE by the backpropagation algorithm.
  Compute the distance $\boldsymbol{S}$ of all samples in the training set.
  Update the R by distance $\boldsymbol{S}$ and hyperparameter ν.
end

The latent representations are obtained once the IAE has been trained. We created an anomaly detection method based on the latent representation in the next subsection.

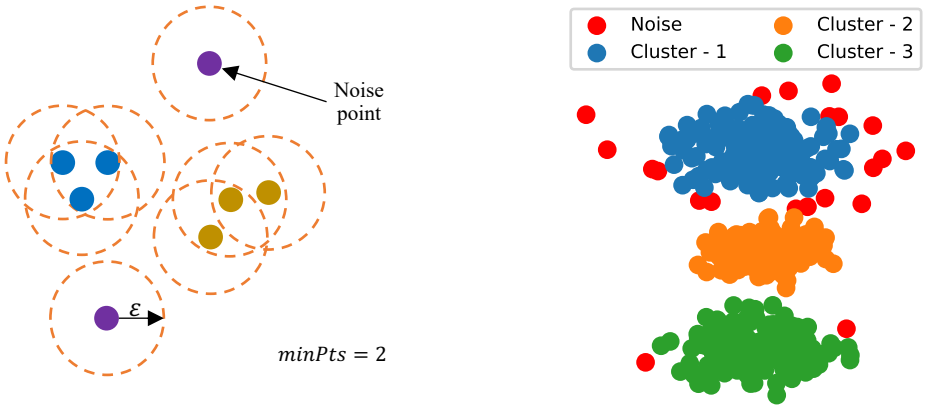
3.3. Anomaly Detection Based on DBSCAN and BGMM

In the previous section, we used the IAE to learn a compact representation of data. The distance within the latent space can be used to detect anomalies [17], but its performance is not satisfactory. Considering the presence of outlier points in the training set, we adopt a hybrid method for anomaly detection. First, we remove the outliers from the training set using DBSCAN. Next, we use a BGMM model to fit the remaining training data. In the following subsections, we provide a detailed explanation of our method.

3.3.1. Density-Based Spatial Clustering of Applications with Noise

DBSCAN is a clustering method [29,30]. Its intuition is to find the high-density regions in the sample space. The radius, ϵ, and the number of neighbors, $minPts$, are two important pre-defined parameters. Within the radius, if the neighbors of one observation are higher than the $minPts$ (including the observation itself), the regions can be classified as high-density regions. In the meantime, some points within the low-density regions would be considered outliers.

In detail, with the DBSCAN algorithm, samples in the dataset can be classified into three types of points: core points, border points, and outlier points. Core points are those with $minPts$ neighbors within a radius of ϵ. The neighbors of core points belong to the same cluster of core points, and they are density-reachable. The points within the cluster that are not core points are called border points. Outlier or noise points are those that are not density-reachable by any core points. An example can be seen in Figure 3a, where there are eight points in a simple example, and two clusters can be concluded. In the meantime, two noise points are recognized. To illustrate the DBSCAN algorithm further, we create a synthetic dataset with three clusters. The clustering results are displayed in Figure 3b. Some points with a larger distance from other points have been classified as noise.



(**a**) (**b**)

Figure 3. Illustration of the DBSCAN algorithm. (**a**) Example of DBSCAN with $minPts = 2$; (**b**) clustering results by DBSCAN on the synthetic dataset.

Presuming that the normal ICS data locate in higher density regions than outliers, we can employ DBSCAN to filter the outlier points first. With the latent representations $\boldsymbol{Z} = \{z_1, z_2, \ldots, z_N\}$, we can use the DBSCAN algorithm to label these observations as normal or noise. We denote the label results using $Y = \{y_1, y_2, \ldots, y_N\}$, where $y_i \in \{0, 1\}$. The points are given "label 1" if they are outlier points and "0" if they are core or boundary points. We compute the neighbors of each point and identify core points using the pre-defined parameters $minPts$ and ϵ. The neighbors of core points and core points themselves are then connected to form clusters. Finally, a point is considered noise if it does not neighbor any core points.

The two parameters in the algorithm significantly affect the clustering results. As a rule of thumb, the $minPts$ can be set as $2 * d$ [29], where d is the dimension of input

data. However, in our study, we conducted experiments on a dataset consisting of tens of thousands of samples. It is reasonable to set it at a higher value. In this study, the *minPts* are set at a certain percentage of the number of training sets. Then, using the *k*-distance graph, where $k = minPts$, we select the value of epsilon [30]. We calculate the *k*-nearest neighbor distance for each observation and then sort the distances from lowest to highest. After that, we compute the "elbow" point in the *k*-distance graph using the Kneedle algorithm [31], and we utilize this value as ϵ.

3.3.2. Bayesian Gaussian Mixture Model

We first filter the training set using the label results Y, keeping the observations z_i with $y_i = 0$. After that, the BGMM is used to learn the distribution for the cleaned training dataset. Gaussian mixture model (GMM) is a probabilistic model that is commonly used for clustering. In this study, we use it to model the normal data and detect anomalous samples with probability density. We lay out the foundations of GMM first.

