electronics

MDPI

Article

Intrusion Detection System Based on One-Class Support Vector Machine and Gaussian Mixture Model

Chao Wang [1,2], Yunxiao Sun [1,2], Sicai Lv [1,2], Chonghua Wang [3], Hongri Liu [1,4] 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] China Industrial Control Systems Cyber Emergency Response Team, Beijing 100040, China
[4] Weihai Cyberguard Technologies Co., Ltd., Weihai 264209, China
* Correspondence: wbl@hit.edu.cn

Abstract: Intrusion detection systems (IDSs) play a significant role in the field of network security, dealing with the ever-increasing number of network threats. Machine learning-based IDSs have attracted a lot of interest owing to their powerful data-driven learning capabilities. However, it is challenging to train the supervised learning algorithms when there are no attack data at hand. Semi-supervised anomaly detection algorithms, which train the model with only normal data, are more suitable. In this study, we propose a novel semi-supervised anomaly detection-based IDS that leverages the capabilities of representation learning and two anomaly detectors. In detail, the autoencoder (AE) is applied to extract representative features of normal data in the first step, and then two semi-supervised detectors, the one-class support vector machine (OCSVM) and Gaussian mixture model (GMM), are trained on the derived features. The two detectors collaborate to detect anomalous samples. The OCSVM predicts the abnormal samples initially, and after that, the GMM is applied to recheck the misclassified samples further. The experiments demonstrate that the AE improves the detection rate, and two detectors are more promising than a single one.

Keywords: intrusion detection; semi-supervised anomaly detection; autoencoder; one-class support vector machine; Gaussian mixture model

Citation: Wang, C.; Sun, Y.; Lv, S.; Wang, C.; Liu, H.; Wang, B. Intrusion Detection System Based on One-Class Support Vector Machine and Gaussian Mixture Model. *Electronics* **2023**, *12*, 930. https://doi.org/10.3390/electronics12040930

Academic Editor: Wojciech Mazurczyk

Received: 20 January 2023
Revised: 4 February 2023
Accepted: 6 February 2023
Published: 13 February 2023

1. Introduction

The increasing number of network attacks pose significant threats for network environments. To enhance the network protection ability, intrusion detection systems (IDSs) have received a lot of attention. According to the detection methodologies, they can be separated into two categories: misuse detection and anomaly detection [1]. Misuse detection approaches employ the signatures of known attacks to examine new samples. They have higher detection rates, but it is hard to identify unknown attacks or the variants of known attacks. Anomaly detection based methods learn the normal profile and find abnormal samples that deviate from it [1].

Machine learning-based anomaly detection methods have been widely employed. They leverage data-driven learning approaches to learn the characteristics of network traffic. Supervised learning methods would learn the decision boundary between anomalous and normal data during the training phase, and then utilize this capability to categorize the new samples during the testing phase—for example, decision trees [2], support vector machine [3], and random forest [4]. However, for these supervised learning algorithms, they need a labeled dataset comprising both normal and abnormal samples. It is difficult and expensive to gather the labeled samples [5,6].

In the anomaly detection techniques, based to whether the dataset has labels or not, they have three types [7]: supervised methods, semi-supervised methods, and unsupervised methods. The semi-supervised methods train on the dataset that only contains normal data. Unsupervised techniques operate with the dataset without label directly. Since it is

easy to gather normal data in a network environment, we can learn about the characteristics of normal data. In this research, we focus on semi-supervised anomaly detection-based IDS.

The one-class learning approach could be employed to handle this problem, for example, one-class support vector machine (OCSVM) [8–10]. The anomalous sample is identified using a hyperplane that separates the normal and abnormal samples. However, it is challenging to cope with high-dimensional data.

There are numerous other kinds of anomaly detection approaches, for example, kernel density estimation (KDE) is employed to learn the characteristics of normal data [11]. Before training KDE, authors apply an autoencoder (AE) to extract the representative features. In this manner, it enhances the detection performance compared with applying the original feature directly. The AE itself could be applied in anomaly detection and reconstruction loss serves as anomaly score [12]. Also, an IDS based on the probabilistic model is proposed [13]. It leverages the Gaussian mixture model (GMM) to learn the probability distribution. These studies learn the features of normal data from different perspectives and generate distinct anomaly scores. The anomalous samples can be recognized if they have higher anomaly scores.

In general, while dealing with high-dimensional data, imposed by the curse of dimensionality, the performance may not be satisfactory. In the mean time, there are still improvement space for a single of detector. In this research, we present a novel semi-supervised anomaly-based IDS utilizing the OCSVM and GMM both. Compared with the supervised techniques, which use a labeled dataset comprising both normal and attack data to train the model, this model uses simply normal data. This technique utilizes the data representation capabilities of deep learning and combines the capacities of two different anomaly detectors to enhance the effectiveness of IDS. In detail, the contributions can be summarized as follows:

1. Before training the anomaly detector, we use the AE to extract representative features from network data. These features are fed into the anomaly detectors. The new features enhance detection performance.
2. After obtaining latent features, we employ them to train OCSVM and GMM further. The OCSVM learns a one-class classification boundary; in the meantime, the GMM learns a probability distribution of the normal data. In specifically, the GMM is utilized to reclassify the samples obtained via OCSVM.
3. We conduct a number of experiments on two intrusion detection datasets to illustrate their performance. The experiment results indicate that our proposed model demonstrates higher detection capability.

The rest of this study is structured as follows. We review some related work in Section 2. Then, Section 3 describes the proposed detection framework. After that, we evaluate the proposed detection method and analyze the experiment results in Section 4. In final, Section 5 draws the corresponding conclusions and points out some future work.

2. Related Work

Aside from detection techniques, IDS can be divided into two categories depending on the data source, namely, host-based IDS (HIDS) and network-based IDS (NIDS) [1]. HIDS analyzes data like logs of the operating system on a host. The NIDS is good at detecting malicious actions by analyzing network traffic transmitted inside the network. In this study, we concentrate on the NIDS.

Machine learning- and deep learning-based NIDS have been developed by a number of researchers [14], as their performance is remarkable. Among them, supervised approaches have gained a lot of attention [15–18]. But as stated previously, it requires to gather labeled datasets [5,6]. Deep learning methods, in particular, are data-hungry [19]. They require huge amounts of data to train the neural network. When there are fewer attack samples, they may confront the problem of imbalanced data [20,21]. Considering these challenges, it is possible to create a semi-supervised anomaly-based NIDS that trains the model using only normal data. Semi-supervised anomaly detection approaches try to learn the features

of normal data. Various approaches use different metrics to quantify the degree of the abnormality. When their anomaly score is high, the abnormal samples can be identified.

Considering the issue of high-dimensional data [5], numerous researchers proposed employing the AE before training anomaly detector. The AE is a special neural network that has been frequently employed in dimension reduction [22] because it can learn a compressed latent of the original input.

