Hindawi
Security and Communication Networks
Volume 2022, Article ID 5713074, 12 pages
https://doi.org/10.1155/2022/5713074

Research Article

Dimension Reduction Technique Based on Supervised Autoencoder for Intrusion Detection of Industrial Control Systems

Chao Wang,[1,2] Hongri Liu,[1,2] Yunxiao Sun,[1,2] Yuliang Wei,[1,2] Kai Wang,[1,2] and Bailing Wang[1,2]

[1]*School of Computer Science and Technology, Harbin Institute of Technology, Weihai264209, China*
[2]*School of Cyber Science and Technology, Harbin Institute of Technology, Harbin150001, China*

Correspondence should be addressed to Bailing Wang; wbl@hit.edu.cn

Received 1 November 2021; Accepted 17 May 2022; Published 10 June 2022

Academic Editor: Gu Zhaoquan

Industrial control systems (ICSs) are closely related to human life. In recent years, many ICSs have been connected to the Internet rather than being physically isolated, which has improved business efficiency while also increasing the risks of being attacked. The security issues of ICSs have gotten a lot of interest in the research community because attack events that aim at ICSs can cause catastrophic damage. An intrusion detection system (IDS) serves as an important tool for providing protection. Many IDS studies using machine learning and deep learning have been proposed. However, high-dimensional data may cause overfitting, resulting in inferior performance. To improve the classification performance, we suggest a dimension reduction technique based on the supervised autoencoder (SupervisedAE) and principal components analysis (PCA) in this study. To obtain more discriminative latent representations, compared with the conventional autoencoder, the SupervisedAE absorbs the label information during the training process. In this way, the improved autoencoder model is trained with reconstruction error and classification error simultaneously. Based on the latent representations extracted from the SupervisedAE, we add the PCA algorithm. The additional PCA algorithm reduces the dimension of features further. We conduct a series of experiments utilizing the suggested technique on a public power system data set to evaluate the performance. Compared with various dimension reduction methods, including autoencoder variants, the technique proposed in this study shows higher performance. In the meanwhile, it outperforms some existing detection methods in terms of accuracy and F1 score.

1. Introduction

Industrial control systems (ICSs) are widely used in industries to fulfill certain industrial goals, such as manufacturing, material transportation, or energy transportation [1]. Since the ICSs were connected to the Internet and exposed to numerous attacks, the security problems have been studied for years. ICSs differ from standard IT systems in that when they are been attacked, it poses a major danger to human health and safety, as well as financial loss [1]. Many ICS attacks have occurred in recent years [2], including Stuxnet [3], BlackEnergy [4], and Industroyer [5].

Intrusion detection system (IDS) has received a lot of attention as a strong tool for providing protection [6–10]. In general, IDS examines data records in the network or from the host to determine whether or not operations are safe. The system would raise alarms if operations are malicious. The operator would then make the decision based on the examination results. IDSs based on machine learning and deep learning [11–14] have been thoroughly investigated in recent years. However, overfitting caused by high-dimensional data still poses a challenge for the performance of IDS. The authors of [15] point out that the dimension reduction techniques, in addition to the classifier algorithms, are important in the IDS research also. There have been many works proposed for the dimension reduction of IDS. To build an efficient IDS, principle component analysis (PCA) and linear discriminant analysis (LDA) are used [16, 17]. A sparse autoencoder [18] is implemented to reduce the dimension of input features for support vector machine (SVM) based IDS.

In this study, we suggest a dimension reduction technique that combines supervised autoencoder (SupervisedAE) with PCA algorithm for the IDS to overcome the challenge of high-dimensional problems. Compared with the conventional autoencoder, SupervisedAE adds label information during the training time to obtain more discriminative latent representations. The improved autoencoder model is trained by reconstruction error and classification error. To further decrease the dimension, the PCA algorithm is applied to the latent representations extracted from SupervisedAE. This research has made the following contributions:

(i) We employ an improved autoencoder called SupervisedAE for the dimension reduction of IDS. The novel model adds a softmax layer that connects to the output of the encoder. With the joint loss function (reconstruction error and classification error), the SupervisedAE learns more discriminative latent representations. Experiment results demonstrate that the SupervisedAE outperforms other autoencoders.

(ii) To reduce the dimension further, we apply the PCA algorithm to the latent representations extracted from SupervisedAE. We use a nested cross-validation procedure to select the optimal number of PCA components in the experiment. The combined technique shows higher performance than some other dimension reduction methods.

(iii) We conduct a series of experiments on a power system data set. There are four distinct classifiers used in the test. With the suggested dimension reduction technique, these classifiers achieve higher performance than using original features. In particular, the K-nearest neighbors (KNN) classifier achieves the best results. The results are higher than other existing detection methods.

The remainder of the paper is laid out as follows. First, some related works are presented in Section 2. Especially, some works that involve dimension reduction are thoroughly examined. Then we have a detailed introduction to the SupervisedAE and the whole framework of IDS in Section 3. After that, Section 4 shows the performance of our method evaluated on the power system data set and compares the results to other methods. Finally, we draw the corresponding conclusion and point out some future works in Section 5.

2. Related Work

The ICSs have been connected to the Internet in recent years, which increases the danger of being hacked. The security issues of ICSs have received a lot of attention [19, 20]. IDS is one of the effective security solutions [21] for monitoring activities and ensuring regular processes (the other three being authentication solutions, privacy-preserving solutions, and key management systems).

Many works about IDS have been proposed in the research field. An SVM-based IDS is built using the features of ICS communication to categorize regular and abnormal packets [6]. The authors of [7] presented an IDS model based on a random forest with an adaptive boosting technique to increase the detection rate. An IDS based on a bidirectional simple recurrent unit is proposed in [8]. With the help of skip connections, the proposed model improves the training effectiveness. To build an effective intrusion detection model, a hybrid deep belief network (DBN) is developed [22], which improves accuracy over previous DBN approaches. Two detection methods based on random subspace methods were proposed [9, 10]. These two methods would be used to compare with our method.

The authors of [15] introduce some key points related to IDS. In particular, they introduce some works that include feature engineering. Because of the curse of dimensionality, high-dimensional data may induce overfitting for trained models, as well as require more memory and computational cost. How to reduce the dimension of features is a significant work. There are two approaches for removing irrelevant features and improving the performance of the model: feature selection and feature extraction. The feature selection method [23, 24] is used to select a subset of features. Compared with feature selection, feature extraction maps the original features into a low-dimensional feature space. Authors of [25] proposed a model that combines oversampling and feature selection. The model employs the gradient penalty Wasserstein generative adversarial networks to generate additional attack samples and the ANOVA approach to choose a subset of features. The combined model shows better performance. An artificial neural network classifier is used to create the IDS [26]. The ranking of information gain and correlation is used to implement feature reduction. The results are promising. A correlation-based feature selection strategy was proposed to eliminate irrelevant features and improve detection performance [27]. SVM, multiple layer perceptron (MLP), and KNN are used to identify attacks using the new subset of features. The KNN technique delivers the best results when compared to results produced utilizing original features. We would compare performance with these methods.

