

# Anomaly Detection by Generative Adversarial Networks with AdaBoost Ensemble Learning

Mingjing Xu<sup>a</sup>, Piero Baraldi<sup>a,\*</sup>, Xuefei Lu<sup>b</sup>, Enrico Zio<sup>a,c,d</sup>

<sup>a</sup>*Energy Department, Politecnico di Milano, Via La Masa 34, 20156 Milan, Italy*

<sup>b</sup>*University of Edinburgh Business School, 29 Buccleuch Place, Edinburgh, EH8 9JS, UK*

<sup>c</sup>*MINES ParisTech, PSL Research University, CRC, Sophia Antipolis, France*

<sup>d</sup>*Aramis Srl, Via pergolesi 5, Milano, Italy*

---

## Abstract

Due to the scarcity of abnormal condition data in industrial applications, one-class classification models trained using only normal condition data are typically considered for anomaly detection. The development of these models for practical use is challenged by the complexity of the distribution of the normal condition data, which is typically non-smooth, multidimensional and characterized by long-term temporal dependencies. Inspired by the idea of Generative Adversarial Networks (*GANs*), this work develops an anomaly detection model based on the use of an Auto-Encoder (*AE*) formed by the generator of a *GAN* and an auxiliary encoder. The reconstruction error of the *AE* is, then, used as anomaly score to detect anomalies. The addition of an adaptive noise to the data and the development of an AdaBoost-based ensemble learning scheme to detect anomalies in small time slices of multivariate time series are the main methodological novelties proposed in this work. Also, the *AE-GAN* model hyperparameters are optimized without the need of performing trial and error approaches on test data, by defining a lower bound of the Jensen-Shannon divergence between generator and normal data distributions. Extensive experiments on synthetic and real industrial datasets show that the proposed ensembled *AE-GAN* model outperforms other state-of-the-art anomaly detection methods.

*Keywords:* Anomaly Detection, One-Class Classification, Long-term Multivariate Time Series, Auto Encoder, Generative Adversarial Networks, AdaBoost Ensemble Learning

---

## 1. Introduction

Anomaly detection aims at identifying novel and unexpected patterns within the data collected [1, 2]. It plays a critical role in several industrial domains, such as network intrusion detection [3, 4], transportation monitoring [5, 6], video anomalous behavior recognition [7] and component fault diagnostics [2, 8, 9, 10]. This latter application is made possible by the fact that industrial components are equipped with sensors that measure a variety of signals for the control of their operation and the monitoring of their behavior: signal observations which deviate from regular observations can indicate a shift in the behavior of the component, caused by the occurrence of anomalous conditions, e.g. deterioration or damage [11, 12, 13, 14, 15].

Anomaly detection approaches are typically categorized as supervised, unsupervised and one-class classification [16]. Supervised methods require the availability of a sufficient number of signal measurements labelled with the information on the component health state, i.e. normal or anomalous. However, this is rare in practice and supervised methods are often impractical in many industrial applications [17]. Unsupervised methods do not need labelled data, but they typically assume that *i*) a sufficient number of patterns collected in both normal and anomalous conditions is available, *ii*) anomalous condition patterns

---

\*Corresponding author

Email address: [piero.baraldi@polimi.it](mailto:piero.baraldi@polimi.it) (Piero Baraldi)

are sufficiently dissimilar to normal condition patterns to allow discriminating them [18]. However, on the contrary, in many industrial applications anomalous conditions are rare and changes in operating and environmental conditions can cause variations of the measured signals that are larger than the variations caused by the onset of a degradation or the damage of a component, at least at the early stages after its occurrence. For this reason, in this work we consider one-class classification methods [19, 20], which are trained on a dataset containing only normal condition patterns. Among them, Support Vector Machines (*SVMs*) [2], nearest neighbor-based methods [8], statistical models [21] and Deep Learning (*DL*) [22] based approaches are the most employed.

One-Class SVM (*OC-SVM*) defines a kernel to identify the region that fits the distribution of the normal condition data. Then, if a test pattern falls out of the learned region, it is declared as anomalous. A hybrid model combining *OC-SVM* and deep learning has been developed to detect anomalies in high-dimensional and large-scale settings in [23]. A model combining *OC-SVM* with a Self-Organized Feature Map (*SOFM*) has been developed for detecting cyberattacks such as worms and spy-wares [24]. Nearest neighbors-based methods use properly defined measures of dissimilarity among patterns and assume that normal condition data are located in dense neighborhoods, whereas anomalies are far from their closest neighbors [25]. For example, the Auto Associative Kernel Regression (*AAKR*) method has been used to detect anomalous conditions in an energy production plant in [26]. The method is based on the reconstruction of the test pattern as a weighted sum of normal condition patterns, where the weights are proportional to the patterns similarity to the test pattern. Two similarity measures based on the Euclidean distance have been introduced in [26] and [27], respectively. Then, if the reconstruction error exceeds an alarm threshold, the test pattern is identified as abnormal. The Sequential Hausdorff Nearest-Neighbor Conformal Anomaly Detector (*SHNN-CAD*) has been proposed and investigated for online learning and sequential anomaly detection in signal trajectories [28]. Statistical model-based methods such as Gaussian Mixture Models (*GMMs*) and Markov statistics construct probabilistic models describing the normal condition patterns: test patterns are, then, detected as anomalous if their likelihood of occurrence based on the probabilistic model of the normal condition data is low [21]. Markov statistics has been applied with success to anomaly detection in fast streaming temporal data [29]. A deep generative model stacked with multiple *GMM*-layers has been proposed to detect abnormal events in video surveillance in [30]. Deep learning-based anomaly detection methods have recently gained a lot of attention due to their ability of effectively learning the characteristics of complex data, such as multivariate time series. For instance, a Multi-Scale Convolutional Recurrent Encoder-Decoder (*MSCRED*) model has been proposed to perform anomaly detection in power plants [31]. A smoothness-inducing sequential Variational Auto-Encoder (*VAE*) model, combined with Recurrent Neural Networks (*RNNs*) to capture latent temporal structures in time series, has been developed for anomaly detection in multivariate time series [32]. These deep learning-based methods assume that small reconstruction errors are achieved for normal condition data, whereas large reconstruction errors are obtained for anomalous condition patterns [33]. However, detecting anomalies using conventional deep learning methods like *RNNs*, Auto-Encoders and hybrid *DNNs* can be challenging due to the long-term time dependency and cross correlation among time series [34].