For example, the AE is used to extract representative features before training the semi-supervised anomaly detection algorithms for identifying cyber attacks in smart grids [23]. According to the experiment results, the semi-supervised algorithms, like OCSVM, gain improvement from the new features. As one of the ensemble learning methods, isolation forest (IForest) is employed in intrusion detection [24]. Similarly, a one-class extreme learning machine is used as anomaly method to detect abnormalities for gas turbine combustor [25]. These approaches benefit from the new features derived from the AE.

In another way, the AE itself could be utilized to identify anomalies. During training, the reconstruction error is utilized as the loss function to train the neural network. The AE attempts to reconstruct the original input as much as possible. In this method, the reconstruction error is utilized as anomaly score. The authors of [12] investigate the effectiveness of several AE variants.

Liao et al. [26] created an ensemble framework utilizing various AEs and generative adversarial networks (GAN). Traditional AE, variational AE, convolutional AE, convolutional variational AE, and GAN are all part of the proposed framework. A weighted average ensemble is used to obtain the final anomaly score from the reconstruction error produced by multiple models after each model has been trained. By comparing the anomaly score to a predefined threshold, the sample is classified as anomalous or not. The results of the experiments demonstrate that the ensemble model outperforms the single model. This research motivates us to employ several detectors to cooperatively identify anomalies.

Considering the works described above, two concerns deserve consideration during the design of IDS: one is the feature engineering required to process high-dimensional data, and the other is the application of a powerful detector. In this study, we propose a novel detection method that employs the AE and two different anomaly detectors.

3. Proposed Methods

In this section, we introduce the proposed model in detail. First, we lay out the basic elements within the detection model in sequence, including the AE, OCSVM, and GMM. After that, we merge these elements to present our whole detection framework.

3.1. AE

The AE is an unsupervised neural network; it can learn efficient representations of the input data. Figure 1 displays the architecture of AE. As the figure illustrates, the AE can be divided into two components the encoder and the decoder [27]. The latent features generated by the encoder have a smaller dimension than the input data usually.

The main objective of the AE is to reconstruct the input data as much as possible. The input data goes through hidden layers in the neural network, and the output layer outputs rebuilt data. The encoder learns a map function ϕ for sample $\boldsymbol{x}_i$ and outputs a compressed latent representation $\boldsymbol{z}_i$, given a set of input features $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$, where N is the number of samples in the dataset.

$$\boldsymbol{z}_i = \phi(\boldsymbol{x}_i) \tag{1}$$

After that, the decoder learns a mapping function ψ and tries to rebuild the input data from compressed latent representation $\boldsymbol{z}_i$. Finally, it outputs $\hat{\boldsymbol{x}}_i$, which is reconstruction of the input.

$$\hat{\boldsymbol{x}}_i = \psi(\boldsymbol{z}_i) \tag{2}$$

The training process of the AE finds the parameters in the encoder and decoder that minimize the reconstruction error. In this work, the mean squared error is utilized to calculate the reconstruction loss. The calculation is defined as

$$L = \frac{1}{N}\sum_{i=1}^{N}\|\hat{\boldsymbol{x}}_i - \boldsymbol{x}_i\|^2 = \frac{1}{N}\sum_{i=1}^{N}\|\psi(\phi(\boldsymbol{x}_i)) - x_i\|^2 \tag{3}$$

After training, we only use the trained encoder to obtain discriminative latent representation z_i. In addition, we use them to train the other anomaly detectors.

As stated before, the reconstruction error measures the degree to which the AE rebuilds the data. As a result, the reconstruction loss can be used as the anomaly score for samples to detect attacks. To decide which samples are attacks, we can use a percentile score from reconstruction loss of training set as a threshold [28]. When the reconstruction error is higher than the threshold, the sample is classified as anomalous. The method will be compared with ours as a baseline.

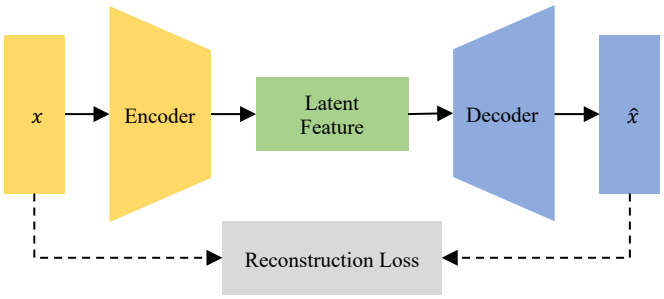


Figure 1. The architecture of an AE. The encoder compresses a original input $\boldsymbol{x}$ into a latent feature, and then the decoder reconstructs it from the latent feature.

3.2. OCSVM

OCSVM [8] is one representative method of one-class classification. The one-class classification method, unlike the binary or multi-class classification methods, only deals with one class of data [29]. The trained model determines whether a new sample belongs to the target class or not. An illustration of OCSVM is shown in Figure 2.

Consider the sample $\boldsymbol{x}_i$, a mapping function φ maps it into a high-dimensional kernel space $\mathcal{F}$. In the space $\mathcal{F}$, the inner product can be computed by some kernel function k, where the $k(\boldsymbol{x},\boldsymbol{y}) =< \varphi(\boldsymbol{x}), \varphi(\boldsymbol{y}) >$ and $<,>$ is the inner product. The main idea of OCSVM is to find a hyperplane in the kernel space $\mathcal{F}$ that separates data from the origin with maximum margin [8]. In detail, the OCSVM solves the quadratic problem as below.

$$\begin{aligned} \min \quad & \frac{1}{2}\|\mathbf{w}\|_{\mathcal{F}}^2 - \rho + \frac{1}{\nu N}\sum_{i=1}^{N}\xi_i \\ s.t. \quad & \langle \mathbf{w}, \varphi(\boldsymbol{x}_i)\rangle_{\mathcal{F}} \geq \rho - \xi_i, \quad \xi_i \geq 0 \ . \end{aligned} \tag{4}$$

where the $\mathbf{w}$ is the weight vector of the hyperplane and ρ is the margin. Nonnegative slack variables ξ_i allow some samples to cross the hyperplane and make the margin to be soft. The hyperparameter $\nu \in (0, 1]$ controls the trade-off in the objective. Also, ν is an upper bound on the fraction of anomalies. In this study, we use Gaussian kernel function $k(\boldsymbol{x},\boldsymbol{y}) = \exp(-\gamma\|\boldsymbol{x}-\boldsymbol{y}\|^2)$.

After solving the problem, for one testing sample, the decision function $f(\boldsymbol{x}_i)$ is computed.

$$f(\boldsymbol{x}_i) = \text{sgn}(\langle \mathbf{w}, \varphi(\boldsymbol{x}_i)\rangle_{\mathcal{F}} - \rho) \tag{5}$$

If new samples are on the wrong side of the hyperplane, they are recognized as anomaly points. The decision function of anomalous points will be negative.