GMM attempts to fit its probability density using a linear combination of K Gaussian distributions (also known as components) with the observation $\boldsymbol{z}$ extracted from the IAE. Its equation can be written as

$$p(\boldsymbol{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu_k}, \boldsymbol{\Lambda_k^{-1}}) \tag{4}$$

where the $\mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu_k}, \boldsymbol{\Lambda_k^{-1}})$ represents the k-th components with the parameter $\boldsymbol{\mu_k}, \boldsymbol{\Lambda_k}$. The $\boldsymbol{\mu_k}$ and $\boldsymbol{\Lambda_k}$ are the mean and precision matrices, respectively, and parameter π_k is the mixing coefficient, which satisfies $\sum_k \pi_k = 1$ and $0 \leq \pi_k \leq 1$. Let $\boldsymbol{v}$ denote the K-dimensional latent variable corresponding to $\boldsymbol{z}$. The latent variable $\boldsymbol{v}$ is a binary random variable satisfying $v_k \in \{0, 1\}$ and $\sum_k v_k = 1$. Taking the mixing coefficients into account, the distribution of $\boldsymbol{v}$ can be written as

$$p(\boldsymbol{v}) = \prod_{k=1}^{K} \pi_k^{v_k} \tag{5}$$

Let the filtered dataset be denoted by $\boldsymbol{Z} = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \cdots, \boldsymbol{z}_N\}$ where N is the number of samples in the filtered dataset, and let all latent variables be denoted by $\boldsymbol{V}$; considering the parameters and latent variables, the conditional distribution of observations is given by

$$p(\boldsymbol{Z}|\boldsymbol{V}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = \prod_{n=1}^{N} \prod_{k=1}^{K} \mathcal{N}(\boldsymbol{z}_n|\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1})^{v_{nk}} \tag{6}$$

In the maximization likelihood framework, we can use the expectation–maximization (EM) algorithm to estimate the parameters. However, there are some limitations to using the EM algorithm to estimate the parameters of the GMM. One issue is that it suffers from the presence of singularities [32]. In this study, we adapt the Bayesian version of GMM and use variational inference to estimate the parameters of the GMM. This approach treats all of the parameters as random variables and assigns prior distributions to them, using conjugate prior distributions to simplify calculations. The Dirichlet distribution and Gaussian–Wishart prior are introduced for $\boldsymbol{\pi}$, $\boldsymbol{\mu}_k$, and $\boldsymbol{\Lambda}_k$, respectively, as shown in

$$p(\boldsymbol{\pi}) = \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}_0) \tag{7}$$

and

$$p(\boldsymbol{\mu}, \boldsymbol{\Lambda}) = p(\boldsymbol{\mu}|\boldsymbol{\Lambda}) p(\boldsymbol{\Lambda}) = \prod_{k=1}^{K} \mathcal{N}(\boldsymbol{\mu}_k|\boldsymbol{m}_0, (\beta_0 \boldsymbol{\Lambda}_k)^{-1}) \mathcal{W}(\boldsymbol{\Lambda}_k|\boldsymbol{a}_0, b_0) \tag{8}$$

where $\mathcal{W}$ is the Wishart distribution, and $\{\boldsymbol{\alpha_0}, \boldsymbol{m}_0, \beta_0, \boldsymbol{a}_0, b_0\}$ are the parameters of the prior distribution. In the Dirichlet distribution, we use the same parameter, α_0, for each component. The goal of variational inference for BGMM is to find an approximation of the

posterior distribution $p(\boldsymbol{V}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}|\boldsymbol{Z})$. To simplify, we denote $\boldsymbol{\Theta}$ as $\{\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}\}$. We refer to the [32]; the log marginal likelihood can be decomposed by

$$\ln p(\boldsymbol{Z}) = \mathcal{L}(q) + \mathrm{KL}(q\|p) \tag{9}$$

where the functional $\mathcal{L}(q)$ is the lower bound of $\ln p(\boldsymbol{Z})$. Moreover, the $\mathrm{KL}(q\|p)$ is the Kullback–Leibler (KL) divergence between the variational posterior distribution $q(\boldsymbol{V}, \boldsymbol{\Theta})$, and the true posterior $p(\boldsymbol{V}, \boldsymbol{\Theta}|\boldsymbol{Z})$. These equations are defined as

$$\mathcal{L}(q) = \sum_{V} \int q(\boldsymbol{V}, \boldsymbol{\Theta}) \ln\{\frac{p(\boldsymbol{Z}, \boldsymbol{V}, \boldsymbol{\Theta})}{q(\boldsymbol{V}, \boldsymbol{\Theta})}\} d\boldsymbol{\Theta} \tag{10}$$

$$\mathrm{KL}(q\|p) = -\sum_{V} \int q(\boldsymbol{V}, \boldsymbol{\Theta}) \ln\{\frac{p(\boldsymbol{V}, \boldsymbol{\Theta}|\boldsymbol{Z})}{q(\boldsymbol{V}, \boldsymbol{\Theta})}\} d\boldsymbol{\Theta} \tag{11}$$

We attempt to maximize the $\mathcal{L}(q)$ with respect to the $q(\boldsymbol{V}, \boldsymbol{\Theta})$, which is equivalent to minimizing the $\mathrm{KL}(q\|p)$. When the $q(\boldsymbol{V}, \boldsymbol{\Theta})$ is approximated as the true posterior distribution $p(\boldsymbol{V}, \boldsymbol{\Theta}|\boldsymbol{Z})$, we have the maximum of the lower bound [32]. With some necessary calculations (for detailed calculations, readers can refer to [32]), the approximate posterior distribution is shown below. We list the detailed variables in the equation in the Appendix A.

$$q^{\star}(\boldsymbol{V}) = \prod_{n=1}^{N} \prod_{k=1}^{K} r_{nk}^{v_{nk}} \tag{12}$$

where r_{nk} plays the role of responsibilities. Moreover, the optimization of $\boldsymbol{\pi}$ and $(\boldsymbol{\mu_k}, \boldsymbol{\Lambda}_k)$ are calculated as below.

$$q^{\star}(\boldsymbol{\pi}) = \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}) \tag{13}$$

$$q^{\star}(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k) = \mathcal{N}(\boldsymbol{\mu}_k|\boldsymbol{m}_k, (\beta_k \boldsymbol{\Lambda}_k)^{-1}) \mathcal{W}(\boldsymbol{\Lambda}_k|\boldsymbol{a}_k, b_k) \tag{14}$$

With the optimal distribution analysis above, we can estimate the parameters by following the steps in the EM algorithm. After the parameters in the prior distribution are initialized, we calculate the (12) in the E-step. Then in the M-step, we keep the responsibilities fixed and use these values to update the distribution in (13) and (14). By updating the parameters while cycling between the two steps until convergence, we can estimate the parameters of BGMM.