As a feature extraction method, PCA has been widely used in the field of intrusion detection [28–30]. A hybrid approach combining information gain and PCA was proposed [29], and the ensemble classifier based on SVM, instance-based learning algorithm, and MLP is used to detect attacks after dimension reduction. The model has achieved encouraging performance. Autoencoder, a sort of neural network, is also used to reduce the dimension of features [31].

Various works have proved that dimension reduction methods could improve the performance of IDS. In this study, we focus on the feature extraction method and improve the autoencoder model by absorbing the label information during training time.

3. Methods

We introduce the overall framework of our suggested technique in this section. To begin, we go through the theory of autoencoder in-depth, covering the conventional autoencoder and sparse autoencoder. In particular, the SupervisedAE is discussed. Then, we explain the PCA algorithm applied to the latent representations extracted from SupervisedAE. Finally, the entire framework of IDS is presented.

3.1. Autoencoder

3.1.1. Basic Autoencoder. Autoencoder (AE) is an unsupervised neural network [32]. Typically, the autoencoder is employed to reduce the dimension of features. The fundamental concept of the autoencoder is to rebuild the input.

As shown in Figure 1, the autoencoder is separated into two parts: encoder and decoder. The encoder converts the input into the latent representation, and in the meanwhile, the decoder reconstructs input from the compressed latent representation. Considering the following set of input data $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, where n is the number of samples in the data set, the encoder function ϕ transforms the input data $\mathbf{x}_i$ into the latent representation $\mathbf{z}_i$. Then, using the decoder function φ, the new reconstructed input $\widehat{\mathbf{x}}_i$ is obtained as follows:

$$\begin{aligned} \mathbf{z}_i &= \phi(\mathbf{x}_i), \\ \widehat{\mathbf{x}}_i &= \varphi(\mathbf{z}_i). \end{aligned} \tag{1}$$

The goal of the training autoencoder is to find parameters in the encoder and decoder that minimize the reconstruction error. In this study, we use the mean squared error to calculate it. The corresponding loss function L_R is as follows:

$$L_R = \frac{1}{n}\sum_{i=1}^{n}\left\|\widehat{\mathbf{x}}_i - \mathbf{x}_i\right\|^2. \tag{2}$$

The classic autoencoder is a three-layer neural network with only one hidden layer. The other two layers are input and output. The number of neurons in the hidden layer is usually fewer than it in the input layer to avoid the network just copying the input into output. The architecture can be expanded to include more hidden layers. The number of neurons in the encoder gradually decreases. And the number of neurons is generally symmetrical for encoder and decoder. The compressed latent representations are learned by the autoencoder in this way. The decoder is usually abandoned after training the autoencoder, and the latent representations are used for following classification or other operations.

3.1.2. Sparse Autoencoder. To extract better feature representations, there are some variants of autoencoder by imposing constraints. The sparse autoencoder (SparseAE) [33] adds a sparse penalty term in the hidden layer. The corresponding loss function L_S is given by

$$L_S = L_R + \beta\sum_{j=1}^{s} KL\left(\rho|\widehat{\rho}_j\right), \tag{3}$$

$$KL\left(\rho|\widehat{\rho}_j\right) = \rho\log\frac{\rho}{\widehat{\rho}_j} + (1-p)\log\frac{1-\rho}{1-\widehat{\rho}_j}. \tag{4}$$

Compared with the conventional autoencoder, the loss function of SparseAE add a sparsity penalty term $\sum_{j=1}^{s} KL(\rho|\widehat{\rho}_j)$, where s is the number of neurons in the hidden layer, ρ is a predefined sparsity parameter, and $\widehat{\rho}_j$ denote the average activation of hidden unit j. KL function is the Kullback–Leibler divergence that is used to measure the divergence between ρ and $\widehat{\rho}_j$. β in (3) is used to control the weight of the penalty.

3.1.3. Supervised Autoencoder. In this study, we utilize a supervised classification model for the development of IDS. It should be trained using the labeled data set with normal and attack samples before being used in practice. When we employ the classic autoencoder to reduce the dimension of features, it is frequently used in an unsupervised manner without the label, as stated before. It seeks to minimize the reconstruction error as much as possible. Then, the IDS model is trained on the compressed latent representations instead of original features.

To improve the conventional autoencoder and obtain more discriminative latent representations for classifying, a supervised autoencoder model was proposed [34, 35]. The novel model adds the class label during the training process of the autoencoder. Figure 2 depicts the architecture of the SupervisedAE model used in this study. It adds a softmax layer connected to the latent layer compared with the autoencoder shown in Figure 1. The label information is processed by the softmax layer. The number of neurons in the softmax layer is the same as the number of classes. In this way, the SupervisedAE is trained by reconstruction error and classification error simultaneously. The implementation of SupervisedAE employed in this study is described as follows.

Softmax is a function used by neural networks to perform classification tasks, in which cross-entropy is used to measure the classification error. Consider the following collection of classes: $C_K = \{1, 2, \ldots, K\}$, with the label $\{y_1, y_2, \ldots, y_n\}$, where $y_i \in C_K$, for input $\mathbf{x}_i$, the softmax function outputs a probability p_{ij} for class j as demonstrated in the following equation:

$$p_{ij} = \text{softmax}_j(\mathbf{q}_i) = \frac{\exp\left(q_{ij}\right)}{\sum_{l=1}^{K}\exp\left(q_{il}\right)}, \tag{5}$$

where $\mathbf{q}_i$ is the output of fully connected layer connected to latent layer.

With the corresponding probability, the classification loss L_C is calculated by

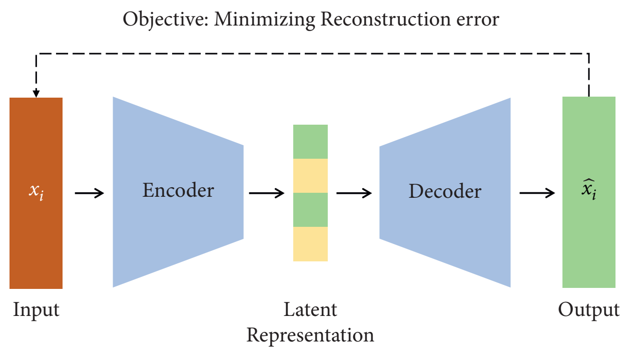

Figure 1: The network architecture of autoencoder.

$$L_C = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{K}\delta(y_i == j)\log(p_{ij}), \tag{6}$$

where $\delta(\text{condition})$ is a function that indicates whether or not the condition is satisfied. If the condition is met, $\delta(\text{condition}) = 1$. Otherwise, the value is 0. The following equation displays the joint loss function L:

$$L = L_R + \alpha L_C. \tag{7}$$

The relationship between reconstruction error and classification error is controlled by the variable α.

In this section, we introduce the SupervisedAE model. The new model is trained using a joint loss function with reconstruction error and classification error. In particular, the additional softmax layer calculates classification error. After training the improved autoencoder, when testing new samples, the decoder and softmax layer are abandoned. The new latent representation is more discriminative compared with that obtained by the conventional autoencoder.