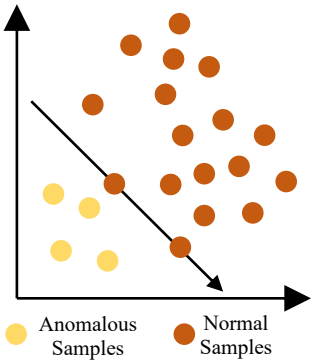
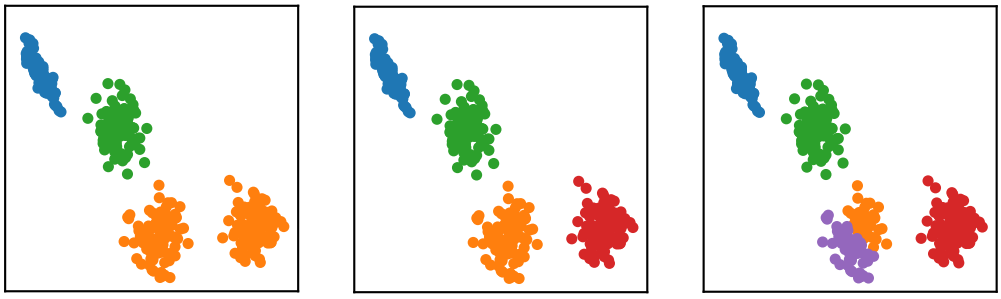


Figure 2. The illustration of OCSVM. It tries to find a hyperplane to separate the normal samples from the anomalous samples. The hyperplane, in particular, has the greatest distance to the origin.

3.3. GMM

GMM is a probabilistic model [30]. It uses a mixture of Gaussian distributions (also called components) to fit the data. It can be used to cluster the data like KMeans [31]. For illustration, we plot the GMM to clustering the data with different components in the Figure 3. There is a synthetic data with four clusters, and the number of components for GMM is set in the range of 3 to 5.



(**a**) Component 3 (**b**) Component 4 (**c**) Component 5

Figure 3. An illustration of GMM trained on the synthetic data. There are three results with different components for the data. We use different colors to mark the different cluster results of the trained GMM. (**a**) Three components. (**b**) Four components. (**c**) Five components.

In comparison to the other two component settings, the GMM fits the data best when the component is set to 4. When using GMM within the job of anomaly detection, it can output corresponding probability densities for the samples after training. As a result, the log-likelihood serves as anomaly score to detect anomalous samples, since abnormal ones have a lower log-likelihood. In the following, we introduce the theory of GMM.

Let input feature $\boldsymbol{x}$ as a D-dimensional vector, the GMM can be expressed as a linear combination of K Gaussian components, as shown below:

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{6}$$

where the $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ represents the k-th Gaussian components with the mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$. In detail, the $\boldsymbol{\mu}_k$ is a $D-$dimensional vector, and $\boldsymbol{\Sigma}_k$ is a $D \times D$ matrix. The π_k is called mixing coefficients. It satisfied the conditions:

$$0 \leq \pi_k \leq 1, \quad \sum_{k=1}^{K} \pi_k = 1 \tag{7}$$

Let define a $K-$dimensional latent binary random variables $\boldsymbol{z}$, it satisfy $z_k \in \{0,1\}$ and $\sum_k z_k = 1$. It is worth noting that only one element z_k equals 1.The distribution of $\boldsymbol{z}$ can be written as

$$p(\boldsymbol{z}) = \prod_{k=1}^{K} \pi_k^{z_k} \tag{8}$$

The following is the conditional distribution of x given the latent variable z.

$$p(\boldsymbol{x}|\boldsymbol{z}) = \prod_{k=1}^{K} \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k} \tag{9}$$

The graphical model of the GMM can be plotted when all variables are considered, as shown in Figure 4.

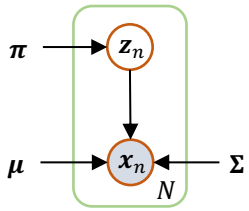


Figure 4. The graphical model of GMM.

The marginal distribution of x can be obtained from the joint distribution $p(\boldsymbol{z})p(\boldsymbol{x}|\boldsymbol{z})$ using these elements.

$$p(\boldsymbol{x}) = \sum_{\boldsymbol{z}} p(\boldsymbol{z})p(\boldsymbol{x}|\boldsymbol{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{10}$$

We also introduce the conditional probability of $\boldsymbol{z}$ given $\boldsymbol{x}$ (also known as responsibilities) because it will be useful in the following calculations. We denote the $p(z_k = 1|\boldsymbol{x})$ as $\gamma(z_k)$. Using Bayes' theorem, the calculation can be obtained as

$$\gamma(z_k) = \frac{\pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{11}$$

With the corresponding parameters, including $\boldsymbol{\pi}$, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, we can obtain the log-likelihood for the $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ as follows:

$$\ln p(\boldsymbol{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln\{\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\} \tag{12}$$

To find the parameters from which to obtain the maximum likelihood, we can use the expectation-maximization (EM) algorithm. First, some initial values for the means $\boldsymbol{\mu}$, covariances $\boldsymbol{\Sigma}$, and mixing coefficients $\boldsymbol{\pi}$ are chosen randomly. After that, we update parameters iteratively between the expectation step and the maximization step (we call them E step and M step in the following). In the E step, we can estimate the responsibilities using current parameters. After that, in the M step, using the new responsibilities, the new parameters are obtained. The new $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$, and $\boldsymbol{\pi}$ are listed:

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \boldsymbol{x}_n \tag{13}$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \gamma(z_{nk})(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T \tag{14}$$

$$\pi_k = \frac{N_k}{N} \tag{15}$$

in which N_k is defined as

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}) \tag{16}$$

With new parameters, the log-likelihood can be evaluated again. We can check the convergence of either parameters or the log-likelihood [30]. If it does not converge, we can go back to step E and re-estimate the responsibilities. In general, before the convergence, we repeat the E and M steps. Readers can refer to [30] for a detailed calculation of the EM algorithm.

After fitting the distribution using GMM, we use the negative log-likelihood as the anomaly score. We need a threshold T as the decision boundary. Usually, we can select a value from the anomaly scores of the training dataset. Let s_i represent the negative log-likelihood output by GMM $\mathcal{G}$ for sample $\boldsymbol{x}_i$, and define the decision function $d(\boldsymbol{x}_i)$ as

$$d(\boldsymbol{x}_i) = \begin{cases} 1, & \text{if } s_i \leq T \\ -1, & \text{if } s_i > T \end{cases} \tag{17}$$

When the s_i is higher than threshold T, the sample $\boldsymbol{x}_i$ is denoted as anomalous.

3.4. Whole Detection Frame

With the previously introduced theory foundation, we propose the total detection framework combining these elements in this subsection. The whole detection framework is shown in Figure 5. The whole framework can be divided into two phases: training and testing.