3.3.3. Anomaly Detection

We introduce the whole training algorithm in Algorithm 2 by combining the DBSCAN and BGMM. We first filter the noise points in the latent representation $\boldsymbol{z}_i$ of the training set. We use the Kneedle algorithm to calculate the value of ϵ using the given parameters $minPts$. The training set is then labeled by DBSCAN, and the labeled results indicate whether the samples are considered noise points. To train the BGMM model, we take samples z_i from the training set where their labeled results are $y_i = 0$ by DBSCAN. With some necessary parameters in the prior distribution initialized, we begin the E-step. The M-step is then carried out to update the parameters. The E-step and M-step are repeated before the convergence. Finally, we have a well-trained BGMM model.

In the testing phase, we first extract latent representations by IAE for the test samples. Then, they are inspected by the well-trained BGMM. The anomaly score is finally calculated using the negative log-likelihood produced by the BGMM, where the anomalous samples have a higher value than the normal samples.

Algorithm 2: Training process of the combination of DBSCAN and BGMM.

Data: Training features z_i. The hyperparameter $minPts$ for DBSCAN. The number of components K, initialized parameters $\{\boldsymbol{\alpha_0}, \boldsymbol{m}_0, \beta_0, \boldsymbol{a}_0, b_0\}$ in BGMM.
Result: Well-trained BGMM
`// Part 1: Outlier removal by DBSCAN`
Compute neighbors of each point in $\boldsymbol{Z}$ and sort the k-nearest neighbor distance where $k = minPts$.
Choose the value of hyperparameter ϵ by the Kneedle algorithm.
Output the label results Y for the training set by DBSCAN with $minPts$ and ϵ.
Filter the training set and keep z_i, whose $y_i = 0$.
`// Step 2:Training BGMM by a variational version of the EM algorithm.`
while not converge, **do**
  `// E-Step`
  Calculation the responsibilities by (12)
  `// M-Step`
  Update the parameters by (13) and (14)
end

4. Evaluation

In this section, we evaluate the performance of the proposed method. The datasets utilized in the experiments are presented first. After that, the comparative methods that serve as the baselines are described. Finally, we introduce the experimental setup and analyze the performance of the proposed method.

4.1. Dataset

In this study, we conducted the experiments using two datasets: the Gas Pipeline dataset [33] and NF-BoT-IoT-v2 dataset [34], in order to assess the performance of the proposed method.

The first dataset, Gas Pipeline [33], was collected using a laboratory-scale gas pipeline environment. Some attacks have been used to simulate anomalous samples. There are six types of attacks, NMRI, CMRI, MSCI, MPCI, DoS, and Reconnaissance. In total, there are 61,156 normal samples and 35,863 anomalous samples. Each sample involves one captured network transaction pair, including network traffic features and payload content features. For every observation, there are 26 numeric features.

The second dataset, NF-BoT-IoT-v2 [34], was created by the Cyber Range Lab of the Australian Center for Cyber Security using Ostinato and Node-RED tools. Apart from normal samples, there are four attacks: DoS, DDoS, Reconnaissance, and Theft. Since there are a large number of records, we randomly sample the records in the dataset, just as in our previous work [35]. There are 65,150 normal samples and 78,598 anomalous samples after sampling. Every record in the dataset is a NetFlow-based feature. Except for some category features, we apply the log function to numeric features to decrease the effect of large feature differences. Additionally, the category features are handled using the one-hot encoder method. There are roughly 200 features left after preprocessing.

Every observation in both datasets is scaled using the min–max normalization into the range of [0, 1]. We divided the dataset into a training set and a testing set with a 4:1 ratio. In order to simulate the contaminated dataset, we remove all anomalous data from the training set. After that, we randomly sample data from the removed anomalous data with a specific ratio (referring to the anomaly ratio) with respect to the remaining training set. The pure training set is then combined with the chosen anomalous samples to form a new training set. It should be noted that we trained the model without any help from the label during training. We used the testing set directly, as it already contained both normal and anomalous data.

4.2. Compared Baselines

We compare the performance of our method against seven baselines, including four traditional machine learning methods and three deep learning-based methods, as listed below.

- OCSVM. In the experiments, we used the radial basis function kernel as the kernel function. We selected ν from $\{0.01, 0.1\}$, and γ from $\{1 \times 10^{-5}, 1 \times 10^{-4}, 1 \times 10^{-3}, 1 \times 10^{-2}, 1 \times 10^{-1}, 1\}$. The best results are reported.
- Isolation forest (IForest). We set the hyperparameters, the samples, and the number of estimators as the original paper recommended, which were 256 and 200, respectively.
- Kernel density estimation (KDE). The kernel function also uses the radial basis function kernel. The bandwidth was set as 0.001.
- BGMM. The BGMM was used to show the influence of the feature extraction and outlier removal steps. Its hyperparameters were set to be the same as our proposed method.
- AE. The reconstruction error was used as the anomaly score. The network architecture and hyperparameters of the AE are identical to those in our method, except for Deep SVDD loss.
- Deep SVDD. As in the original paper, we trained an AE first, and the encoder was used to train the Deep SVDD neural network. During training, the value of ν was set to 0.1.
- IAE. IAE trains the data with reconstruction loss and Deep SVDD loss. The distance between samples and the center in the latent space is used as the anomaly score. The IAE uses the same settings as our method.