3.2. PCA Algorithm. The PCA algorithm identifies the principal components in a data set that account for the largest amount of variance [36]. It has been used as a feature extraction method widely. The SupervisedAE introduced before can reduce the dimension of features to any predefined values. Motivated by the previous work that combines the sparse autoencoder and PCA [37], we employ the PCA algorithm to reduce the dimension of latent representations further. In this way, we can set the hidden layer settings of the SupervisedAE to some reasonable values and use the nested cross-validation procedure to choose the best number of components for PCA. The final reduced features output by PCA is used to train various classifiers.

The steps for extracting principal components are outlined below. Considering the latent representations extracted from SupervisedAE, the mean value is calculated first from each dimension. The covariance matrix is computed after removing the mean value for all dimensions. Then, derive the eigenvectors and eigenvalues of the covariance matrix. The eigenvectors are chosen based on the number of components predefined and sorted by eigenvalues. A feature matrix is constructed by combining selected eigenvectors and sorting them by eigenvalues. Finally, the latent representations are transformed using the feature matrix. Readers can refer to [38] for further information on PCA calculation.

3.3. Framework. The basic elements of our proposed IDS model are presented with the introduction of the SupervisedAE and PCA algorithm. This section introduces the entire working procedure.

Figure 3 displays the whole framework. There are two phases: training and testing. The data set is divided into a training set and a testing set, and the training set will be utilized to train the overall model. The model has three parts: normalization module, dimension reduction module, and classifier module. In the training process, we first train the dimension reduction module. The detailed process is shown in Algorithm 1. All features are scaled using the min-max normalization. After that, we use the scaled training set and corresponding label information to train the SupervisedAE model as shown in lines 2–6. Then, the PCA algorithm is trained on the latent representations extracted from the SupervisedAE. Finally, the training set whose features have been reduced by the reduction module is used to train the various classifiers.

In the testing phase, all of the trained modules are merged as the IDS model. And the integrated model predicts the class label for the testing set.

4. Evaluation

In this section, we display the performance of our proposed dimension reduction technique conducted on a power system data set. First, we introduce the data set used in the experiments. Then, the metrics used to evaluate the performance are described. Finally, the experiment results are presented and discussed.

4.1. Data Set. To evaluate the performance, we use the power system attack data set [39] created by Mississippi State University and Oak Ridge National Laboratory. The data set was generated by the configuration shown in Figure 4. The data set mainly includes measurements from each phasor measurement unit and data log from Snort, a simulation control panel, and relays.

There are two power generators in the configuration: G1 and G2. There are also four breakers (BR1 to BR4) that can be turned on or off by intelligent electronic devices (IEDs, R1 to R4). The data set has 128 features and 1 label for each sample. Each phasor measurement unit (PMU) has 29 different types of measurements, accounting for 116 different attributes in total. Table 1 contains a collection of corresponding features and their descriptions. The work state of PMU is described by these features. In addition, there are 12 features including control panel logs, snort alerts, and relay logs.

In the data set, there are 37 power system event scenarios, including natural events (8), no events (1), and attack events (28). In Table 2, the associated events and labels are listed. There are three attacks: data injection,

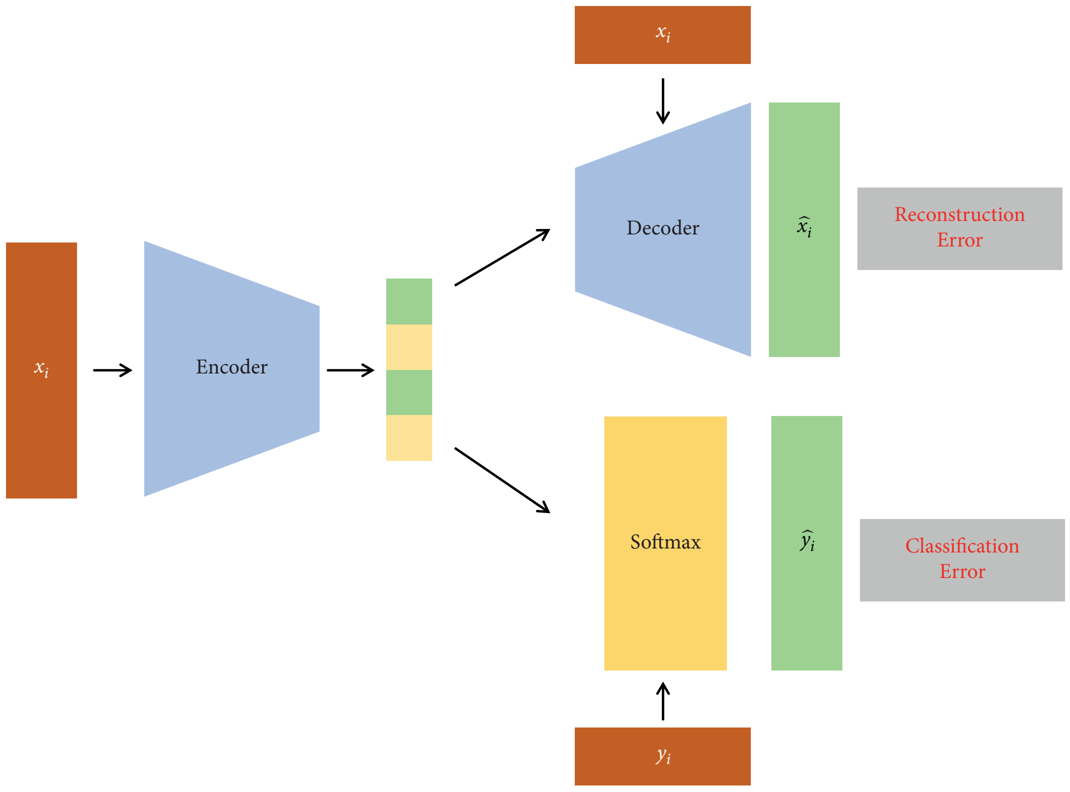

Figure 2: The proposed supervised autoencoder framework.

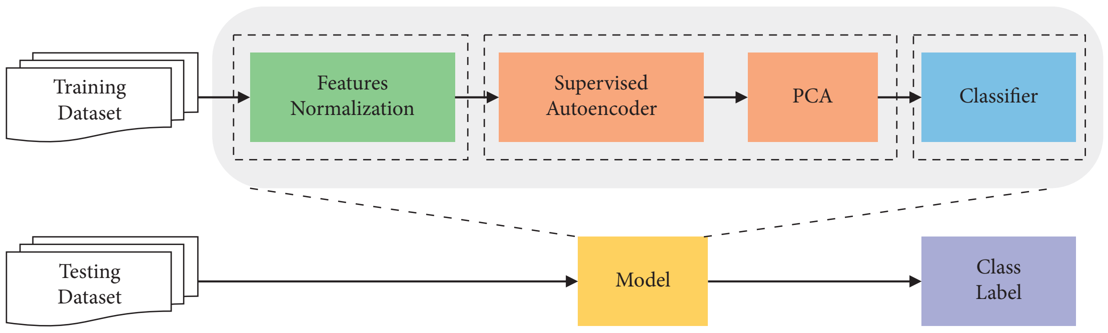

Figure 3: The whole framework of the proposed IDS.

remote tripping command injection, and relay setting change. The data injection attack is attackers try to blind the operator by sending a fake alert. Relay setting change is attacker alters the relay setting to disable their function. Remote tripping command injection is the attacker sends a command to make a breaker open. There are 15 files in the data set. On average, each file has around 5,300 samples.