Generative Adversarial Network (*GAN*) is a deep learning method which consists of a generator and a discriminator, where the generator is trained to reproduce the training data distribution and the discriminator provides the probability of a new pattern coming from the same training [35]. *GAN* is capable of learning complex distributions, for instance, a low dimensional manifold embedded in high-dimensional space [36]. *GAN*-based anomaly detection techniques were first proposed in [37, 38] for medical image analysis. In the transport field, a data augmentation method has been proposed for synthesizing anomalies of the minority classes in lane detection. The method uses *GAN* to learn the distribution of anomalous condition patterns, and the synthesized anomalies are then used to train a supervised anomaly detection model [6]. However, similarly to the other data augmentation methods, the model cannot be used when abnormal condition patterns are completely missing. A deep End-to-End One-Class Classifier applying the adversarial training technique like *GAN* has been proposed [33], in which the model is composed of an Auto-Encoder and a Discriminator which are trained in an adversarial way: the Auto-Encoder can reconstruct well the normal condition patterns but not the abnormal (unseen) patterns; then, anomalies can be detected by the Discriminator [33].

The objective of this work is to develop a methodology for detecting anomalies in components behaviour using measurements collected from components in normal conditions.

We propose an Auto-Encoder aided *GAN* (*AE-GAN*) to associate an anomaly score to each (small) time slice in the synchronized multidimensional signal time series; then, an ensemble anomaly detector is developed by adapting the AdaBoost ensemble learning scheme to output the final anomaly detection result. More specifically, *AE-GAN* is composed of an encoder network and a *GAN*. Firstly, the *GAN* is trained to obtain a generator which reproduces the distribution of normal condition patterns, i.e. time slices of multivariate time series. Then, the encoder and the trained generator form an Auto-Encoder, which is trained to minimize the reconstruction errors of normal condition patterns (note that the parameters of the generator are fixed during the Auto-Encoder training). The test pattern is declared as anomalous if the Auto-Encoder reconstruction error is larger than a certain threshold. Two *AE-GANs* variants are developed in this work: variant *a*) sets up an individual *AE-GAN* for every time slice, whereas variant *b*) sets up a universal *AE-GAN* for all time slices, which means that the universal *AE-GAN* reconstructs all time slices and obtains their reconstruction errors.

A synthetic case study is worked out to verify the performance of the *AE-GAN*, and a real-world industrial case study is performed to verify the feasibility of the ensemble *AE-GAN* developed by using AdaBoost-based ensemble learning scheme. The synthetic case study considers three complex distributions of normal condition patterns, e.g. Cone, Two-Gaussian Ball and Bowl Manifold distributions, to simulate the realistic difficulties encountered in industry. The real-world industrial case considers automatic doors in high speed trains: data collected during doors opening and closing are used to detect whether the doors are working in normal or anomalous conditions. The proposed method shows superior performance in comparison with state-of-the-art anomaly detection techniques, e.g., *OC-SVM*, *AAKR*, *GMM* and Auto-Encoder.

The contributions of this work are:

- 1) An *AE-GAN* is originally developed to detect anomalies in data characterized by manifold distributions;
- 2) A lower bound of Jensen-Shannon divergence between real data and generator distributions based on *GAN* is defined and used to guide the search for the *AE-GAN* model hyper-parameters;
- 3) The proposed addition of adaptive noise to the data solves anomaly detection problems for data distributions with non-smooth density;
- 4) The proposed AdaBoost-based learning scheme not only outputs the overall anomaly detection result of long-term time series, but also learns the weights for the time slices which contribute to anomaly detection.

The remaining of the paper is organized as follows: Section 2 states the problem and illustrates the work objectives; Section 3 introduces the background and preliminaries of the proposed methodology and Section 4 specializes the proposed methodology of anomaly detection for long-term multivariate time series; Section 5 introduces the numerical synthetic case study with three complex distributions and the real-world industrial case study of the automatic doors in high speed trains, and then discusses the results obtained; finally, some conclusions and remarks are given in Section 6.

## 2. Problem Statement

We consider  $N_{normal}$  components operating in normal conditions. For each component,  $N_f$  features related to its health condition are measured. The  $N_f \times L$  matrix,  $\mathbf{X}^r$ ,  $r = 1, \dots, N_{normal}$ , contains the feature time series data of length  $L$  collected during operation in normal conditions. The aim of this work is to build a detection model to identify the normal/abnormal health state of a test component given the measurements  $\mathbf{X}^{test}$ . The model is developed using the normal condition data  $\mathbf{X}^r$ ,  $r = 1, \dots, N_{normal}$ .

## 3. Preliminary and Background

### 3.1. Generative Adversarial Networks

A *GAN* consists of a generator and a discriminator, where the generator is a multilayer perceptron aiming at regenerating patterns from the data distribution of the training set and the discriminator is a

multilayer perceptron aiming at providing the probability that a generated pattern comes from the same data distribution [35]. The *GAN* architecture is shown in Figure 1.

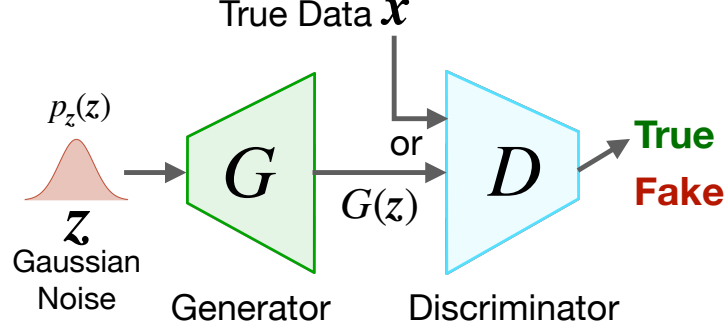


Figure 1: The *GAN* architecture.