For comparison, we directly used the trained IAE model in our method, which means that both the IAE and our method use the same latent representations. Therefore, we can demonstrate that the BGMM can produce a better anomaly score than the distance calculated from the IAE.

4.3. Experiment Settings

The suggested method in this study was implemented using the Python programming language. The PyTorch framework was used to program the IAE and other deep learning methods. All neural network methods employed Adam [36] as their optimizer with a learning rate of 0.001. The batch size was set to 256, and the epochs to 200. With the aim of a warm start, the first 20 epochs were trained with reconstruction loss only. A weight of 1×10^{-5} was used for the Deep SVDD loss. The value of ν in the Deep SVDD loss function was set to 0.1.

For hidden layer settings in the IAE, we used a five-layer hidden layer architecture. The architecture used by the encoder and decoder is symmetrical. For the NF-BoT-IoT-v2 dataset, we set "144-80-16-80-144" in the hidden layer. Moreover, the Gas Pipeline dataset was "20-14-8-14-20". We used PReLU [37] as the activation function in the hidden layer. The activation function in the output layer was a sigmoid function.

We implemented DBSCAN, BGMM, and other traditional machine learning methods using scikit-learn [38]. In terms of the *minPts* in the DBSCAN, we observe that larger values may decrease the performance because they lead to more outliers being labeled as normal. As a result, we set the value as a fraction of the training set chosen from $\{1\%, 5\%\}$ and report the best results. Since BGMM could choose the optimal components by itself, with the components that provide insufficient contributions removed [32], we fixed the number of components to 10, and the number of training iterations to 100.

We used the area under the receiver operating characteristic curve (AUROC), which is frequently employed in the task of anomaly detection, as the evaluation metric. To guarantee accurate results and prevent randomness during the training phase, we repeated each approach five times.

4.4. Results

We examine the effectiveness of the suggested approach in the following content of this section. We first contrast it with various baselines. Additionally, several anomaly ratios are utilized to demonstrate their robustness in handling contaminated datasets. To demonstrate the effects of the different components of our method, we conducted thorough ablation research. Finally, we examined the impact of the settings for the hyperparameter.

4.4.1. Detection Performance

To begin the performance analysis, we report the results of two lower anomaly ratios, 5% and 10%. The total performance results, which include the mean value and standard deviation of the AUROC for each approach, are shown in Table 1.

Table 1. AUROC comparisons with different baseline methods for two datasets. There are eight methods, including our proposed method in the first column. Both mean values and the standard deviation of the AUROC are reported.

Method	Gas Pipeline		NF-BoT-IoT-v2	
	5%	10%	5%	10%
OCSVM	86.89 ± 0.26	86.42 ± 0.06	60.02 ± 1.39	58.57 ± 0.92
KDE	87.04 ± 0.28	74.27 ± 0.30	86.40 ± 0.11	83.83 ± 0.14
IForest	88.51 ± 0.44	87.89 ± 0.50	85.53 ± 0.46	81.80 ± 0.56
BGMM	89.65 ± 1.33	79.75 ± 5.22	77.17 ± 5.06	72.16 ± 5.03
AE	83.55 ± 6.17	73.84 ± 8.51	84.91 ± 2.45	82.80 ± 4.11
Deep SVDD [16]	66.24 ± 5.61	64.19 ± 4.67	70.51 ± 6.69	63.20 ± 4.25
IAE [17]	76.90 ± 4.81	78.03 ± 2.18	77.33 ± 4.88	80.39 ± 3.69
Ours	$\mathbf{89.89 \pm 2.96}$	$\mathbf{88.04 \pm 0.91}$	$\mathbf{92.00 \pm 0.73}$	$\mathbf{91.73 \pm 1.06}$

Bold font indicates best results.

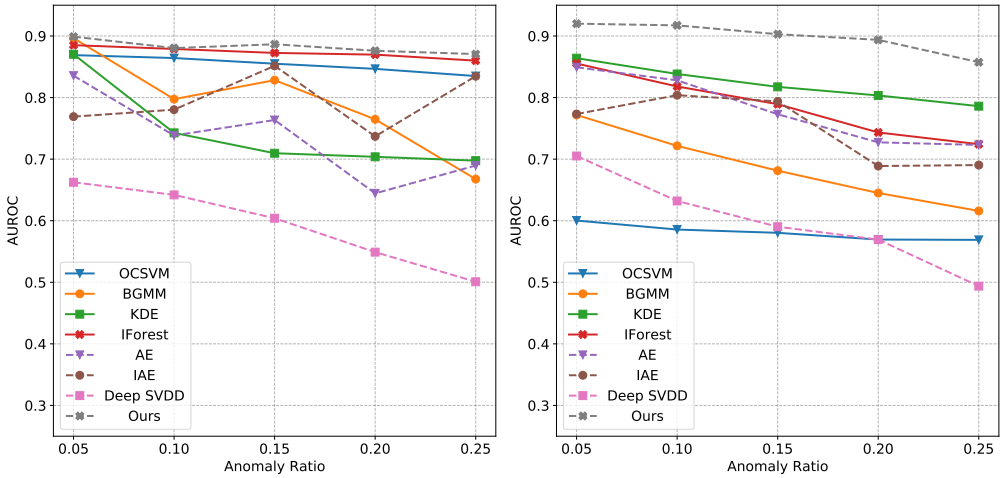
The detection performance of our method shows the greatest performance for both datasets, as can be seen from the table above. The majority of methods show a decreasing trend when the anomaly ratio rises from 5% to 10%. First, we analyze the results for the Gas Pipeline dataset. The four traditional machine learning methods outperform the remaining three deep learning baselines for this dataset. When the anomaly ratio is 5%, BGMM produces the best results out of these four methods. Despite an increase in the anomaly ratio to 10%, OCSVM and IForest maintain greater values. IAE performs well for both anomaly ratios, and it shows a higher performance compared to AE when the anomaly ratio is 10%. However, Deep SVDD performs poorly compared to other methods.