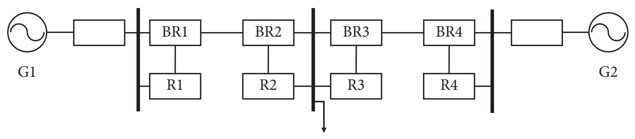

Figure 4: The experiment power system framework configuration.

4.2. Evaluation Metrics. The metrics utilized in classification tasks are used to evaluate our technique in this study. When an attack class is used as a positive class, four different types of classification results are displayed below:

(i) True positive (TP): the attack sample is correctly classified as attack class

(ii) False positive (FP): normal sample is wrongly classified as attack class

(iii) True negative (TN): normal sample is classified as normal correctly

(iv) False negative (FN): one attack sample is classified as normal

We calculate performance measures using formulas as follows based on the classification results provided above:

```
    Data: Training features $\mathbf{x}_i$ with label $y_i$. Hyperparameter $\alpha$, the number of iteration $t$.
    Result: Parameters of dimension reduction module and reduced features
    // Step 1: Preprocessing the training dataset
(1) Normalize data $\mathbf{x}_i$ with Min-Max normalization by $\widetilde{x}_i = (x_i - x_{\min})/(x_{\max} - x_{\min})$
    // Step 2: Training SupervisedAE with the normalized dataset
(2) while not converge do
(3)         $t \leftarrow t + 1$
(4)         Compute the joint loss by $L = L_R + \alpha L_C$
(5)         Train SupervisedAE using the joint loss and update the parameters
(6)   End
    // Step 3: Computing the latent representation $\mathbf{z}_i$ by encoder function
(7) $\mathbf{z}_i = \phi(\widetilde{x}_i)$
    // Step 4: Reducing the dimension of latent representations $\mathbf{z}$ by PCA
(8) $\widehat{\mathbf{z}} = \mathrm{PCA}(\mathbf{z})$
```

Algorithm 1: Training process of dimension reduction module.

Table 1: Features and descriptions.

Feature	Description
PA1:VH–PA3:VH	Phase A–C voltage phase angle
PM1:V–PM3:V	Phase A–C voltage phase magnitude
PA4:IH–PA6:IH	Phase A–C current phase angle
PM4:I–PM6:I	Phase A–C current phase magnitude
PA7:VH–PA9:VH	Pos.–Neg.–zero voltage phase angle
PM7:V–PM9:V	Pos.–Neg.–zero voltage phase magnitude
PA10:VH–PA12:VH	Pos.–Neg.–zero current phase angle
PM10:V–PM12:V	Pos.–Neg.–zero current phase magnitude
F	Frequency for relays
DF	Frequency delta (dF/dt) for relays
PA:Z	Appearance impedance for relays
PA:ZH	Appearance impedance angle for relays
S	Status flag for relays

$$\begin{aligned}\text{Accuracy} &= \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}},\\ \text{Precision} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}},\\ \text{Recall} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}},\\ F1 &= \frac{2*\text{Precision}*\text{Recall}}{\text{Precision}+\text{Recall}}.\end{aligned} \tag{8}$$

Because the data set contains a large number of classes, we calculate the results for all of them. After that, we calculate the average results for the overall performance.

4.3. Results. We use Python programming language to implement the proposed method on our machine that consists of Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40 GHz, three NVIDIA Tesla P100 PCIe 12 GB graphics cards, and 128 GB of RAM. To create neural network models, we use the Keras framework [40]. The scikit-learn library [41] is used to implement machine learning methods, including classifiers and dimension reduction techniques. Apart from the autoencoder and its variants, there are two other techniques (PCA and LDA) involved in comparing the dimension reduction performance with our methods.

LDA [42], compared with the previously mentioned PCA, is a supervised dimension reduction method. Unlike PCA, LDA aims to find a projection function that minimizes within-class distance while maximizing between-class distance simultaneously. Four classifiers are applied to classify samples and evaluate the performance of our dimension reduction technique. The classifiers are described as follows:

(i) K-nearest neighbors (KNN): KNN is a supervised classification method. It is a nonparametric classifier. It uses the majority class of K-nearest neighbors assigned to the test samples.

(ii) Decision tree (DT): DT is a tree-based classifier. The internal node represents an if rule accounts for an attribute. The leaf node represents the final output label.

(iii) Adaptive boosting (AdaBoost): AdaBoost is an ensemble algorithm. It constructs a strong classifier from a collection of weak classifiers. The construction method is sequential. The new model is created to correct the error from the last model.

(iv) Bagging: Bagging is an ensemble learning algorithm also. A number of the base classifier is trained on different data sets. Bootstrap resampling is used to create the data sets.

We employ DT as the base classifier for the AdaBoost and Bagging. In the hidden layers of the neural network, we apply the ReLU activation function [43]. Adam optimization [44] is used to train the neural network. The neuron number settings of a neural network are hard to select. We employ three hidden layers for the SupervisedAE since a single layer autoencoder is too shallow to learn a better representation. Some other hyperparameters are listed in Table 3. The hyperparameters of other autoencoders have the same settings as the SupervisedAE.

After extracting the latent representations from SupervisedAE, the PCA algorithm is used to reduce dimensions further, and various classifiers are trained to classify samples. To select optimal hyperparameters for these models and evaluate the performance, we use the nested tenfold cross-validation procedure. In the process of nested cross-validation, there are two loops: the inner loop and the outer loop. The outer loop splits the data set into a training set and a testing set with a ratio of 9:1. The inner loop uses the classical cross-validation procedure to select optimal hyperparameters on the training set. And then the outer loop evaluates the performance using optimal hyperparameters derived from the inner loop. Both the inner loop and the outer loop are repeated ten times. The optimization target in the inner loop is accuracy in this study. The final performance is the averaged results calculated on the testing set. Table 4 displays the detailed hyperparameter settings. For PCA and LDA algorithms, we vary the number of components from 1 to 16. It should be noted that the PCA used in our proposed model has the same setting. Some hyperparameters for the classifier are selected from a collection of values.

Next, we compare our proposed dimension reduction technique that leverages the SupervisedAE and PCA, with several other dimension reduction methods. The performance results are compared to other existing detection methods then. Finally, we conduct the performance analysis of hyperparameter settings.

Table 2: Event types and class labels.

Event type	Description	Labels
Normal event	Normal operation	41
Natural events	SLG faults	1–6
	Line maintenance	13–14
Attack events	Data injection	7–12
	Remote tripping command injection	15–20
	Relay setting change	21–30, 35–40

Table 3: Hyperparameters of SupervisedAE.

Hyperparameter	Value
Hidden layers	96-64-96
Learning rate	0.001
Batch size	32
Epochs	400
Weight of classification error	1.0

Table 4: Hyperparameter settings in the experiments. Except for the hyperparameters of SupervisedAE, the hyperparameter names of machine learning techniques are consistent with the function of the scikit-learn library. The hyperparameter "None" of DT indicates that the default settings will be used.