The generator  $G(\mathbf{z}; \theta_G) : \mathcal{Z} \rightarrow \mathcal{X}$  with associated parameters  $\theta_G$  maps the latent variable  $\mathbf{z}$  from the latent space  $\mathcal{Z} \subseteq \mathbb{R}^{N_z}$  to the data space of patterns  $\mathcal{X} \subseteq \mathbb{R}^{N_x}$ . We denote the data pattern as  $\mathbf{x}, \mathbf{x} \in \mathcal{X}$ , which follows the probability distribution  $p_{data}$ ; we denote the latent variable as  $\mathbf{z}, \mathbf{z} \in \mathcal{Z}$ ; entries of vector  $\mathbf{z}$  are independent with each other and are usually assumed to follow a standard Gaussian distribution  $\mathcal{N}(0, 1)$ . The discriminator  $D(\mathbf{x}; \theta_D) : \mathcal{X} \rightarrow [0, 1]$ , with associated parameters  $\theta_D$ , discriminates whether the input pattern  $\mathbf{x}$  is true or is a generated pattern, by estimating the probability that  $\mathbf{x}$  comes from the true data distribution  $p_{data}$ . The generator  $G$  is trained to approximate  $p_{data}$ , whereas the discriminator  $D$  is trained to distinguish the training patterns from the patterns generated by  $G$ . Mathematically, the *GAN* is trained by conducting a minmax optimization with loss function  $\mathcal{F}(\theta_D, \theta_G)$ :

$$\min_{\theta_G} \max_{\theta_D} \mathcal{F}(\theta_D, \theta_G) = \mathbb{E}_{\mathbf{x} \sim p_{data}} [\log D(\mathbf{x}; \theta_D)] + \mathbb{E}_{\mathbf{z} \sim p_z} [\log(1 - D(G(\mathbf{z}; \theta_G); \theta_D))] \quad (1)$$

where  $p_z(\mathbf{z})$  is the prior probability distribution function of latent variable  $\mathbf{z}$ .

For any given generator parameter  $\theta_G$ , the optimal discriminator is (Proposition 1 in [35]):

$$D(\mathbf{x}; \theta_D^*(\theta_G)) = \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x}) + p_G(\mathbf{x})} \quad (2)$$

Where  $p_G(\mathbf{x})$  is the data distribution of generated patterns. Notice that the optimal discriminator parameter  $\theta_D^*(\theta_G)$  depends on  $\theta_G$ ; then,  $\theta_D$  in Equation (1) is substituted with the optimal discriminator parameter  $\theta_D^*(\theta_G)$  and the minmax optimization with loss function  $\mathcal{F}(\theta_D, \theta_G)$  becomes [35]:

$$\begin{aligned} \min_{\theta_G} \max_{\theta_D} \mathcal{F}(\theta_D, \theta_G) &= \min_{\theta_G} \mathcal{F}(\theta_D^*(\theta_G), \theta_G) \\ &= \min_{\theta_G} [-\log(4) + \\ &\quad KL(p_{data} \parallel \frac{p_{data} + p_G}{2}) + KL(p_G \parallel \frac{p_{data} + p_G}{2})] \\ &= \min_{\theta_G} [-\log(4) + 2 \cdot JSD(p_{data} \parallel p_G)] \\ &= -\log(4) + 2 \cdot \min_{\theta_G} JSD(p_{data} \parallel p_G) \end{aligned} \quad (3)$$

where  $KL(\cdot \parallel \cdot)$  is the Kullback–Leibler (*K-L*) divergence and  $JSD(\cdot \parallel \cdot)$  is the Jensen–Shannon (*J-S*)

divergence, which measure the dissimilarity between two probability distributions. Given the log base  $e$  in Equation (1), the  $J$ - $S$  divergence is bounded in the range  $[0, \ln(2)]$  and a  $J$ - $S$  divergence equal to 0 indicates that the two distributions are identical.

In practice, the generator is trained by minimizing  $JSD(p_{data} \parallel p_G)$ . However, using *GAN* to generate long-term multi-variate time series is still a challenge [39].

### 3.2. Auto-Encoder

An Auto-Encoder is a neural network composed of an encoder and a generator, trained to replicate its input data [40]. The encoder maps the data space  $\mathcal{X}$  into the latent space  $\mathcal{Z}$ , whereas the generator reconstructs the input data from the latent variable  $\mathbf{z}$ .

A typical form of an encoder  $E$  is a composition of a nonlinear activation function  $f$  and an affine transformation:

$$E(\mathbf{x}; \theta_E) = f(\mathbf{W}_E \mathbf{x} + \mathbf{b}_E) \quad (4)$$

where  $\theta_E = \{\mathbf{W}_E, \mathbf{b}_E\}$  is the set of encoder parameters, with  $\mathbf{W}_E$  being the  $N_z \times N_x$  weight matrix and  $\mathbf{b}_E$  the offset vector of dimension  $N_z$ .

In the generator  $G$ , the resulting latent variable  $\mathbf{z}$  is, then, mapped back to a reconstructed  $N_x$ -dimensional vector  $\hat{\mathbf{x}}$ , whose typical form is similar to  $E$ :

$$G(\mathbf{z}; \theta_G) = f_G(\mathbf{W}_G \mathbf{z} + \mathbf{b}_G) \quad (5)$$

where  $\theta_G = \{\mathbf{W}_G, \mathbf{b}_G\}$  is the set of generator parameters, with  $\mathbf{W}_G$  being the  $N_x \times N_z$  weight matrix and  $\mathbf{b}_G$  the offset vector of dimension  $N_x$ ;  $f_G$  is the nonlinear activation function, e.g.  $\tanh(\cdot)$ .

The Auto-Encoder is trained by minimizing the reconstruction error  $\mathcal{L}_{rec}$ , which quantifies the expected distance between the input vector  $\mathbf{x}$  and its reconstruction  $\hat{\mathbf{x}}$ :

$$\mathcal{L}_{rec} = \mathbb{E}_{\mathbf{x} \in \mathcal{X}} \|\mathbf{x} - \hat{\mathbf{x}}\|^2 = \mathbb{E}_{\mathbf{x} \in \mathcal{X}} \|\mathbf{x} - G(E(\mathbf{x}))\|^2 \quad (6)$$

where ‘ $\|\cdot\|$ ’ denotes the L2 norm.

### 3.3. AdaBoost Ensemble learning

AdaBoost is an ensemble learning algorithm which constructs a boosted classifier as linear combination of several weak classifiers [41, 42]. In practice, the output of the boosted classifier is the weighted sum of the outputs of the weak classifiers. AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers.

For a two-class classification task, suppose that there are  $N$  training patterns  $\mathbf{x}^1, \dots, \mathbf{x}^i, \dots, \mathbf{x}^N$  with target labels  $l^1, \dots, l^i, \dots, l^N, l^i \in \{-1, 1\}$ . The AdaBoost algorithm builds an ensembled classifier  $H : \mathbf{x} \rightarrow \{-1, 1\}$  by linearly combining the base classifiers  $h_t : \mathbf{x} \rightarrow \{-1, 1\}$ :

$$H(\mathbf{x}) = \text{sgn} \left( \sum_{t=1}^T \alpha_t h_t(\mathbf{x}) \right) \quad (7)$$

where  $h_t$  denotes the  $t$ -th base classifier,  $\alpha_t$  denotes the weight assigned to  $h_t$  and  $T$  is the number of base classifiers.