When it comes to the NF-BoT-IoT-v2 dataset, both OCSVM and BGMM perform worse than they do in the Gas Pipeline dataset, which may be due to the curse of dimensionality. AE continues to perform well in terms of both anomaly ratios. The performance of Deep SVDD is also the lowest in this dataset. In this dataset, KDE and IForest both maintain higher values. However, compared to KDE and IForest, our method shows greater improvement.

We conducted more experiments with more varied anomaly ratios to further demonstrate the robustness of the proposed method. The performances for both datasets are shown in Figure 4, with the anomaly ratio increasing from 5% to 25% with a step of 5%.

Since it can be difficult to eliminate some anomalous samples with higher anomaly ratios, most techniques generally exhibit decreasing trends for both datasets. The IAE and AE display different patterns in Figure 4a. IAE performs better than AE in general. However, compared to other promising baselines, such as IForest and OCSVM, our approaches have not shown significant improvement. For the NF-BoT-IoT-v2 dataset, the experimental results are displayed in Figure 4b. In most cases, OCSVM produces the worst results. Our method exhibits a greater improvement than the best baseline method, KDE.

In conclusion, our proposed detection method exhibits robustness and offers promising detection results with a contaminated dataset, compared with other baseline methods.



(**a**) Gas pipeline. (**b**) NF-BoT-IoT-v2

Figure 4. AUROC performance under different anomaly ratios for both datasets. There are eight methods in both figures. We report the mean values of the methods only.

4.4.2. Ablation Study

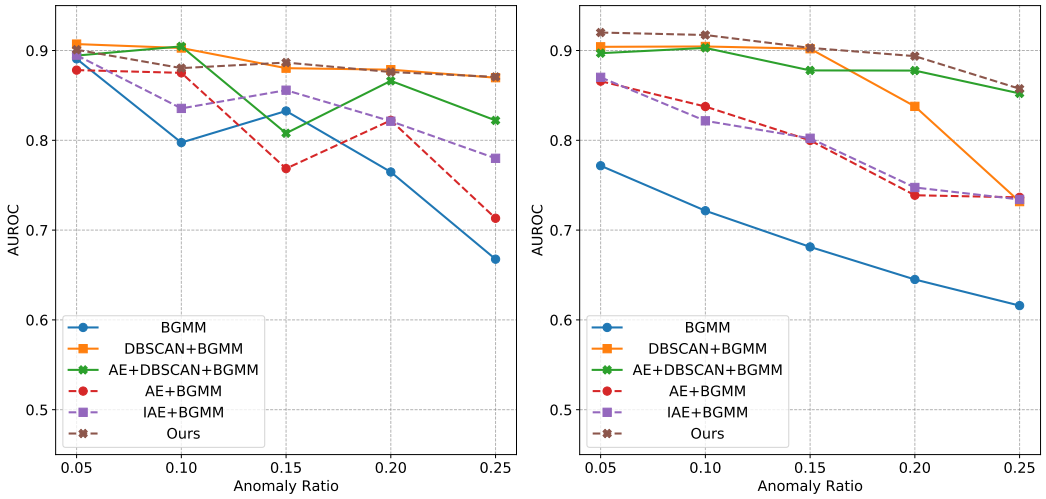
As previously stated, we improved the approach from two aspects. One is the compact latent data representations generated by the IAE, and the other is the outlier removal step by DBSCAN. In this part, we perform an ablation study to demonstrate the effects of these two parts. There are five variants, depending on whether or not IAE or DBSCAN is used, including methods that use original features or AE to extract features. The difference between each method can be inferred from the name. The detailed comparisons for the two datasets are shown in Figure 5, where the x-axis is the anomaly ratio used to exhibit the robustness for dealing with contaminated datasets.

For the Gas Pipeline dataset in Figure 5a, our approach exhibits comparable detection performance to the method employing the original features with DBSCAN. Since the dimension of this dataset is not particularly high, employing the original features can also produce good results. Additionally, the introduction of DBSCAN demonstrates improvement; for instance, our methods maintain higher accuracy than the "IAE + BGMM" method for all anomaly ratios.

In the next part, we analyze the NF-BoT-IoT-v2 dataset with higher dimensions and more detail. Figure 5b shows a higher difference between different methods than the results of the Gas Pipeline dataset. Starting with the simplest method, BGMM, which learns the distribution directly, it yields the poorest outcomes compared to the other approaches. When adding AE or IAE into the feature extraction step, both methods have an increment of 10% over BGMM only under different anomaly ratios. These two methods have similar results as their curves almost entirely overlap. However, since there are outliers in the training set, both methods suffer from performance degradation.

Both methods utilizing AE and IAE show improvement when the DBSCAN module is introduced. The effects of IAE are also starting to show. As can be seen from the figure, our methods produce better results than those utilizing AE alone because the outliers have been removed and IAE has more compact representations than AE. However, the improvement is lower when the anomaly ratio is higher. It should be noted that the methods using DBSCAN on the original features produce promising results when the anomaly ratio is lower, indicating the importance of the outlier removal step. However, when the anomaly ratio rises to 20% or 25%, the results of this method become worse. With a higher anomaly ratio, it is more difficult to remove outliers from high-dimensional data. Compared to the "DBSCAN + BGMM" method, our method has two advantages with the introduction of IAE: it has a more representative distribution and reduces computational costs by reducing dimensions.

In conclusion, the IAE and DBSCAN modules both enhance the performance of anomaly detection, especially with the higher-dimensional dataset.



(**a**) Gas pipeline. (**b**) NF-BoT-IoT-v2

Figure 5. AUROC comparisons between our method and corresponding variants of our proposed method.