Model	Hyperparameter	Value
PCA	n_components	[1, 16]
LDA	n_components	[1, 16]
KNN	n_neighbors	{1,3,5,7,9}
DT	max_depth	{2,4,6,8,10,None}
AdaBoost	n_estimators	{20,40,60}
Bagging	n_estimators	{20,40,60}

4.3.1. Comparisons with Other Methods.

4.3.1. Comparisons with Other Methods. The performance of our suggested technique is compared with the performance of classifiers utilizing original features first. The purpose of the comparisons is to show that our suggested technique achieves the desired dimension reduction results. The accuracy and F1 score are displayed in Tables 5 and 6, respectively.

In Table 5, we present the accuracy under three conditions: no dimension reduction, SupervisedAE only, and SupervisedAE combined with PCA. There are four classifiers for each condition. When using the original features, the Bagging approach achieves the best accuracy with 0.8986. The KNN classifier comes in second with an accuracy of 0.8691. The results of the other two methods are lower. Each of the four classifiers improves differently while using reduced features extracted by the SupervisedAE. The KNN classifier, in particular, has the best accuracy of 0.9365. It has an increase of around 0.067 when compared to the results utilizing original features. The DT and AdaBoost have both increased by roughly 0.04. The Bagging is raised by roughly 0.028, which is smaller than the change in KNN. The additional PCA reduces the dimension even further (from 64 to some value in 1–16), as can be seen in the table, yet the performance does not suffer significantly. On some specific data sets, the model performs worse than without PCA when using the KNN classifier, but the overall average results are comparable. Both methods yield more stable results than using original features as the standard deviation illustrates. The KNN classifier has a standard deviation of just 0.005.

The F1 score displayed in Table 6 can draw a similar conclusion. As previously stated, the nested tenfold cross-validation procedure chooses hyperparameters based on accuracy; hence, the F1 score is slightly lower than accuracy. In conclusion, when compared to the classifiers using original features, our suggested technique successfully reduces the dimension of the features. Furthermore, all four classifiers have better performance in terms of accuracy and F1 score when using the reduced features. The additional PCA method decreases the dimension further without sacrificing too much performance.

To further verify the effectiveness of our method, we compare the corresponding results with various dimension reduction methods in Figures 5 and 6. In the figures, the mean values of accuracy and F1 score calculated over 15 data sets are presented. We compare all of the dimension reduction methods using four classifiers. Dimension reduction methods include PCA, LDA, AE, and SparseAE. The "None" means that the classifiers are trained on the original features.

Table 5: Comparisons of accuracy. The first two rows are dimension reduction methods and classifiers, respectively. The data set number is shown in the first column. The last two rows provide the mean value and standard deviation of the results. “None” indicates using the original features.

No.	None				SupervisedAE				SupervisedAE + PCA			
	KNN	DT	AdaBoost	Bagging	KNN	DT	AdaBoost	Bagging	KNN	DT	AdaBoost	Bagging
1	0.8635	0.8276	0.8200	0.8963	0.9265	0.8699	0.8679	0.9255	0.9303	0.8689	0.8661	0.9211
2	0.8580	0.8398	0.8382	0.9041	0.9256	0.8631	0.8712	0.9148	0.9313	0.8674	0.8611	0.9102
3	0.8792	0.8569	0.8556	0.9034	0.9324	0.8702	0.8696	0.9250	0.9278	0.8735	0.8753	0.9151
4	0.8716	0.8358	0.8406	0.9022	0.9385	0.8766	0.8766	0.9275	0.9350	0.8658	0.8660	0.9248
5	0.8727	0.8318	0.8347	0.9014	0.9320	0.8690	0.8615	0.9198	0.9316	0.8617	0.8553	0.9173
6	0.8712	0.8474	0.8496	0.9108	0.9426	0.8863	0.8800	0.9360	0.9418	0.8824	0.8842	0.9330
7	0.8699	0.8382	0.8281	0.8923	0.9332	0.8642	0.8753	0.9278	0.9318	0.8803	0.8684	0.9158
8	0.8666	0.8551	0.8517	0.9101	0.9332	0.8707	0.8745	0.9223	0.9340	0.8766	0.8715	0.9208
9	0.8577	0.8174	0.8208	0.8798	0.9390	0.8629	0.8672	0.9257	0.9348	0.8633	0.8663	0.9213
10	0.8729	0.8373	0.8391	0.8944	0.9427	0.8872	0.8813	0.9325	0.9382	0.8775	0.8836	0.9307
11	0.8756	0.8235	0.8244	0.8983	0.9444	0.8865	0.8915	0.9370	0.9461	0.8835	0.8867	0.9313
12	0.8773	0.8267	0.8245	0.8951	0.9412	0.8786	0.8754	0.9244	0.9418	0.8800	0.8786	0.9242
13	0.8668	0.8425	0.8397	0.9034	0.9418	0.8812	0.8816	0.9275	0.9448	0.8744	0.8831	0.9290
14	0.8659	0.8282	0.8272	0.8972	0.9363	0.8762	0.8704	0.9320	0.9396	0.8751	0.8739	0.9247
15	0.8681	0.8215	0.8254	0.8906	0.9378	0.8753	0.8791	0.9263	0.9375	0.8789	0.8751	0.9229
Mean	0.8691	0.8353	0.8346	**0.8986**	**0.9365**	0.8745	0.8749	0.9269	**0.9364**	0.8740	0.8730	0.9228
Std	0.0061	0.0114	0.0111	0.0076	0.0056	0.0080	0.0071	0.0056	0.0053	0.0068	0.0089	0.0063

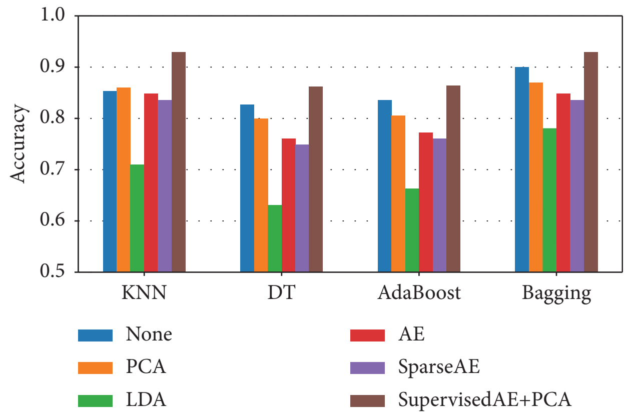

Figure 5: Comparisons of accuracy.

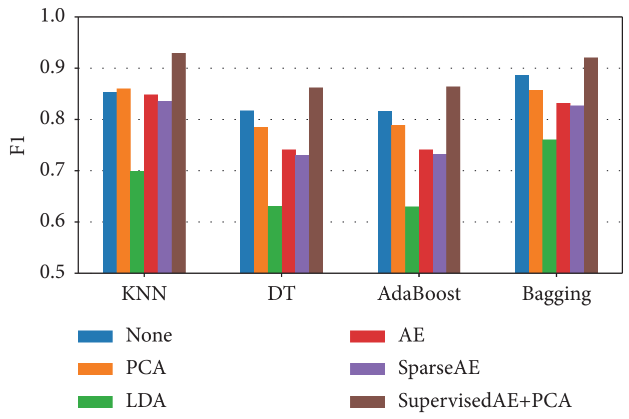

Figure 6: Comparisons of F1 score.