### 3.4. Adam Optimization

The Adam optimization algorithm is an extension to stochastic gradient descent that has recently seen broad adoption for deep learning applications in computer vision and natural language processing [43]. Adam optimization combines the advantages of two other extensions of stochastic gradient descent, specifically:

- **Adaptive Gradient Algorithm (AdaGrad)** that maintains a per-parameter learning rate that improves performance on problems with sparse gradients.

- **Root Mean Square Propagation (RMSProp)** that also maintains per-parameter learning rates that are adapted based on the average of recent magnitudes of the gradients for the weight (e.g. how quickly it is changing); this means that the algorithm does well on non-stationary problems.

The Adam algorithm that realizes the benefits of both AdaGrad and RMSProp is illustrated in Algorithm 1, below.

---

<b>Algorithm 1:</b> Adam Optimization	
<b>Require:</b> Objective function $f(\mathbf{x}; \theta)$ is $\mathcal{F}(\mathbf{x}; \theta_D, \theta_G)$ for $GAN$ and $\mathcal{L}_{rec}(\mathbf{x}; \theta_E, \theta_G)$ for $AE$ , initial parameters $\theta_0 = \{\theta_D, \theta_G\}$ for $GAN$ and $\theta_0 = \{\theta_E, \theta_G\}$ for $AE$ , learning rate $\eta$ , exponential decay rates $\beta_1, \beta_2$ for moment estimates, tolerance parameter $\epsilon=10^{-8}$ for numerical stability, and decision rule for declaring convergence of $\theta_t$ .	
1	$m_0, v_0, t \leftarrow [0, 0, 0]$ // Initialize moment estimates
2	<b>while</b> $\theta_t$ has not converged <b>do</b>
3	$t \leftarrow t + 1$ // Update iteration step
4	$g_t \leftarrow \nabla_{\theta} f(\mathbf{x}; \theta_{t-1})$ // Compute gradient of objective
5	$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ // Update first moment estimate
6	$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ // Update second moment estimate
7	$\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ // Compute bias-corrected first moment estimate
8	$\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ // Compute bias-corrected second raw moment estimate
9	$Adam(\nabla_{\theta} f(\mathbf{x}; \theta_{t-1}); \beta_1, \beta_2) = \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ // Obtain updating term
10	$\theta_t \leftarrow \theta_{t-1} - \eta \cdot Adam(\nabla_{\theta} f(\mathbf{x}; \theta_{t-1}); \beta_1, \beta_2)$ // Update objective parameters
	<b>Return:</b> $\theta_t$ // Resulting parameter

---

## 4. The Proposed Anomaly Detection Methodology

### 4.1. Base Anomaly Detector with Auto Encoder aided Generative Adversarial Networks (AE-GAN)

The proposed base anomaly detector utilizes a deep learning model based on  $GAN$  to reconstruct the patterns and measure the reconstruction error to discriminate the abnormal pattern. The proposed approach includes two main steps: 1) training  $GAN$  on normal condition patterns and 2) query the latent variable of patterns and reconstruct patterns.

In step 1), the  $GAN$  is trained to minimize  $JSD(p_G \parallel p_{\mathcal{X}_{normal}})$ , where  $\mathcal{X}_{normal}$  denotes the normal patterns set and  $p_{\mathcal{X}_{normal}}$  the probability distribution of the normal patterns. Notice that if the generator in  $GAN$  is perfectly trained, then  $JSD(p_G \parallel p_{\mathcal{X}_{normal}})$  converges to 0 [44]. Let  $\theta_G = \{\mathbf{W}_G, \mathbf{b}_G\}$  denote the generator parameter and  $\theta_D = \{\mathbf{W}_D, \mathbf{b}_D\}$  denote the discriminator parameter. For brevity, the generator in  $GAN$  can be formulated as the same of AutoEncoder in Section 3.2, Equation (5). Discriminator  $D$  is formulated as:

$$D(\mathbf{x}; \theta_D) = f_D(\mathbf{W}_D \mathbf{x} + \mathbf{b}_D) \quad (8)$$

where  $\mathbf{W}_D$  is a  $N_z \times N_x$  weight matrix,  $\mathbf{b}_D$  is an offset vector of dimensionality  $N_z$  and  $f_D$  is the nonlinear activation function, e.g.  $f_D = \text{sigmoid}(\cdot)$ . For brevity, a single-layer neural network for  $G$  and  $D$  is illustrated, whereas multiple-layers neural networks are employed in the case study. For the purpose of anomaly detection, in the  $GAN$  loss function (Equation (1)), we set  $p_{data} = p_{\mathcal{X}_{normal}}$  and  $p_z$  as a Gaussian distribution of  $\mathcal{N}(0, 1)$ . Then, the  $GAN$  loss function  $\mathcal{F}$  becomes:

$$\min_{\theta_G} \max_{\theta_D} \mathcal{F}(\theta_D, \theta_G) = \mathbb{E}_{\mathbf{x} \sim p_{\mathcal{X}_{normal}}} [\log D(\mathbf{x}; \theta_D)] + \mathbb{E}_{\mathbf{z} \sim \mathcal{N}(0, 1)} [\log(1 - D(G(\mathbf{z}; \theta_G); \theta_D))] \quad (9)$$