4.4.3. Analysis of Hyperparameters

In the proposed method, we automatically set some hyperparameters, such as the radius of DBSCAN or the number of components in BGMM. However, the *minPts* of DBSCAN is determined empirically. As previously stated, this value is very important for finding outliers. Therefore, we analyze this parameter using different anomaly ratios.

When *minPts* is set to a value that is too high, most samples will be merged into one cluster. As we compute the ϵ using the Kneedle algorithm, its value will increase with the increase of *minPts* in most cases. Some outliers may be included in clusters with bigger *minPts* and ϵ and are incorrectly labeled as normal. We set the value of *minPts* as a proportion of the number of the training dataset, ranging from 0.5% to 5% with the 0.5% step. For simplicity, we only run tests on the NF-BoT-IoT-v2 dataset. A comparison of the results is shown in Figure 6.

Overall, the AUROC shows a decreasing trend as the anomaly ratio or the number of *minPts* increases. The AUROC remains greater than 0.80 for all alternative values of *minPts*, with an anomaly ratio of no more than 15%. When the *minPts* parameter is set to 0.5% or 1%, the maximum value for AUROC is obtained. However, we observed that DBSCAN incorrectly labeled some normal samples as noise, which indeed degraded the performance. This suggests that we can further improve the detection rate with optimized parameters for DBSCAN, and we leave this as a direction for future work.

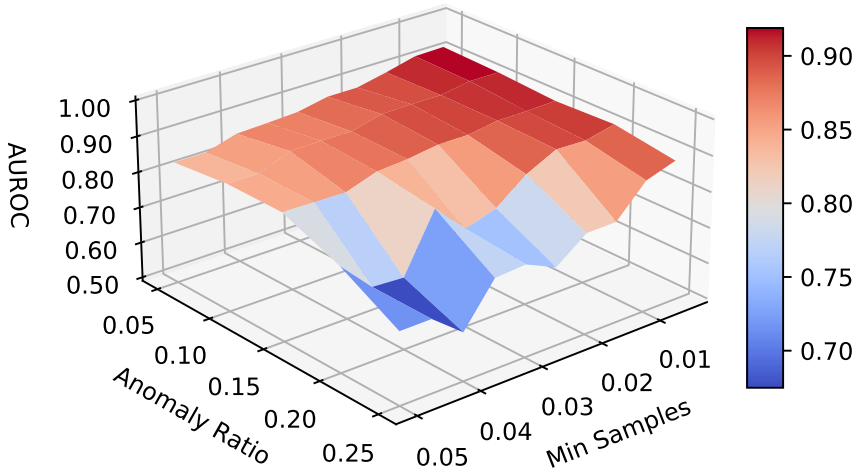


Figure 6. Effect of the *minPts* hyperparameter on the detection performance under different anomaly ratios for the NF-BoT-IoT-v2 dataset.

5. Conclusions and Future Work

Due to the increasing cyber security threats to ICSs, it is critical to develop efficient security solutions. Anomaly detection-based IDS are essential tools for ensuring security. However, the high-dimensional data and presence of outliers in the training set pose challenges to detection accuracy. In this study, we propose a hybrid intrusion detection model to address these issues.

The proposed method improves the effectiveness of anomaly detection in two aspects. The IAE in the first part derives data representations from ICS traffic features. In the second part, before training the anomaly detection model, we introduce an outlier removal step. In more detail, we provide IAE that incorporates the Deep SVDD loss function during training. Together, the additional loss and reconstruction loss enable the IAE to learn a more compact latent representation. Based on the improved latent representation, the DBSCAN method is employed to first remove the outlier from the training set. This process is important because it reduces the influence of outliers. The BGMM eventually learns the probability distribution for the training set. The results of the experiments conducted on two intrusion datasets reveal that the proposed method outperforms other baseline methods in terms of the detection rate and robustness to outliers.

There are numerous possibilities for the future direction of our work. Despite the removal of outliers from the training set by DBSCAN, as previously mentioned, it also incorrectly classified some normal data, negatively impacting the performance because some normal data could not be learned. Using an advanced optimization method to select parameters for DBSCAN would be important to reduce the number of false positives. Additionally, by using measurements such as the reconstruction error or distance during the training phase of the AE, we could directly remove some outliers with greater confidence. In this way, during training, we could reduce the effect of outliers on the latent representation of normal data. Finally, the normal data could be modeled using a more advanced model than the BGMM.

Author Contributions: Conceptualization, C.W. and Y.S.; data curation, C.W. and W.W.; formal analysis, H.L., C.L. and Y.S.; funding acquisition, B.W.; investigation, C.L. and W.W.; methodology, C.W., H.L. and Y.S.; project administration, B.W.; resources, H.L., Y.S. and W.W.; software, C.W., C.L. and Y.S.; supervision, H.L. and B.W.; validation, H.L. and C.L.; visualization, C.W.; Writing—original draft, C.W.; writing—review and editing, H.L. and B.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key Research and Development Program of China (no. 2021YFB2012400).

Data Availability Statement: Two datasets were used in this research: Gas Pipeline and NF-BoT-IoT-v2. Readers who are interested in our research can access the datasets from the corresponding papers and reproduce our results.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

In the Expectation step, to calculate the equation of (12), some necessary variables are listed below.