The accuracy results are shown in Figure 5. For the four classifiers, the LDA yields the worst results. The performance is lower than when utilizing original features. When employing the KNN classifier, PCA produces better results than when using original features, indicating that the dimension of features is properly reduced. For other classifiers, PCA has poor results. For all four classifiers, AE and SparseAE have similar results. SparseAE has slightly poorer performance than AE. However, their results are inferior to PCA. In theory, they should be able to achieve similar results to PCA. The experimental result here may be caused by insufficient training of AE. But the training settings are the same with our proposed method, which illustrates the improvement of our method. In comparison to other dimension reduction methods, our proposed technique that combines SupervisedAE and PCA shows promising results. The F1 score in Figure 6 can draw a similar conclusion.

From the tables and figures above, we can observe that the KNN classifier has the best performance. We utilize it as the final classifier to compare with some other existing attack detection methods. There are four distinct baselines. RSKNN [10] and RSRT [9] are the first two baselines. These two techniques use the random subspace method to build a large number of trained classifiers. The majority voting rule is used to determine the final result. KNN and random tree are used as the base classifier. CFS-MLP [27] and CFS-KNN [27] are the remaining two baselines. These methods use a correlation-based feature selection method first and then train MLP and KNN classifier on the selected features, respectively. The comparisons of accuracy and F1 score are displayed in Table 7.

With an accuracy of 0.9188 and an F1 of 0.9187, CFS-KNN outperforms the other three baselines. The second one is RSRT, which has an accuracy of 0.9131. Table 7 confirms that our suggested method has the best accuracy and F1 score. When compared to the best baseline CFS-KNN, the accuracy of our method is 0.018 higher than it. Furthermore, our method has a low standard deviation, indicating that its performance is stable. In conclusion, when compared to other detection methods, our technique yields the best results.

Table 6: Comparisons of F1 score. This table has same layout as last one.

No	None				SupervisedAE				SupervisedAE + PCA			
	KNN	DT	AdaBoost	Bagging	KNN	DT	AdaBoost	Bagging	KNN	DT	AdaBoost	Bagging
1	0.8468	0.8040	0.7976	0.8851	0.9217	0.8633	0.8626	0.9203	0.9290	0.8631	0.8622	0.9182
2	0.8438	0.8240	0.8211	0.8952	0.9182	0.8516	0.8618	0.9069	0.9238	0.8574	0.8480	0.9015
3	0.8676	0.8449	0.8373	0.8970	0.9288	0.8590	0.8595	0.9209	0.9234	0.8673	0.8663	0.9117
4	0.8587	0.8160	0.8295	0.8885	0.9322	0.8649	0.8692	0.9247	0.9284	0.8559	0.8558	0.9206
5	0.8644	0.8250	0.8306	0.8988	0.9291	0.8632	0.8586	0.9198	0.9303	0.8587	0.8540	0.9172
6	0.8649	0.8359	0.8431	0.9063	0.9377	0.8804	0.8758	0.9326	0.9360	0.8761	0.8791	0.9315
7	0.8459	0.8129	0.8000	0.8808	0.9218	0.8454	0.8608	0.9227	0.9188	0.8689	0.8553	0.9072
8	0.8490	0.8401	0.8420	0.9018	0.9262	0.8582	0.8645	0.9143	0.9270	0.8641	0.8634	0.9149
9	0.8388	0.7996	0.8028	0.8700	0.9333	0.8494	0.8571	0.9230	0.9334	0.8504	0.8606	0.9186
10	0.8521	0.8156	0.8117	0.8769	0.9361	0.8675	0.8648	0.9232	0.9296	0.8618	0.8692	0.9235
11	0.8644	0.8113	0.8068	0.8876	0.9384	0.8812	0.8806	0.9317	0.9411	0.8742	0.8727	0.9263
12	0.8574	0.8041	0.8072	0.8763	0.9337	0.8605	0.8602	0.9169	0.9340	0.8629	0.8672	0.9161
13	0.8598	0.8327	0.8301	0.8948	0.9412	0.8750	0.8784	0.9265	0.9442	0.8664	0.8796	0.9257
14	0.8563	0.8123	0.8120	0.8868	0.9300	0.8656	0.8597	0.9273	0.9330	0.8677	0.8634	0.9221
15	0.8554	0.8014	0.8007	0.8763	0.9315	0.8638	0.8686	0.9205	0.9307	0.8676	0.8625	0.9191
Mean	0.8550	0.8187	0.8182	**0 . 8881**	**0 . 9307**	0.8633	0.8655	0.9221	**0 . 9308**	0.8642	0.8640	0.9183
Std	0.0084	0.0140	0.0155	0.0103	0.0064	0.0099	0.0072	0.0063	0.0064	0.0066	0.0086	0.0073

4.3.2. Analysis of Hyperparameters. In this section, we analyze the influence of different hyperparameters. During the experiment, the hidden layer settings of SupervisedAE and the number of PCA components are most important. We compare the different hidden layer settings first and then show the influence of various PCA components.

In general, choosing optimal hidden layer settings for neural networks is difficult because the training time is longer than traditional machine learning methods and the search space of the hyperparameter is huge. As previously stated, we set the hidden layers to three, as one layer is difficult to obtain a better representation, and adding additional layers would increase the training time. Next, we use six combinations to demonstrate the differences, and the results are displayed in Table 8. Except for the neuron numbers in the hidden layer, all of the experiments in this part have identical settings. Also, all of these use the PCA algorithm to reduce the dimension of features further.

Table 8 confirms that when there is only one layer, the performance is poor. Its results are the lowest, especially when "32" is used. The performance improves as the number of layers increases. The accuracy of three layers, "64-32-64," increases by 0.02 than one layer setting of "96." In addition, when the neuron number of three layers is "96-32-96," the accuracy is lower than when it is "96-64-96." This is most likely related to the fast decline of neuron numbers. The F1 score can also reach the same conclusions. In conclusion, the three hidden layers have reached desired results.

In the above experiments, the optimal number of PCA components is chosen by nested cross-validation. But it is necessary to demonstrate performance with the different number of PCA components. In this part, we conduct experiments involving two conditions. The two conditions differ in whether to use SupervisedAE. The condition "With SupervisedAE" means that the SupervisedAE and PCA are combined to reduce dimension. The condition "Without SupervisedAE" means that only the PCA algorithm is used to reduce dimension. We plot the relationship between accuracy and the number of components in Figure 7. It includes four classifiers. After combining with two conditions, there are eight lines in the figure. The different number of PCA components are represented by the x-axis. As previously stated, we set the number of components from 1 to 16. The y-axis represents the accuracy values.

The accuracy of all classifiers in Figure 7 presents an increasing trend as the PCA components rise. First, we analyze the condition of employing the SupervisedAE. The accuracy gradually improves and eventually becomes stable. The accuracy is poor with a lower PCA component. When it is 1, the accuracy for all lines is approximately 0.30. As it is increased to 2, the accuracy improves dramatically, reaching about 0.80. The accuracy becomes stable when the

Table 7: Performance comparisons with other methods.