Note that the components of vector  $\mathbf{z}$  are independent of each other. Before the optimization of generator parameter  $\theta_G$ , the discriminator parameter for a given generator,  $\theta_D^*(\theta_G)$ , is obtained firstly by a gradient optimization method based on Adam (Section 3.4):

$$\theta_D^{(k+1)} = \theta_D^{(k)} + \eta \cdot \text{Adam} \left( \nabla_{\theta_D} \mathcal{F}(\theta_D^{(k)}, \theta_G); \beta_1, \beta_2 \right), \quad (10)$$

$$\theta_D^*(\theta_G) = \lim_{k \rightarrow \infty} \theta_D^{(k)}, \quad (11)$$

where the updating term  $\text{Adam} \left( \nabla_{\theta_D} \mathcal{F}(\theta_D^{(k)}, \theta_G); \beta_1, \beta_2 \right)$  is determined by  $\nabla_{\theta_D} \mathcal{F}(\theta_D^{(k)}, \theta_G)$ , the gradient of the loss function  $\mathcal{F}$  with respect to  $\theta_D$ , and  $\beta_1, \beta_2$  are the control parameters of Adam [43],  $\eta$  is the learning rate,  $\theta_D^{(k)}$  is the optimization result at the previous  $k$ -th gradient descent iteration step and  $\theta_D^{(0)} = \theta_D$ . The generator parameter is also optimized by the Adam (Section 3.4):

$$\theta_G = \theta_G - \eta \cdot \text{Adam} \left( \nabla_{\theta_G} \mathcal{F}(\theta_D^*, \theta_G); \beta_1, \beta_2 \right). \quad (12)$$

where the updating term  $\text{Adam} \left( \nabla_{\theta_G} \mathcal{F}(\theta_D^*, \theta_G); \beta_1, \beta_2 \right)$  is determined by  $\nabla_{\theta_G} \mathcal{F}(\theta_D^*, \theta_G)$ , the gradient of the loss function  $\mathcal{F}$  with respect to  $\theta_G$ . Note that for each updating step of  $\theta_G$  (Equation (12)), there are  $k$  updating steps of  $\theta_D^{(k)}$  (Equation (10)), because  $\theta_D^{(k)}$  depends on  $\theta_G$ .

In step 2), in order to obtain the reconstruction  $\hat{\mathbf{x}}$  of data  $\mathbf{x}$ , it is necessary to first query its latent variable  $\mathbf{z} \in \mathcal{Z}$  and, then, map  $\mathbf{z}$  into the data space  $\mathcal{X}$  by generator,  $\hat{\mathbf{x}} = G(\mathbf{z})$ . For the anomaly detection problem, it is usually assumed that the probability distribution of abnormal data is significantly different from that of normal data [45]: thus, for an optimal generator  $G$ , given an optimal query  $\mathbf{z}_{\text{optimal}}$  of a normal condition pattern  $\mathbf{x}$ , the reconstruction error  $\|\mathbf{x} - G(\mathbf{z}_{\text{optimal}})\|^2$  should be zero, whereas, on the contrary, given an abnormal condition pattern, the reconstruction error is large because the generator always maps its query  $\mathbf{z}$  into a normal data. The literature work [37] regards the search of  $\mathbf{z}_{\text{optimal}}$  as an optimization task,  $\min_{\mathbf{z}} \|\mathbf{x} - G(\mathbf{z}; \theta_G^*)\|^2$ . However, this optimization may suffer from issues such as high computational complexity caused by the need to perform an optimization of  $\mathbf{z}$  for each pattern  $\mathbf{x}$  [46]. Therefore, this work proposes to use an encoder  $E$  as an auxiliary network for searching  $\mathbf{z}_{\text{optimal}}$  w.r.t. data  $\mathbf{x}$  [47]. Unlike the direct optimization of  $\mathbf{z}$ , the auxiliary encoder  $E$  and the generator  $G$  form an auto-encoder to train a querying model. In the auxiliary auto-encoder, only the encoder  $E$  is optimized, wherein the optimization process is achieved by minimizing the reconstruction error:

$$\begin{aligned} \mathcal{L}_{\text{rec}}(\mathbf{x}; \theta_E, \theta_G^*) &= \mathbb{E}_{\mathbf{x} \in \mathcal{X}_{\text{normal}}} \|\mathbf{x} - G(E(\mathbf{x}; \theta_E); \theta_G^*)\|^2, \\ \theta_E^* &= \underset{\theta_E}{\text{argmin}} \mathcal{L}_{\text{rec}}(\mathbf{x}; \theta_E, \theta_G^*), \end{aligned} \quad (13)$$

$$\mathbf{z}_{\text{optimal}} = E(\mathbf{x}; \theta_E^*), \quad (14)$$

where  $\theta_E^*$  is the optimal parameter for encoder  $E$ . The encoder parameter  $\theta_E$  is optimized by Adam (Section 3.4):

$$\theta_E = \theta_E - \eta \cdot \text{Adam}(\nabla_{\theta_E} \mathcal{L}_{\text{rec}}(\mathbf{x}; \theta_E, \theta_G^*); \beta_1, \beta_2). \quad (15)$$

where the updating term  $\text{Adam}(\nabla_{\theta_E} \mathcal{L}_{\text{rec}}(\mathbf{x}; \theta_E, \theta_G^*); \beta_1, \beta_2)$  is determined by  $\nabla_{\theta_E} \mathcal{L}_{\text{rec}}(\mathbf{x}; \theta_E, \theta_G^*)$ , the gradient of the loss function  $\mathcal{L}_{\text{rec}}$  with respect to  $\theta_E$ . The above gradient-based optimization is typically applied using a small learning rate and multiple iteration steps, which leads to a large number of epochs  $N_{\text{epoch}}$  [43].

The anomaly score function of pattern  $\mathbf{x}$  is:

$$\mathcal{A}(\mathbf{x}) = \|\mathbf{x} - \hat{\mathbf{x}}\|^2 = \|\mathbf{x} - G(E(\mathbf{x}; \theta_E^*); \theta_G^*)\|^2 \quad (16)$$

If  $\mathcal{A}(\mathbf{x})$  is larger than a threshold value,  $A_{threshold}$ , set considering normal condition data:

$$A_{threshold} = \max_{\mathbf{x} \in \mathcal{X}_{normal}} \mathcal{A}(\mathbf{x}) \quad (17)$$

then  $\mathbf{x}$  is declared as anomalous, otherwise,  $\mathbf{x}$  is normal. The anomaly detection mechanism is illustrated in Figure 2. If an abnormal data is input into the *AE-GAN* model, the reconstruction of it will locate in the region of normal data distribution, so the reconstruction error will be large and the pattern will be declared as anomalous.