$$r_{nk} = \frac{\rho_{nk}}{\sum_{j=1}^{K} \rho_{nj}} \tag{A1}$$

where

$$\ln \rho_{nk} = \mathbb{E}(\ln \pi_k) + \frac{1}{2}\mathbb{E}(\ln |\mathbf{\Lambda}_k|) - \frac{D}{2}\ln(2\pi) - \frac{1}{2}\mathbb{E}_{\boldsymbol{\mu}_k,\mathbf{\Lambda}_k}((\boldsymbol{z}_n - \boldsymbol{\mu}_k)^T \mathbf{\Lambda}_k (\boldsymbol{z}_n - \boldsymbol{\mu}_k)) \tag{A2}$$

$$\mathbb{E}[\ln |\mathbf{\Lambda}_k|] = \sum_{i=1}^{D} \psi(\frac{b_k + 1 - i}{2}) + D\ln 2 + \ln |\boldsymbol{a}_k| \tag{A3}$$

$$\mathbb{E}_{\boldsymbol{\mu}_k,\mathbf{\Lambda}_k}((\boldsymbol{z}_n - \boldsymbol{\mu}_k)^T \mathbf{\Lambda}_k (\boldsymbol{z}_n - \boldsymbol{\mu}_k)) = D\beta_k^{-1} + b_k(\boldsymbol{z}_n - \boldsymbol{m}_k)^T \boldsymbol{a}_k (\boldsymbol{z}_n - \boldsymbol{m}_k) \tag{A4}$$

$$\mathbb{E}(\ln \pi_k) = \psi(\alpha_k) - \psi(\sum_k \alpha_k) \tag{A5}$$

In (A2), D is the dimension of input data. In (A3), the $\psi(\cdot)$ is the digamma function.From the responsibilities r_{nk}, we can define some statistics,

$$N_k = \sum_{n=1}^{N} r_{nk} \tag{A6}$$

$$\bar{\boldsymbol{z}}_k = \frac{1}{N_k}\sum_{n=1}^{N} r_{nk}\boldsymbol{z}_n \tag{A7}$$

$$\boldsymbol{S}_k = \frac{1}{N_k}\sum_{n=1}^{N} r_{nk}(\boldsymbol{z}_n - \bar{\boldsymbol{z}}_k)(\boldsymbol{z}_n - \bar{\boldsymbol{z}}_k)^T \tag{A8}$$

In the Maximization step, the parameters in the prior distribution are updated by the following equations.

$$\alpha_k = \alpha_0 + N_k \tag{A9}$$

$$\beta_k = \beta_0 + N_k \tag{A10}$$

$$\boldsymbol{m}_k = \frac{1}{\beta_k}(\beta_0 \boldsymbol{m}_0 + N_k \bar{\boldsymbol{z}}_k) \tag{A11}$$

$$\boldsymbol{a}_k^{-1} = \boldsymbol{a}_0^{-1} + N_k \boldsymbol{S}_k + \frac{\beta_0 N_k}{\beta_0 + N_k}(\bar{\boldsymbol{z}}_k - \boldsymbol{m}_0)(\bar{\boldsymbol{z}}_k - \boldsymbol{m}_0)^T \tag{A12}$$

$$b_k = b_0 + N_k \tag{A13}$$

References

1. Stouffer, K.; Pillitteri, V.; Lightman, S.; Abrams, M.; Hahn, A. Guide to Industrial Control Systems (ICS) Security NIST Special Publication 800-82 Revision 2. In *NIST Special Publication 800-82 Rev 2*; 2015 . Available online: https://csrc.nist.gov/publications/detail/sp/800-82/rev-2/final (accessed on 25 April 2023).