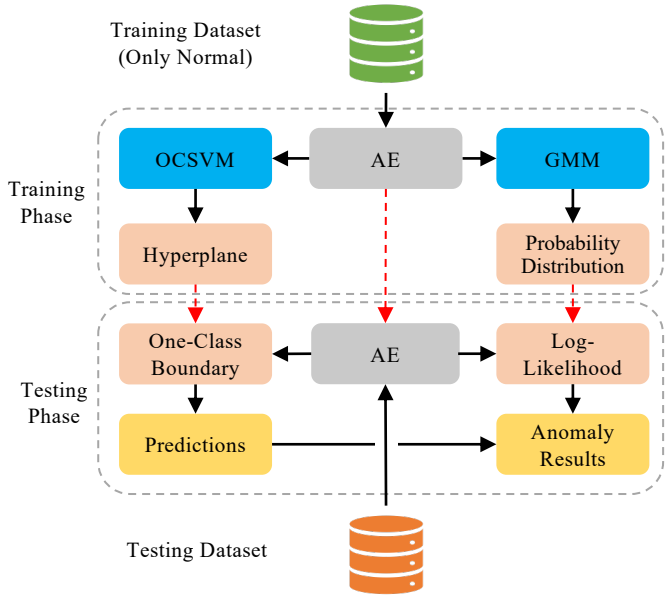


Figure 5. The proposed detection framework.

In the training step, we first train an AE model to extract representative features. After that, we employ the latent features to train two anomaly detectors simultaneously, OCSVM and GMM. These two detectors learn the characteristics of normal data from different viewpoints. In particular, the OCSVM finds a hyperplane, and the GMM learns the probability distribution. In summary, we learn about their models in the training phase. It should be noted that, in the training phase, only normal data are utilized.

Anomalies in the data can be identified if they deviate from the normal profile. Although the OCSVM learns a hyperplane for the normal samples, it may produce some false

positives. Since the GMM learns the distribution of normal data, we can use it to verify the prediction results produced via OCSVM.

With these well-trained models, we employ them to detect network attacks. We list the detection process in the Algorithm 1. In sequence, we employ the AE to extract the latent feature z_i of the testing sample x_i. We then apply OCSVM and GMM to them. The one-class boundary of OCSVM produces the relevant prediction result y_i first. Then, the negative log-likelihood provided by GMM $\mathcal{G}$ is utilized to recheck the prediction result. In particular, if a sample has been identified as attack but it has a lower negative log-likelihood than predefined threshold T, we reclassified the it as normal.

Algorithm 1: Testing process of our proposed methods.

Data: Original input data x_i. Encoder function ϕ of trained AE. The decision function $f(\cdot)$ of trained OCSVM. The trained GMM $\mathcal{G}$ and decision threshold T.

Result: Anomaly detection result for x_i.

`/* Step 1: Extract latent feature from encoder. */`

$z_i \leftarrow \phi(x_i)$

`/* Step 2: Predict the anomaly detection by OCSVM. */`

$y_i \leftarrow f(z_i)$

`/* Step 3: Recheck the misclassified predictions. */`

if $y_i == -1$ **then**
  $s_i = \mathcal{G}(z_i)$
  if $s_i \leq T$ **then**
    $y_i \leftarrow 1$
  end
end

`/* Step 4: Return the anomaly detection result. */`

if $y_i == -1$ **then**
  x_i is an anomaly
else
  x_i is not an anomaly
end

4. Experiment Results

In this part, we evaluate the effectiveness of our proposed detection model. First, we introduce the dataset utilized in the research. Then, the evaluation metrics are listed. After that, the detailed experiment settings are laid out. In the final part, we analyze the detection performance.

4.1. Dataset

In this work, there are two datasets utilized in the experiments, namely, NF-BoT-IoT-v2 and NF-CSE-CIC-IDS2018-v2 respectively [32]. There are seven types of traffic in the first dataset and five types of traffic in the second dataset. In the research, we refer to these two dataset as BoT-IoT and IDS2018 correspondingly.

Since there are a huge number of samples in the original datasets, we select some samples for each class at random. In detail, the BoT-IoT dataset comprises 143,748 samples, of which 65,150 are normal. The sampled IDS2018 dataset includes 296,502 samples and 120,000 is normal. Also, the precise distribution of various attacks is displayed in Figure 6.

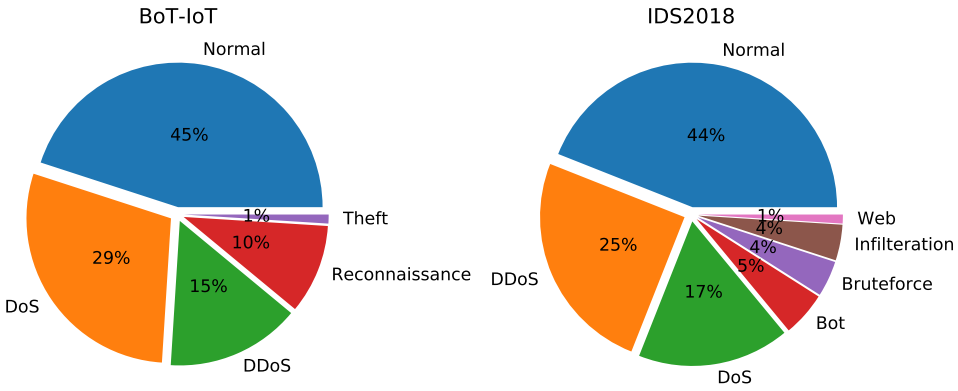


Figure 6. The distribution of different categories traffic in two datasets.

Every record is created with NetFlow v9 features set. After removing several irrelevant columns, such source or destination IP, there are 39 features left. We apply the log function to the numeric features first in order to minimize the influence of some large values. To further handle the category features in the data, the one-hot encoder is applied. After processing, there are roughly 200 features for the BoT-IoT dataset and 300 features for the IDS2018 dataset.

During the experiments, we use the 5-fold cross-validation method to evaluate the performance. As we seek to build a semi-supervised anomaly detection model, all the attack samples in the training set are removed. Before training the model, we use Min-Max normalization to scale all the features into a range of [0, 1].

4.2. Evaluation Metric

In Table 1, we present the confusion matrix. Considering the attack class as a positive type, there are four classification results between the true class and prediction class: true positive (TP), false negative (FN), false positive (FP), and true negative (TN). One TP record, for example, demonstrates that an attack sample is accurately classified as abnormal.

Table 1. Confusion matrix of detection results.

Actual Condition	**Predicted Condition**	
	Attack	**Normal**
Attack	True Positive	False Negative
Normal	False Positive	True Negative

We can derive some performance metrics from the four categories of classification results. The accuracy represents the proportion of samples that were properly classified, including both attack and normal samples, as shown below.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{18}$$

There are other two types of performance metrics: detection rate (DR) and false positive rate (FPR), as illustrated blow.

$$DR = \frac{TP}{TP + FN} \tag{19}$$

$$FPR = \frac{FP}{FP + TN} \tag{20}$$

Also, we include the three metrics frequently used in the classification field, including precision, recall, and F1.

$$Precision = \frac{TP}{TP + FP} \tag{21}$$

$$Recall = \frac{TP}{TP + FN} \tag{22}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{23}$$

The recall and DR have the same format. In the following comparisons, we only use the recall.