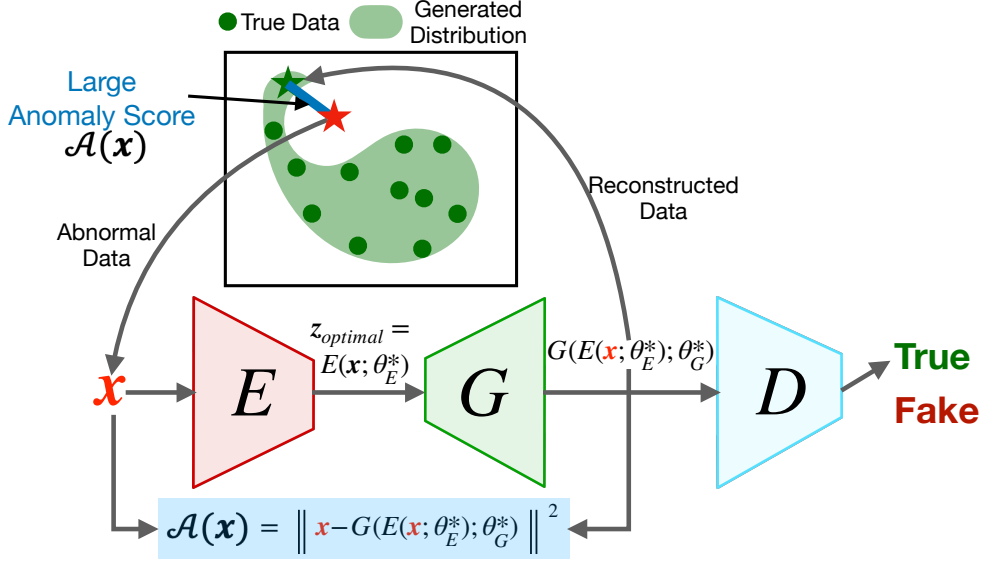


Figure 2: The mechanism of anomaly detection by using *AE-GAN*.

#### 4.2. *AE-GAN* hyper-parameter optimization

The setting of *GAN* hyperparameters has a major impact on the convergence efficiency and stability of *GAN* training [48]. Although  $JSD(p_G \parallel p_{\mathcal{X}_{normal}})$  can be used as an actual objective to optimize the *GAN* architecture, its true value cannot be obtained during *GAN* training [49]. Inspired by [50], this work derives a lower bound of *J-S* divergence between  $p_G$  and  $p_{\mathcal{X}_{normal}}$ , denoted as  $JSD_{LB}(p_G \parallel p_{\mathcal{X}_{normal}})$  which can be computed during training and, therefore, used for the setting of the hyperparameters. Such as, number of hidden neurons, number of hidden layers, size of latent space in generator,  $N_z$ , iteration steps of discriminator per each iteration of generator,  $k$ , and number of epochs,  $N_{epoch}$ .

The optimization objective for the generator satisfies:

$$\mathcal{F}(\theta_D^{(k)}, \theta_G) < \max_{\theta_D} \mathcal{F}(\theta_D, \theta_G) = \mathcal{F}(\theta_D^*(\theta_G), \theta_G) \quad (18)$$

Then, according to Equation (3), we obtain:

$$\begin{aligned} \mathcal{F}(\theta_D^{(k)}, \theta_G) &< -\log 4 + 2 \cdot JSD(p_G \parallel p_{\mathcal{X}_{normal}}) \\ \stackrel{\text{equivalent}}{\iff} \underbrace{\mathcal{F}(\theta_D^{(k)}, \theta_G) / 2 + \log 2}_{JSD_{LB}(p_G \parallel p_{\mathcal{X}_{normal}})} &< JSD(p_G \parallel p_{\mathcal{X}_{normal}}) \end{aligned} \quad (19)$$



where  $\mathcal{F}(\theta_D^{(k)}, \theta_G)/2 + \log 2$  is defined as  $JSD_{LB}(p_G \parallel p_{\mathcal{X}_{normal}})$ . Notice that the encoder module of the *AE-GAN* shares the same network architecture with the discriminator module, and encoder, generator and discriminator are all multiple layers perceptrons with the same number of hidden layers, and each hidden layer has the same number of hidden neurons.

#### 4.3. Ensembled Anomaly Detector by AdaBoost Algorithm

This section details the proposed ensembled anomaly detection method based on *AE-GAN*. The illustration of the methodology is shown in Figure 3. There are two issues in real industrial applications which cause difficulties for anomaly detection with *AE-GAN*: 1) the densities of data distributions are not smooth and 2) the curse of dimensionality.

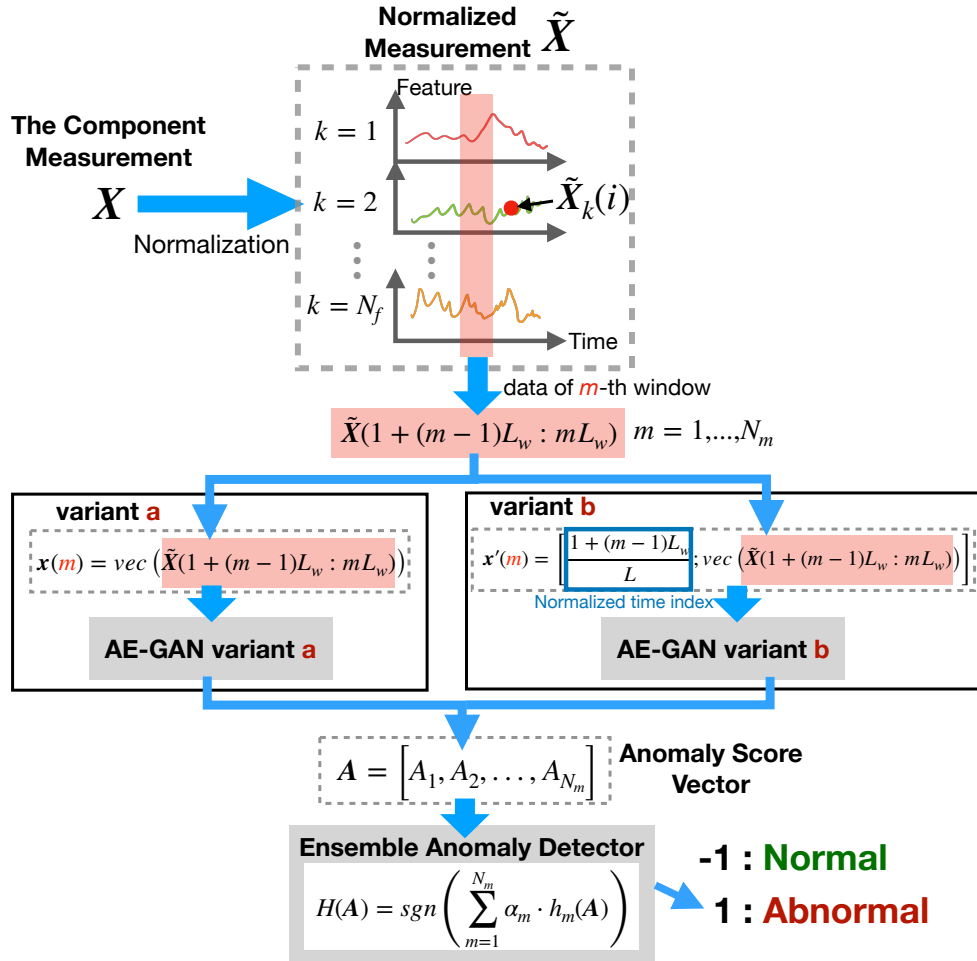


Figure 3: The flowchart of the ensembled anomaly detection method based on *AE-GAN* with variants a) and b).