Method	Accuracy	F1 score
RSKNN [10]	0.9012 ± 0.0061	—
RSRT [9]	0.9131 ± 0.0052	—
CFS-MLP [27]	0.6266 ± 0.0740	0.6233 ± 0.0752
CFS-KNN [27]	0.9188 ± 0.0133	0.9187 ± 0.0132
SupervisedAE	0.9365 ± 0.0056	0.9307 ± 0.0064
SupervisedAE + PCA	0.9364 ± 0.0053	0.9308 ± 0.0064

Table 8: The performance of various hidden layer settings. Both accuracy and F1 score results are included in the table.

Hidden layer settings	Accuracy	F1 score
32	0.9061 ± 0.0055	0.8967 ± 0.0066
64	0.9108 ± 0.0067	0.9027 ± 0.0080
96	0.9125 ± 0.0044	0.9047 ± 0.0060
64-32-64	0.9340 ± 0.0059	0.9284 ± 0.0066
96-32-96	0.9328 ± 0.0063	0.9263 ± 0.0075
96-64-96	0.9364 ± 0.0053	0.9308 ± 0.0064

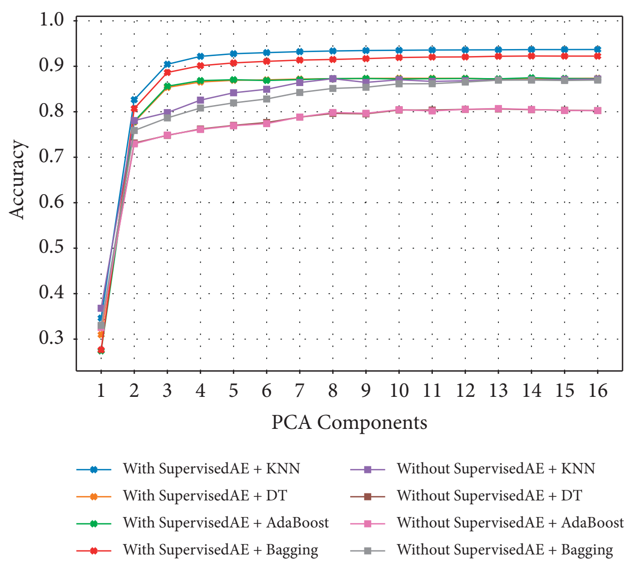

FIGURE 7: Accuracy comparisons of the different number of PCA components.

number of components is increased to 9–10. The KNN and Bagging classifiers achieve higher results. However, the mean accuracy values of Bagging are lower than the KNN classifier. This is consistent with the conclusion from Table 5. The AdaBoost and DT classifiers produce nearly identical results. In the figure, these two plot lines are overlapping.

When using the PCA algorithm to reduce the dimensions only, the accuracy of classifiers improves as the number of PCA components grows also. When the number of components reaches 10, the performance remains stable. The highest accuracy is achieved by KNN and Bagging classifiers. The DT and AdaBoost classifiers produce identical accuracy as the lines of these two methods are overlapping. But the performance is lower than that with SupervisedAE. For the sake of simplicity and space-saving, we have omitted the F1 figure, as the figures for these two metrics are nearly identical. In conclusion, the additional PCA algorithm produces stable results when the number of components is higher. In the experiments, the nested cross-validation would choose the optimal number of PCA components.

5. Conclusions and Future Work

Since many attack events that aim at ICSs have been reported, the security issues of ICSs are becoming increasingly important. To provide security protection, IDS examines data records in the ICSs and raises alerts for malicious operations. Especially, IDS based on machine learning and deep learning has been investigated thoroughly. However, high-dimensional data still poses a challenge. To improve the performance of IDS, we propose a dimension reduction technique based on SupervisedAE and PCA algorithm in this study.

This research uses an improved autoencoder named SupervisedAE by introducing label information during the training time. In this way, the new autoencoder model is trained with reconstruction error and classification error simultaneously. Compared with the conventional autoencoder, the SupervisedAE obtains more discriminative latent representations. The experiment results show that the performance of classifiers trained on latent representations extracted from SupervisedAE outperforms that trained on original features. In addition, the PCA algorithm is applied to reduce the dimension of features further. When compared to other dimension reduction methods, our combined technique performs best. The KNN classifier, in particular, yields the greatest results. Furthermore, when compared to other existing detection methods, the suggested technique has higher accuracy and F1 score, demonstrating its efficacy.

In the future, there are several directions to extend our work. Some other variants of autoencoder are worth investigating such as denoising autoencoder, and variational autoencoder. Since we focus on the dimension reduction technique in this paper, only a few machine learning classifiers (e.g., KNN) are used. To boost performance even further, more complicated classifiers could be applied. In addition, we only test the dimension reduction technique on the power system data set in this work. It is critical to test the technique on more ICS data sets to verify its efficacy and robustness.

Data Availability

The data set we used in this paper is available at https://sites.google.com/a/uah.edu/tommy-morris-uah/ics-data-sets. Readers who are interested in our research can access the data set and reproduce our results.

Conflicts of Interest

All the authors hereby declare that there are no conflicts of interest.

Acknowledgments

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

References

[1] K. Stouffer, V. Pillitteri, S. Lightman, M. Abrams, and A. Hahn, *Guide to Industrial Control Systems (ICS) Security*, National Institute of Standards and Technology, Gaithersburg, Md, USA, 2015.

[2] T. Alladi, V. Chamola, and S. Zeadally, "Industrial control systems: cyberattack trends and countermeasures," *Computer Communications*, vol. 155, pp. 1–8, 2020.

[3] R. Langner, "Stuxnet: dissecting a cyberwarfare weapon," *IEEE Security and Privacy Magazine*, vol. 9, no. 3, pp. 49–51, 2011.

[4] R. Khan, P. Maynard, K. McLaughlin, D. Laverty, and S. Sezer, "Threat analysis of blackenergy malware for synchrophasor based real-time control and monitoring in smart grid," in *Proceedings of the 4th international symposium for ICS &*

SCADA cyber security research 2016, pp. 53–63, Swindon, UK, August 2016.

[5] A. Cherepanov and R. Lipovsky, "Industroyer: Biggest threat to industrial control systems since stuxnet," *WeLiveSecurity, ESET*, vol. 12, 2017.

[6] A. Terai, S. Abe, S. Kojima, Y. Takano, and I. Koshijima, "Cyber-attack detection for industrial control system monitoring with support vector machine based on communication profile," in *Proceedings of the 2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pp. 132–138, Paris, France, April 2017.

[7] S. Sivakumar, B. S. Ananthanarayanan, and U. Arumugam, "Intrusion detection system for securing the SCADA industrial control systemLecture Notes on Data Engineering and Communications Technologies," in *Proceedings of the International Conference on Computational Intelligence, Data Science and Cloud Computing*, pp. 645–658, Kolkata, India, September 2021.

[8] J. Ling, Z. Zhu, Y. Luo, and H. Wang, "An intrusion detection method for industrial control systems based on bidirectional simple recurrent unit," *Computers & Electrical Engineering*, vol. 91, Article ID 107049, 2021.