4.3. Experiment Settings

In this section, we describe the specifics of the experiment settings, including several baseline methods and parameter settings. We utilize six baseline techniques to compare our method. Particularly, the GMM and OCSVM are employed to illustrate the enhancement of our combination strategy. Except for the GMM, OCSVM, and AE mentioned earlier, the three additional methodologies are

- IForest. IForest builds numerous isolation trees from random subsets of data and provides an anomaly score by aggregating the results from each tree.
- KDE. KDE is a non-parametric approach to estimate the probability density function using a kernel function. The probability density can be utilized as anomaly score. In this work, we employ the Gaussian kernel.
- Deep support vector data description (DSVDD) [33]. Like OCSVM, the support vector data description (SVDD) is an one-class learning method. It aims to learn the smallest hypersphere that encloses most of the target data. DSVDD utilizes SVDD as a loss function to train the neural network and finds a hypersphere with a minimum volume.

During experiments, we utilize scikit-learn [34] to implement the semi-supervised detection techniques. Accounting for the AE, we apply the Keras [35] framework to program the neural network. We utilize a three-hidden-layer setting for both datasets. As for the number of neurons in the hidden layers, we choose "150-100-50-100-150" and "210-140-70-140-210" for the BoT-IoT and IDS2018 dataset, respectively, and PReLU [36] as the activation function in the hidden layer. During the training of the AE, the batch size is 256, the epoch is 200, and the learning rate is 0.001.

For the OCSVM, since there are no attack samples in the training set, it is difficult to pick the optional parameters. Two parameters have a significant influence on performance: ν and γ. As most samples are normal, we set the value of ν as 0.01. And the γ is 1.0. For GMM, we utilize 20 components.

For the IForest, we utilize the parameter recommended by the original paper, in which the number of estimators is 100 and the maximum number of samples is 256. The bandwidth of KDE is set to 1.0 by default in scikit-learn. The total training epoch of DSVDD is 200. It has the same network architecture as the encoder within AE. We train a AE model with first 50 epochs, and then DSVDD uses the weights from the trained AE encoder for initialization. The DSVDD is then trained over the last 150 epochs using SVDD loss. The batch size and learning rate are identical to the AE we used.

Except for the OCSVM, when employing other techniques like GMM or KDE as detectors, a threshold has to be selected for checking new samples. We take the 90th percentile of the anomaly score obtained from the training set as the threshold. It should be noted that our combination approaches employ the same parameter settings.

4.4. Experiment Results

In this subsection, we conduct several experiments to illustrate their comprehensive performance. In the first, we compare the detection performance with other baselines.

As the AE plays a critical role in the detecting process, we investigate its influence next. In the final part, we examine several significant hyperparameters that effect performance.

4.4.1. Detection Performance

We provide the detection performance, including five metrics, in the Tables 2 and 3 for both datasets. In the tables, we present both the mean value and the standard deviation. It should be emphasized that all the approaches are trained on the latent representations derived from the AE.

Table 2. Detection performance of BoT-IoT dataset.

Method	Accuracy	Precision	Recall	F1	FPR
IForest	66.29 ± 7.53	84.37 ± 3.74	46.55 ± 14.04	59.26 ± 12.64	9.84 ± 0.57
KDE	89.97 ± 0.38	91.54 ± 0.22	89.99 ± 0.54	90.76 ± 0.37	10.06 ± 0.24
GMM	95.45 ± 0.12	92.35 ± 0.18	$\mathbf{99.98 \pm 0.01}$	96.01 ± 0.10	10.02 ± 0.26
OCSVM	88.65 ± 6.97	83.58 ± 8.66	99.96 ± 0.01	90.84 ± 5.18	25.02 ± 15.41
AE	94.42 ± 2.01	92.17 ± 0.35	98.15 ± 3.65	95.04 ± 1.91	10.08 ± 0.25
DSVDD	95.36 ± 0.11	92.21 ± 0.17	$\mathbf{99.98 \pm 0.01}$	95.94 ± 0.09	10.22 ± 0.24
Ours	$\mathbf{98.73 \pm 0.33}$	$\mathbf{97.78 \pm 0.59}$	99.95 ± 0.01	$\mathbf{98.85 \pm 0.30}$	$\mathbf{2.75 \pm 0.74}$

Bold font indicates best results. The following table are the same.

Table 3. Detection performance of IDS2018 dataset.

Method	Accuracy	Precision	Recall	F1	FPR
IForest	66.57 ± 11.87	86.94 ± 4.82	50.60 ± 19.87	62.58 ± 17.27	9.93 ± 0.15
KDE	92.86 ± 0.15	93.20 ± 0.23	94.93 ± 0.04	94.06 ± 0.12	10.18 ± 0.37
GMM	93.25 ± 0.14	93.03 ± 0.20	95.84 ± 0.11	94.41 ± 0.11	10.57 ± 0.33
OCSVM	92.62 ± 0.68	92.05 ± 1.01	$\mathbf{95.90 \pm 0.05}$	93.93 ± 0.53	12.20 ± 1.67
AE	91.30 ± 2.81	92.89 ± 0.29	92.45 ± 4.86	92.62 ± 2.59	10.39 ± 0.36
DSVDD	92.36 ± 1.23	93.15 ± 0.14	94.08 ± 2.13	93.61 ± 1.10	10.17 ± 0.21
Ours	$\mathbf{95.10 \pm 0.13}$	$\mathbf{96.10 \pm 0.30}$	95.65 ± 0.13	$\mathbf{95.87 \pm 0.11}$	$\mathbf{5.72 \pm 0.46}$

We analyze the results of the BoT-IoT dataset first. IForest shows the lowest performance for most metrics and a larger standard deviation. Other methods present a value of F1 higher than 90%. They exhibit a remarkable improvement over IForest. Except for IForest and KDE, all other approaches demonstrate a higher recall with a value of 98%, which indicates they identify most of the attack samples. AE, GMM, and DSVDD present similar results on all metrics. OCSVM displays a lower precision value. Particularly, OCSVM has the highest FPR. When it comes to our approach, since it combines the power of GMM and OCSVM, it has the highest results in terms of all metrics except for recall. The FPR is the lowest in the table, with a value of 2.75%.

From the Table 3 concerning the IDS2018 dataset, the OCSVM performs well on this dataset compared with the BoT-IoT dataset. Our approach continues to deliver the highest results, as before. However, the F1 achieves roughly 96%, which is lower than the 99% in the previous table. The recall of our techniques is a little lower than the maximum result of OCSVM. In addition, the FPR is reduced as compared to other techniques.

From the results mentioned above, in conclusion, our approach displays the highest result in terms of most metrics. When compared to a single model, GMM or OCSVM, our approach employs both and delivers a better result. In detail, we use GMM to check the samples that are predicted to be anomalous by the OCSVM. In this way, the false positive samples can be reclassified. From the results shown in the table, the FPR has been reduced significantly for both datasets, which proves the improvement of our combination method.

4.4.2. Effect of Feature Extraction by AE

In the previous content analysis, we provided the performance anomaly detector trained on the latent features extracted from the AE. It is reasonable to assess the impact

of the AE. In Figures 7 and 8, we compare the performance of various detectors whether trained with original features or not. For simplicity, we plot the precision, recall, and F1 only. It should be noted that whether latent features are employed or not, detectors use the same parameter settings.