With respect to 1), non-smooth data distribution occur in many industrial applications [51], including switching of equipment operational conditions, such as equipment turning on and off [52]. In our work, we have found that training *GAN* on distributions whose densities are not smooth prevents  $JSD_{LB}(p_G \parallel p_{\mathcal{X}_{normal}})$  from converging (see Figure 4), which results in the impossibility of obtaining the optimal generator and detecting anomalies using the base *AE-GAN* anomaly detector.

According to [53], if the probability distributions  $p_G$  and  $p_{\mathcal{X}_{normal}}$  are disjoint manifolds, then the optimal discriminator  $D(x; \theta_D^*(\theta_G)) = 1$  for any true data  $x \in \mathcal{X}_{normal}$ , but on the other hand, it is 0 for any gener-

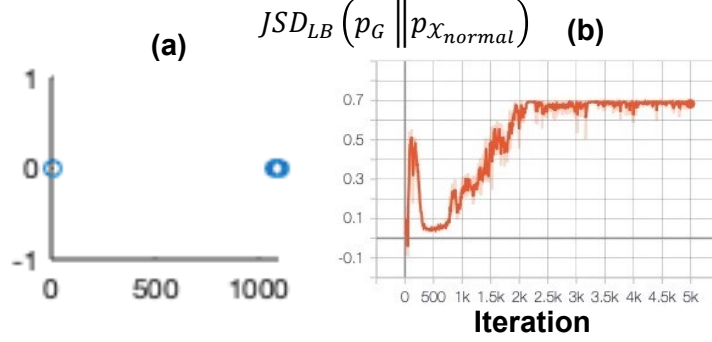


Figure 4: a) example of distribution with non-smooth density in an industrial case; b) the plot of  $JSD_{LB}(p_G \parallel p_{\mathcal{X}_{normal}})$  vs. iteration number for generator.

ated data  $G(\mathbf{z})$ , and thus  $GAN$  loss  $\mathcal{F}(\theta_D^*(\theta_G), \theta_G)$  will be zero. As a result, the value of  $JSD(p_G \parallel p_{\mathcal{X}_{normal}})$  is equal to  $\ln(2) \approx 0.69$ . Figure 4b illustrates a situation in which the distributions  $p_{\mathcal{X}_{normal}}$ , whose density is not smooth, and  $p_G$  are disjoint. According to [53], when a Gaussian noise  $\mathcal{N}(0, \sigma^2)$  is added on the data distribution  $p_{\mathcal{X}_{normal}}$ , this latter will be converted into a distribution with continuous density. As a consequence, the gradient of  $JSD(p_G \parallel p_{\mathcal{X}_{normal}})$  over  $\theta_G$  will not vanish during the training of  $GAN$ , and  $JSD(p_G \parallel p_{\mathcal{X}_{normal}})$  will converge.

Therefore, this work defines a normally distributed random variable  $\epsilon_k(i)$ , which represents an adaptive noise added on the data  $\mathbf{X}_k^r(i)$ , denoting the  $k$ -th feature at the  $i$ -th time stamp from  $r$ -th healthy components. The data with adaptive noise  $\mathbf{X}_k'^r(i)$ , then, becomes:

$$\begin{aligned} \mathbf{X}_k'^r(i) &= \mathbf{X}_k^r(i) + \epsilon_k(i), \\ \epsilon_k(i) &\sim \mathcal{N}\left(0, \sigma_k(i)^2\right), k = 1, \dots, N_f, i = 1, \dots, L \end{aligned} \quad (20)$$

$$\sigma_k(i) = \gamma \cdot \text{std}\left(\{\mathbf{X}_k^r(i)\}_{r=1, \dots, N_{normal}}\right) + \delta \quad (21)$$

where the standard deviation  $\sigma_k(i)$  is a variable that changes according to the standard deviation of  $\{\mathbf{X}_k^r(i)\}_{r=1, \dots, N_{normal}}$ ,  $\gamma \in (0, +\infty)$  is a scaling factor and  $\delta \in (0, +\infty)$  is a bias term to ensure that  $\sigma_k(i) > 0$ , because  $\mathbf{X}_k^r(i)$  can be a constant for  $r = 1, \dots, N_{normal}$ .

With respect to 2), if the data dimensionality  $N_x$  is much larger than the number of the patterns, it is usually considered as high dimensionality which could make computation infeasible or computation complexity increasing exponentially. The typical situation of industrial application is long-term multivariate time series, with a large number of features  $N_f$  and long sequence length  $L$ , which makes it become a high-dimensional problem. As shown in Figure 5, which reports the evolution of  $JSD_{LB}(p_G \parallel p_{\mathcal{X}_{normal}})$  over iteration number of training  $GAN$  in a real industrial application characterized by a large number of features and long sequence length,  $JSD_{LB}(p_G \parallel p_{\mathcal{X}_{normal}})$  tends to the constant  $\ln(2)$  during the  $GAN$  training.

To reduce the dimensionality of long-term multivariate time series, this work proposes to use non-overlapped sliding time windows to split multivariate time series and treat each time window as a separate pattern for anomaly detection; then, AdaBoost algorithm is adopted to aggregate the anomaly detection results for each time window. Particularly, this work has developed two variants: *variant a)* trains individual  $AE$ -GANs for each time window and *variant b)* trains one general  $AE$ -GAN for all time windows.