[9] M. M. Hassan, A. Gumaei, S. Huda, and A. Almogren, "Increasing the trustworthiness in the industrial IoT networks through a reliable cyberattack detection model," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 6154–6162, 2020.

[10] A. Derhab, M. Guerroumi, A. Gumaei et al., "Blockchain and random subspace learning-based IDS for SDN-enabled industrial IoT security," *Sensors*, vol. 19, no. 14, p. 3119, 2019.

[11] S. I. Popoola, R. Ande, B. Adebisi, G. Gui, M. Hammoudeh, and O. Jogunola, "Federated deep learning for zero-day botnet attack detection in IoT-edge devices," *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3930–3944, 2022.

[12] S. I. Popoola, B. Adebisi, M. Hammoudeh, G. Gui, and H. Gacanin, "Hybrid deep learning for botnet attack detection in the internet-of-things networks," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4944–4956, 2021.

[13] R. Zhao, J. Yin, Z. Xue et al., "An efficient intrusion detection method based on dynamic autoencoder," *IEEE Wireless Communications Letters*, vol. 10, no. 8, pp. 1707–1711, 2021.

[14] R. Zhao, G. Gui, Z. Xue et al., "A novel intrusion detection method based on lightweight neural network for internet of things," *IEEE Internet of Things Journal*, p. 1, 2021, https://ieeexplore.ieee.org/document/9566308.

[15] A. Thakkar and R. Lohiya, "A Survey on Intrusion Detection System: Feature Selection, Model, Performance Measures, Application Perspective, Challenges, and Future Research Directions," *Artificial Intelligence Review*, vol. 55, no. 1, pp. 453–563, 2021.

[16] A. A. Aburomman and M. Bin Ibne Reaz, "Ensemble of binary SVM classifiers based on PCA and LDA feature extraction for intrusion detection," in *Proceedings of the 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, pp. 636–640, Xi'an, China, Octo 2016.

[17] K. Ibrahimi and M. Ouaddane, "Management of intrusion detection systems based-KDD99: analysis with LDA and PCA," in *Proceedings of the 2017 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pp. 1–6, Rabat, Morocco, Nove 2017.

[18] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep learning approach combining sparse autoencoder with SVM for network intrusion detection," *IEEE Access*, vol. 6, pp. 52843–52856, 2018.

[19] G. Yadav and K. Paul, "Architecture and security of SCADA systems: a review," *International Journal of Critical Infrastructure Protection*, vol. 34, Article ID 100433, 2021.

[20] M. R. Gauthama Raman, C. M. Ahmed, and A. Mathur, "Machine learning for intrusion detection in industrial control systems: challenges and lessons from experimental evaluation," *Cybersecurity*, vol. 4, no. 1, p. 27, 2021.

[21] M. A. Ferrag, M. Babaghayou, and M. A. Yazici, "Cyber security for fog-based smart grid SCADA systems: solutions and challenges," *Journal of Information Security and Applications*, vol. 52, Article ID 102500, 2020.

[22] A. A. Süzen, "Developing a multi-level intrusion detection system using hybrid-DBN," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 2, pp. 1913–1923, 2021.

[23] H. Alazzam, A. Sharieh, and K. E. Sabri, "A feature selection algorithm for intrusion detection system based on pigeon inspired optimizer," *Expert Systems with Applications*, vol. 148, Article ID 113249, 2020.

[24] Z. Chkirbene, A. Erbad, R. Hamila, A. Mohamed, M. Guizani, and M. Hamdi, "TIDCS: a dynamic intrusion detection and classification system based feature selection," *IEEE Access*, vol. 8, pp. 95864–95877, 2020.

[25] X. Liu, T. Li, R. Zhang, D. Wu, Y. Liu, and Z. Yang, "A GAN and feature selection-based oversampling technique for intrusion detection," *Security and Communication Networks*, vol. 2021, Article ID 9947059, 15 pages, 2021.

[26] I. M. Akashdeep, I. Manzoor, and N. Kumar, "A feature reduced intrusion detection system using ANN classifier," *Expert Systems with Applications*, vol. 88, pp. 249–257, 2017.

[27] A. Gumaei, M. M. Hassan, S. Huda et al., "A robust cyberattack detection approach using optimal features of SCADA power systems in smart grids," *Applied Soft Computing*, vol. 96, Article ID 106658, 2020.

[28] S. P. R.M., P. K. R. Maddikunta, P. M et al., "An effective feature engineering for DNN using hybrid PCA-GWO for intrusion detection in IoMT architecture," *Computer Communications*, vol. 160, pp. 139–149, 2020.

[29] F. Salo, A. B. Nassif, and A. Essex, "Dimensionality reduction with IG-PCA and ensemble classifier for network intrusion detection," *Computer Networks*, vol. 148, pp. 164–175, 2019.

[30] F. Kuang, W. Xu, and S. Zhang, "A novel hybrid KPCA and SVM with GA model for intrusion detection," *Applied Soft Computing*, vol. 18, pp. 178–184, 2014.

[31] H. Deng and T. Yang, "Network intrusion detection based on sparse autoencoder and IGA-BP network," *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 9510858, 11 pages, 2021.

[32] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT press, Cambridge, MA, USA, 2016.

[33] A. Ng, "Sparse autoencoder," *CS294A Lecture notes*, vol. 72, pp. 1–19, 2011.

[34] A. Gogna and A. Majumdar, "Discriminative autoencoder for feature extraction: application to character recognition," *Neural Processing Letters*, vol. 49, no. 3, pp. 1723–1735, 2019.

[35] L. Le, A. Patterson, and M. White, "Supervised autoencoders: improving generalization performance with unsupervised regularizers," in *Proceedings of the Advances in neural information processing systems*, Montréal, Canada, December 2018.

[36] A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to*

Build Intelligent Systems, O'Reilly Media, Sebastopol, CA, USA, 2019.

[37] Y. Wang, M. Liu, Z. Bao, and S. Zhang, "Stacked sparse autoencoder with PCA and SVM for data-based line trip fault diagnosis in power systems," *Neural Computing & Applications*, vol. 31, no. 10, pp. 6719–6731, 2019.

[38] L. I. Smith, "A Tutorial on Principal Components Analysis," 2002, https://www.cs.cmu.edu/~elaw/papers/pca.pdf.

[39] J. Beaver, R. Borges, M. Buckner, T. Morris, U. Adhikari, and U. Pan, "Machine learning for power system disturbance and cyber-attack discrimination," in *Proceedings of the 7th International Symposium on Resilient Control Systems*, Denver, CO, USA, August 2014.

[40] F. Chollet, "Keras," 2015, https://keras.io/.

[41] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, and E. Duchesnay, Scikit-learn: machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[42] A. M. Martinez and A. C. Kak, "Pca versus lda," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 2, pp. 228–233, 2001.

[43] Y. Li and Y. Yuan, "Convergence Analysis of Two-Layer Neural Networks with Relu Activation," 2017, https://arxiv.org/abs/1705.09886.

[44] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," 2017, https://arxiv.org/abs/1412.6980.