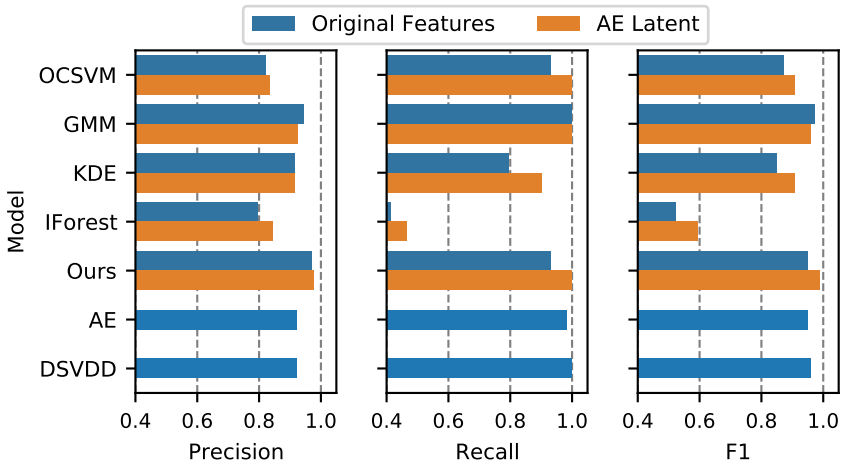


Figure 7. Performance comparisons of whether using an AE for the BoT-IoT dataset. The mean value of three metrics, including precision, recall, and F1, is presented. Since AE or DSVDD directly applies to original features, there is one bar for the AE and DSVDD.

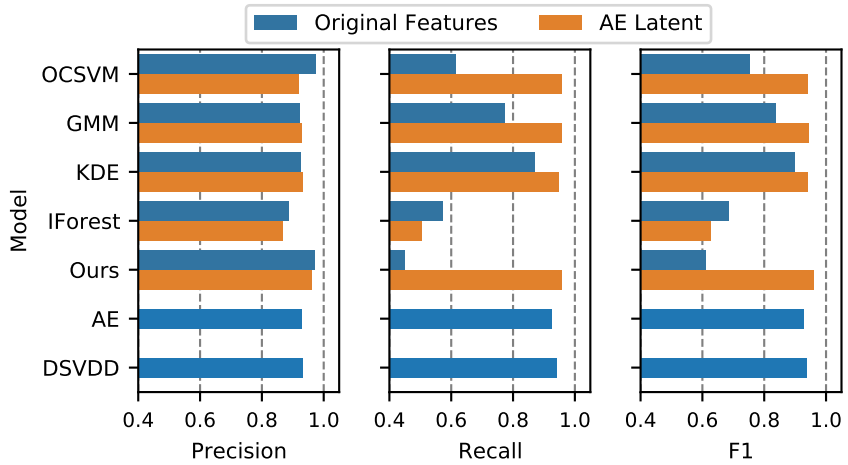


Figure 8. Performance comparisons of whether using AE for IDS2018 dataset.

From the comparisons for both datasets, when using latent features, most methods exhibit improvements in terms of F1, which is a metric combining both precision and recall. IForest indicates decrements for both datasets. GMM indicates a little bit of a decline in BoT-IoT and an increase in IDS2018. For our combination method, it demonstrates improvements in both datasets.

It should be noted that, if we do not employ AE features, the combination method does not demonstrate improvement in terms of F1. When employing original features for the IDS2018 dataset, for example, our combination technique show lower performance than the single model, OCSVM or GMM. This is because both OCSVM and GMM perform poorly on original features. It illustrates that the AE extracts useful features for two detectors in this way.

4.4.3. Parameter Analysis

In this part, we investigate two parameters within the GMM: the number of components and the selection of threshold. Also, we provides the approaches that utilize original

features for comparison. In the experiments, there are four techniques involving GMM and our method.

In the first stage, the number of GMM components is examined. During experiments, the threshold is fixed at the 90th percentile of the anomaly score from the training dataset. The components are taken from [1, 5, 10, 15, 20]. The Figures 9 and 10 illustrate the comparisons for both dataset.

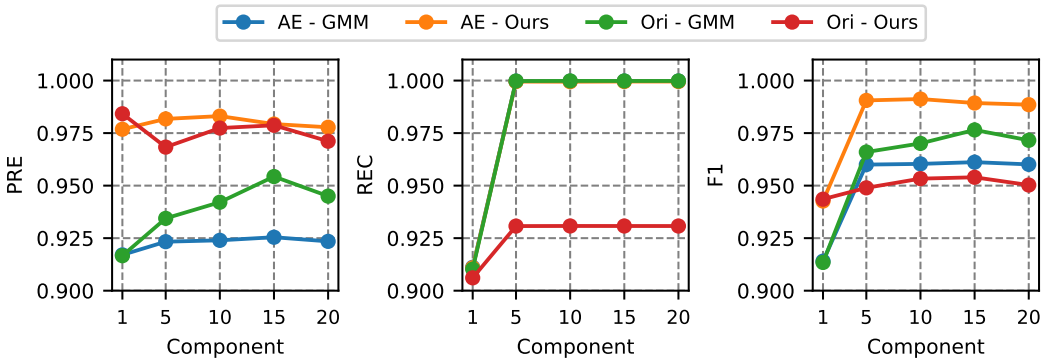


Figure 9. Performance comparisons with different number of GMM components for BoT-IoT dataset.

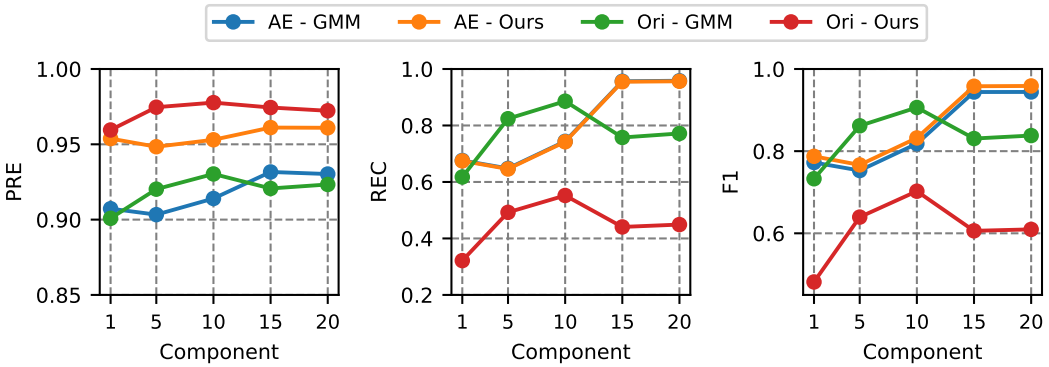


Figure 10. Performance comparisons with different number of GMM components for IDS2018 dataset.