2. Kaouk, M.; Flaus, J.M.; Potet, M.L.; Groz, R. A review of intrusion detection systems for industrial control systems. In Proceedings of the 2019 6th International Conference on Control, Decision and Information Technologies, CoDIT, Paris, France, 23–26 April 2019; pp. 1699–1704. [CrossRef]
3. Anton, S.D.; Fraunholz, D.; Lipps, C.; Pohl, F.; Zimmermann, M.; Schotten, H.D. Two decades of SCADA exploitation: A brief history. In Proceedings of the 2017 IEEE Conference on Applications, Information and Network Security, AINS, Miri, Malaysia, 13–14 November 2017; pp. 98–104. [CrossRef]
4. Alladi, T.; Chamola, V.; Zeadally, S. Industrial Control Systems: Cyberattack trends and countermeasures. *Comput. Commun.* **2020**, *155*, 1–8. [CrossRef]
5. Hemsley, K.E.; Fisher, R.E. History of Industrial Control System Cyber Incidents. In *INL/CON-18-44411-Revision-2*; 2018. Avaiable online: https://www.osti.gov/servlets/purl/1505628/ (accessed on 25 April 2023)
6. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity* **2019**, *2*, 20. [CrossRef]
7. Mokhtari, S.; Abbaspour, A.; Yen, K.K.; Sargolzaei, A. A machine learning approach for anomaly detection in industrial control systems based on measurement data. *Electronics* **2021**, *10*, 407. [CrossRef]
8. Lopez Perez, R.; Adamsky, F.; Soua, R.; Engel, T. Machine Learning for Reliable Network Attack Detection in SCADA Systems. In *Proceedings—17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications and 12th IEEE International Conference on Big Data Science and Engineering, Trustcom/BigDataSE 2018*; IEEE: New York, NY, USA, 2018; pp. 633–638. [CrossRef]
9. Cao, V.L.; Nicolau, M.; McDermott, J. Learning Neural Representations for Network Anomaly Detection. *IEEE Trans. Cybern.* **2019**, *49*, 3074–3087. [CrossRef]
10. Shang, W.; Li, L.; Wan, M.; Zeng, P. Industrial communication intrusion detection algorithm based on improved one-class SVM. In Proceedings of the 2015 World Congress on Industrial Control Systems Security, WCICSS, London, UK, 14–16 December 2015; pp. 21–25. [CrossRef]
11. Maglaras, L.A.; Jiang, J.; Cruz, T.J. Combining ensemble methods and social network metrics for improving accuracy of OCSVM on intrusion detection in SCADA systems. *J. Inf. Secur. Appl.* **2016**, *30*, 15–26. [CrossRef]
12. Qi, R.; Rasband, C.; Zheng, J.; Longoria, R. Detecting cyber attacks in smart grids using semi-supervised anomaly detection and deep representation learning. *Information* **2021**, *12*, 328. [CrossRef]
13. Kim, S.J.; Jo, W.Y.; Shon, T. APAD: Autoencoder-based Payload Anomaly Detection for industrial IoE. *Appl. Soft Comput. J.* **2020**, *88*, 106017. [CrossRef]
14. Xiao, Y.; Wang, H.; Xu, W.; Zhou, J. Robust one-class SVM for fault detection. *Chemom. Intell. Lab. Syst.* **2016**, *151*, 15–25. [CrossRef]
15. Beggel, L.; Pfeiffer, M.; Bischl, B. *Robust Anomaly Detection in Images Using Adversarial Autoencoders*; Springer International Publishing: Cham, Switzerland, 2020; Volume 11906, pp. 206–222. [CrossRef]
16. Ruff, L.; Vandermeulen, R.; Goernitz, N.; Deecke, L.; Siddiqui, S.A.; Binder, A.; Müller, E.; Kloft, M. Deep One-Class Classification. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; Volume 80, pp. 4393–4402.
17. Cheng, Z.; Wang, S.; Zhang, P.; Wang, S.; Liu, X.; Zhu, E. Improved autoencoder for unsupervised anomaly detection. *Int. J. Intell. Syst.* **2021**, *36*, 7103–7125. [CrossRef]
18. Cao, V.L.; Nicolau, M.; McDermott, J. A hybrid autoencoder and density estimation model for anomaly detection. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Cham, Switzerlannd, 2016; Volume 9921, pp. 717–726. [CrossRef]
19. Langner, R. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Secur. Priv.* **2011**, *9*, 49–51. [CrossRef]
20. E-ISAC. *Analysis of the Cyber Attack on the Ukrainian Power Grid*; Technical Report; 2016. Available online: https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2016/05/20081514/E-ISAC_SANS_Ukraine_DUC_5.pdf (accessed on 25 April 2023)
21. Alabugin, S.K.; Sokolov, A.N. Applying of Generative Adversarial Networks for Anomaly Detection in Industrial Control Systems. In Proceedingsof the 2020 Global Smart Industry Conference, GloSIC, Chelyabinsk, Russia, 17–19 November 2020; pp. 199–203. [CrossRef]
22. Filonov, P.; Kitashov, F.; Lavrentyev, A. Rnn-based early cyber-attack detection for the tennessee eastman process. *arXiv* **2017**, arXiv:1709.02232.
23. Goh, J.; Adepu, S.; Tan, M.; Lee, Z.S. Anomaly detection in cyber physical systems using recurrent neural networks. In Proceedings of the IEEE International Symposium on High Assurance Systems Engineering, Singapore, 12–14 January 2017; pp. 140–145. [CrossRef]
24. Pang, G.; Shen, C.; Cao, L.; Hengel, A.V.D. Deep Learning for Anomaly Detection: A Review. *ACM Comput. Surv.* **2021**, *54*, 1–38. [CrossRef]
25. Yan, B.; Han, G. Effective Feature Extraction via Stacked Sparse Autoencoder to Improve Intrusion Detection System. *IEEE Access* **2018**, *6*, 41238–41248. [CrossRef]

26. Schneider, P.; Böttinger, K. High-performance unsupervised anomaly detection for cyber-physical system networks. In Proceedings of the ACM Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 1–12. [CrossRef]
27. Ghojogh, B.; Samad, M.N.; Mashhadi, S.A.; Kapoor, T.; Ali, W.; Karray, F.; Crowley, M. Feature Selection and Feature Extraction in Pattern Analysis: A Literature Review. *arXiv* **2019**, arXiv:1905.02845.
28. Tax, D.M.; Duin, R.P. Support vector data description. In *Machine Learning*; Springer: Cham, Switzerlannd, 2004; Volume 54, pp. 45–66. [CrossRef]
29. Ester, M.; Kriegel, H.P.; Sander, J.; Xu, X. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, KDD'96, Portland, OR, USA, 2–4 August 1996; pp. 226–231.
30. Schubert, E.; Sander, J.; Ester, M.; Kriegel, H.P.; Xu, X. DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN. *ACM Trans. Database Syst.* **2017**, *42*, 1–21. [CrossRef]
31. Satopää, V.; Albrecht, J.; Irwin, D.; Raghavan, B. Finding a "kneedle" in a haystack: Detecting knee points in system behavior. In Proceedings of the International Conference on Distributed Computing Systems, Minneapolis, MN, USA, 20 June–24 June 2011; pp. 166–171. [CrossRef]
32. Bishop, C.M.; Nasrabadi, N.M. *Pattern Recognition and Machine Learning*; Springer: New York, NY, USA, 2006; Volume 4.
33. Morris, T.; Gao, W. Industrial control system traffic data sets for intrusion detection research. In *International Conference on Critical Infrastructure Protectio*n; Springer: Berlin/Heidelberg, Germany, 2014. [CrossRef]
34. Sarhan, M.; Layeghy, S.; Portmann, M. Towards a Standard Feature Set for Network Intrusion Detection System Datasets. *Mob. Netw. Appl.* **2021**, *27*, 357–370. [CrossRef]
35. Wang, C.; Wang, B.; Sun, Y.; Wei, Y.; Wang, K.; Zhang, H.; Liu, H. Intrusion Detection for Industrial Control Systems Based on Open Set Artificial Neural Network. *Secur. Commun. Netw.* **2021**, *2021*, 4027900. [CrossRef]
36. Kingma, D.P.; Ba, J.L. Adam: A method for stochastic optimization. In Proceedings of the 3rd International Conference on Learning Representations, ICLR—Conference Track Proceedings, San Diego, CA, USA, 7–9 May 2015; pp. 1–15.
37. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE international Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1026–1034.
38. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.