The recall of the BoT-IoT dataset becomes steady as components reach 10. The results of the recall from the IDS2018 dataset show a different pattern. Our method achieves the highest recall when the number of components is 15. For both datasets, the F1 score begins to grow with the number of GMM components. When employing original features to train our method, the performance is lowest in terms of recall and F1. For both datasets, our technique provides the highest F1 when the components are 20.

In the following, we investigate the influence of threshold selection. During the experiment, we chose 20 GMM components. Then we pick the threshold from the anomaly score obtained by the training set, with some representative percentile values, [80, 85, 90, 95, 99]. The detail results are displayed in Figures 11 and 12.

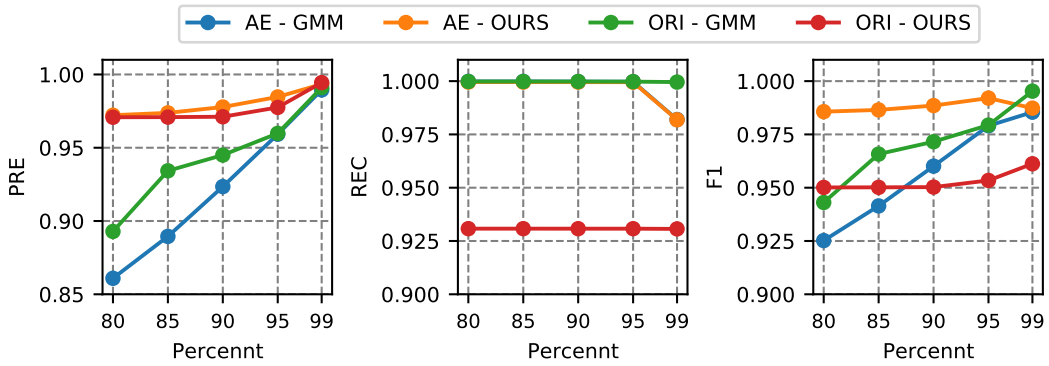


Figure 11. Performance comparisons with different threshold for BoT-IoT dataset.

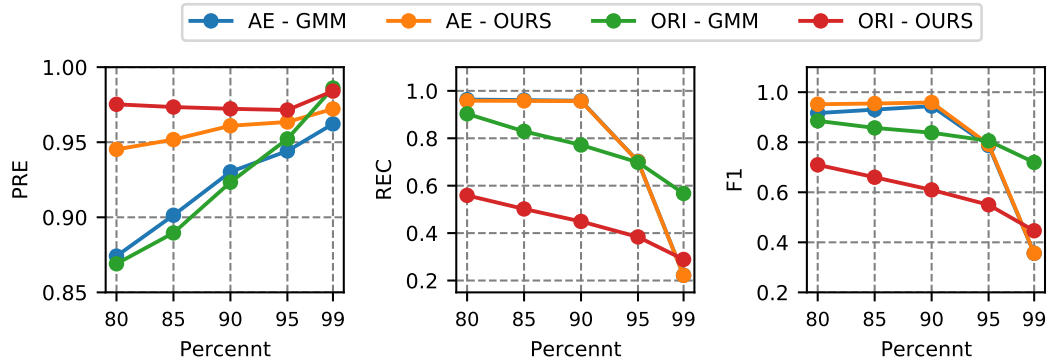


Figure 12. Performance comparisons with different threshold for IDS2018 dataset.

The F1 of the BoT-IoT dataset displays increasing tendencies as the threshold increases. In particular, its recall is steady at the various thresholds. but the precision is rising. The results of the IDS2018 dataset are different. The precision is growing, but the recall is decreasing after reaching 90th. For the F1 score, our method delivers the highest results at 90th.

Since there are no attack samples during training, it is difficult to determine optimal parameters. We can pick reasonable parameters based on domain experience. From the comparisons stated above, the parameters selected for GMM are acceptable.

5. Conclusions

With the increased danger of network attacks, it is critical to develop effective IDS. Machine learning and deep learning have been extensively employed in the field of IDS. However, the challenge of high-dimensional data and the lack of labeled datasets pose an obstacle to the development of IDS.

In this paper, we present a semi-supervised anomaly-based IDS that is trained on the normal dataset only. The model utilizes an AE first to alleviate the influence of high-dimensional data. After that, GMM and OCSVM are trained on representative features derived from the AE. Then, they are used to examine the samples collectively. The results of the experiment indicate the effectiveness of the suggested approach. In one part, the AE boosts the performance of detectors. In another part, the combination method produces better results than a single detector.

However, there are some limitations of our method. The hyperparameters used within the models are set by domain experience. Although it performs well on these two datasets, it may not generally perform well on other ones. In addition, if the OCSVM performs badly, it is hard to obtain a high detection performance even with the help of the GMM, as the trained GMM is only applied to the samples that are predicted to be anomalous by the OCSVM.

In the future, there are several directions that deserve investigation. Some researchers have proposed some methods to select ν or γ within the OCSVM model when only normal

data are available. We can apply these hyperparameter selection methods to obtain higher performance. Since combining various detectors can produce better detection results, it is meaningful to try different detector combinations. It is reasonable to check the normal samples predicted by OCSVM also. However, the strategy should be designed with caution.

Author Contributions: Conceptualization, C.W. (Chao Wang) and Y.S.; Data curation, C.W. (Chao Wang) and S.L.; Formal analysis, H.L.; Funding acquisition, B.W.; Investigation, C.W. (Chao Wang), S.L. and C.W. (Chonghua Wang); Methodology, C.W. (Chao Wang); Project administration, B.W.; Resources, B.W.; Software, C.W. (Chao Wang) and S.L.; Supervision, H.L. and B.W.; Validation, C.W. (Chao Wang), Y.S. and S.L.; Visualization, C.W. (Chao Wang); Writing—original draft, C.W. (Chao Wang); Writing—review & editing, B.W. All authors have read and agreed to the published version of the manuscript.

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

Data Availability Statement: In this study, we use the intrusion detection datasets introduced in [32]. Readers can refer the corresponding paper for detail information.

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

References

1. Liu, H.; Lang, B. Machine learning and deep learning methods for intrusion detection systems: A survey. *Appl. Sci.* **2019**, *9*, 4396. [CrossRef]
2. Ferrag, M.A.; Maglaras, L.; Ahmim, A.; Derdour, M.; Janicke, H. RDTIDS: Rules and Decision Tree-Based Intrusion Detection System for Internet-of-Things Networks. *Future Internet* **2020**, *12*, 44. [CrossRef]
3. Mohammadi, M.; Rashid, T.A.; Karim, S.H.; Aldalwie, A.H.M.; Tho, Q.T.; Bidaki, M.; Rahmani, A.M.; Hosseinzadeh, M. A comprehensive survey and taxonomy of the SVM-based intrusion detection systems. *J. Netw. Comput. Appl.* **2021**, *178*, 102983. [CrossRef]
4. Bhavani, T.T.; Rao, M.K.; Reddy, A.M. Network Intrusion Detection System Using Random Forest and Decision Tree Machine Learning Techniques. In *First International Conference on Sustainable Technologies for Computational Intelligence: Proceedings of ICTSCI 2019*; Luhach, A.K., Kosa, J.A., Poonia, R.C., Gao, X.Z., Singh, D., Eds.; Springer: Singapore, 2020; pp. 637–643.
5. Cao, V.L.; Nicolau, M.; McDermott, J. Learning Neural Representations for Network Anomaly Detection. *IEEE Trans. Cybern.* **2019**, *49*, 3074–3087. [CrossRef]
6. Choi, H.; Kim, M.; Lee, G.; Kim, W. Unsupervised learning approach for network intrusion detection system using autoencoders. *J. Supercomput.* **2019**, *75*, 5597–5621. [CrossRef]
7. Chandola, V.; Banerjee, A.; Kumar, V. Anomaly Detection: A Survey. *ACM Comput. Surv.* **2009**, *14*, 1–58. [CrossRef]
8. Schölkopf, B.; Platt, J.C.; Shawe-Taylor, J.; Smola, A.J.; Williamson, R.C. Estimating the support of a high-dimensional distribution. *Neural Comput.* **2001**, *13*, 1443–1471. [CrossRef]
9. Alazzam, H.; Sharieh, A.; Sabri, K.E. A lightweight intelligent network intrusion detection system using OCSVM and Pigeon inspired optimizer. *Appl. Intell.* **2022**, *52*, 3527–3544. [CrossRef]
10. Al Shorman, A.; Faris, H.; Aljarah, I. Unsupervised intelligent system based on one class support vector machine and Grey Wolf optimization for IoT botnet detection. *J. Ambient Intell. Humaniz. Comput.* **2020**, *11*, 2809–2825. [CrossRef]
11. Cao, V.L.; Nicolau, M.; McDermott, J. A Hybrid Autoencoder and Density Estimation Model for Anomaly Detection. In *Parallel Problem Solving from Nature—PPSN XIV*; Handl, J., Hart, E., Lewis, P.R., López-Ibáñez, M., Ochoa, G., Paechter, B., Eds.; Springer: Cham, Switzerland, 2016; pp. 717–726.
12. Vaiyapuri, T.; Binbusayyis, A. Application of deep autoencoder as an one-class classifier for unsupervised network intrusion detection: A comparative evaluation. *PeerJ Comput. Sci.* **2020**, *6*, 1–26. [CrossRef]
13. Blanco, R.; Malagón, P.; Briongos, S.; Moya, J.M. Anomaly Detection Using Gaussian Mixture Probability Model to Implement Intrusion Detection System. In *Hybrid Artificial Intelligent Systems*; Pérez García, H., Sánchez González, L., Castejón Limas, M., Quintián Pardo, H., Corchado Rodríguez, E., Eds.; Springer: Cham, Switzerland, 2019; pp. 648–659.
14. Ahmad, Z.; Shahid Khan, A.; Wai Shiang, C.; Abdullah, J.; Ahmad, F. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Trans. Emerg. Telecommun. Technol.* **2021**, *32*, 1–29. [CrossRef]
15. Vinayakumar, R.; Alazab, M.; Soman, K.P.; Poornachandran, P.; Al-Nemrat, A.; Venkatraman, S. Deep Learning Approach for Intelligent Intrusion Detection System. *IEEE Access* **2019**, *7*, 41525–41550. [CrossRef]
16. Yang, Y.; Zheng, K.; Wu, C.; Yang, Y. Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network. *Sensors* **2019**, *19*, 2528. [CrossRef]
17. Malaiya, R.K.; Kwon, D.; Suh, S.C.; Kim, H.; Kim, I.; Kim, J. An Empirical Evaluation of Deep Learning for Network Anomaly Detection. *IEEE Access* **2019**, *7*, 140806–140817. [CrossRef]

18. Thapa, N.; Liu, Z.; Kc, D.B.; Gokaraju, B.; Roy, K. Comparison of machine learning and deep learning models for network intrusion detection systems. *Future Internet* **2020**, *12*, 167. [CrossRef]
19. Alzubaidi, L.; Zhang, J.; Humaidi, A.J.; Al-Dujaili, A.; Duan, Y.; Al-Shamma, O.; Santamaría, J.; Fadhel, M.A.; Al-Amidie, M.; Farhan, L. Review of Deep Learning: Concepts, CNN Architectures, Challenges, Applications, Future Directions. *J. Big Data* **2021**, *8*, 53. [CrossRef]
20. Abdelmoumin, G.; Whitaker, J.; Rawat, D.B.; Rahman, A. A Survey on Data-Driven Learning for Intelligent Network Intrusion Detection Systems. *Electronics* **2022**, *11*, 213. [CrossRef]
21. Fu, Y.; Du, Y.; Cao, Z.; Li, Q.; Xiang, W. A Deep Learning Model for Network Intrusion Detection with Imbalanced Data. *Electronics* **2022**, *11*, 898. [CrossRef]
22. Abdulhammed, R.; Musafer, H.; Alessa, A.; Faezipour, M.; Abuzneid, A. Features dimensionality reduction approaches for machine learning based network intrusion detection. *Electronics* **2019**, *8*, 322. [CrossRef]
23. 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]
24. Sadaf, K.; Sultana, J. Intrusion detection based on autoencoder and isolation forest in fog computing. *IEEE Access* **2020**, *8*, 167059–167068. [CrossRef]
25. Yan, W. Detecting Gas Turbine Combustor Anomalies Using Semi-Supervised Anomaly Detection with Deep Representation Learning. *Cogn. Comput.* **2020**, *12*, 398–411. [CrossRef]
26. Liao, J.; Teo, S.G.; Pratim Kundu, P.; Truong-Huu, T. ENAD: An ensemble framework for unsupervised network anomaly detection. In Proceedings of the 2021 IEEE International Conference on Cyber Security and Resilience (CSR), Rhodes, Greece, 26–28 July 2021; pp. 81–88.
27. Géron, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2022.
28. Beggel, L.; Pfeiffer, M.; Bischl, B. Robust Anomaly Detection in Images Using Adversarial Autoencoders. In *Proceedings of the Machine Learning and Knowledge Discovery in Databases*; Brefeld, U., Fromont, E., Hotho, A., Knobbe, A., Maathuis, M., Robardet, C., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 206–222.
29. Seliya, N.; Abdollah Zadeh, A.; Khoshgoftaar, T.M. A Literature Review on One-Class Classification and Its Potential Applications in Big Data. *J. Big Data* **2021**, *8*, 122. [CrossRef]
30. Bishop, C.M.; Nasrabadi, N.M. *Pattern Recognition and Machine Learning*; Springer: New York, NY, USA, 2006; Volume 4.
31. Aggarwal, C.C. *Outlier Analysis*; Springer: New York, NY, USA, 2013; pp. 1–446.
32. Sarhan, M.; Layeghy, S.; Portmann, M. Towards a Standard Feature Set for Network Intrusion Detection System Datasets. *Mob. Netw. Appl.* **2021**, *27*, 357–370.
33. 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; pp. 4393–4402.
34. 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.
35. Keras. 2015. Available online: https://keras.io (accessed on 10 February 2023).
36. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1026–1034.