Before the training of  $AE$ -GANs and testing for anomaly detection, the training normal data and testing data are normalized in the range  $[-1, 1]$ . Let  $\mathbf{X}_k(i)$  be the general notation of the  $k$ -th feature at the  $i$ -th time stamp for the normalized training/test data. In the training phase, data is added with the adaptive noise,

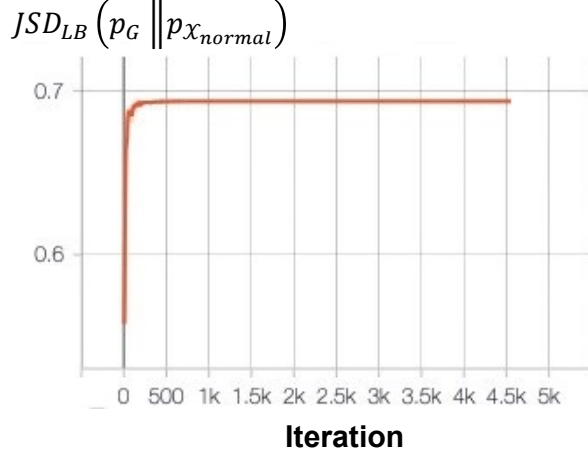


Figure 5: the plot of the  $JSD_{LB}(p_G \parallel p_{x_{normal}})$  of an industrial application example w.r.t iteration number for generator.

whereas in the testing phase, there is no need to add adaptive noise to the data. Thus, the normalization methods for the training and testing phases are as follows:

$$\begin{aligned}
\mathbf{X}_k'^{max}(i) &= \max\{\mathbf{X}_k'^r(i)\}_{r=1,\dots,N_{normal}} \\
\mathbf{X}_k'^{min}(i) &= \min\{\mathbf{X}_k'^r(i)\}_{r=1,\dots,N_{normal}} \\
\widetilde{\mathbf{X}}_k(i) &= 2 \cdot \frac{\mathbf{X}_k(i) + \epsilon_k(i) - \mathbf{X}_k'^{min}(i)}{\mathbf{X}_k'^{max}(i) - \mathbf{X}_k'^{min}(i)} - 1 \quad (\text{for training phase}) \\
\widetilde{\mathbf{X}}_k(i) &= 2 \cdot \frac{\mathbf{X}_k(i) - \mathbf{X}_k'^{min}(i)}{\mathbf{X}_k'^{max}(i) - \mathbf{X}_k'^{min}(i)} - 1 \quad (\text{for testing phase})
\end{aligned} \tag{22}$$

*Variant a).* Let  $\mathbf{x}(m)$  denote generic data in the  $m$ -th time window,  $L_W$  the size of the time window and  $N_m = \text{ceil}(\frac{L}{L_W})$  the number of time windows; this work assigns  $\mathbf{x}(m)$  with the normalized data in the  $m$ -th window:

$$\mathbf{x}(m) = \text{vec}\left(\widetilde{\mathbf{X}}(1 + (m-1)L_W : mL_W)\right), \quad m = 1, \dots, N_m \tag{23}$$

where  $\text{vec}(\cdot)$  denotes the matrix vectorization operation, which stacks the columns of the matrix on top of one another. Note that the time index interval for the  $m$ -th window is  $1 + (m-1)L_W : mL_W$ . This work constructs a data set of all normal condition patterns in window  $m$ :

$$\mathcal{X}_{normal,m} = \{\mathbf{x}^1(m), \dots, \mathbf{x}^r(m), \dots, \mathbf{x}^{N_{normal}}(m)\}, \quad m = 1, \dots, N_m \tag{24}$$

where  $\mathbf{x}^r(m)$  denotes data in the  $m$ -th time window collected from healthy components  $r = 1, \dots, N_{normal}$ . The proposed *AE-GAN* is trained on the data set  $\mathcal{X}_{normal,m}$  for each  $m$ -th time window by applying Equations (9)(10)(12)(13); then, the optimal generator  $G(\mathbf{z}; \theta_{G_m}^*)$  and encoder  $E(\mathbf{x}; \theta_{E_m}^*)$  for the  $m$ -th time window can be obtained. Finally, the anomaly score function of the  $m$ -th time window is obtained:

$$\mathcal{A}_a(\mathbf{x}(m); m) = \|\mathbf{x}(m) - G(E(\mathbf{x}(m); \theta_{E_m}^*); \theta_{G_m}^*)\|^2, \quad m = 1, \dots, N_m \quad (25)$$

The *AE-GAN* anomaly detector with *variant a*) is illustrated in Figure 6, where  $A_m$  and  $A_{m+1}$  denote the anomaly score of the  $m$ -th window and  $m+1$ -th window, respectively.

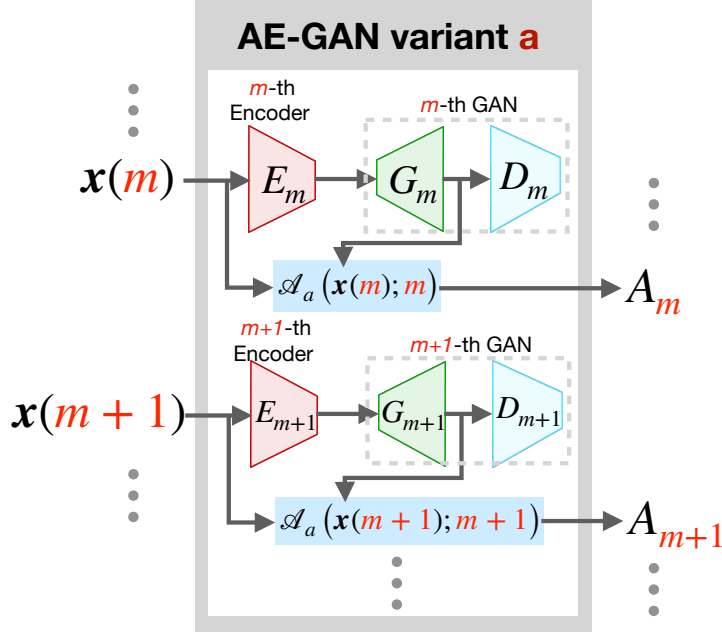


Figure 6: The *AE-GAN* anomaly detector with *variant a*).

*Variant b*). Let  $\mathbf{x}'(m)$  denote the concatenation of the normalized time index in the range  $[0, 1]$  and the generic data of the  $m$ -th time window:

$$\mathbf{x}'(m) = \left[ \frac{1 + (m-1) \cdot L_W}{L}; \text{vec} \left( \widetilde{\mathbf{X}}(1 + (m-1)L_W : mL_W) \right) \right], \quad m = 1, \dots, N_m \quad (26)$$

where symbol ‘;’ represents the vertical concatenation. Note that the dimension of column vector  $\mathbf{x}'(m)$  is  $N_f \cdot L_W + 1$ . This work constructs the data set of normal condition patterns for all time windows:

$$\mathcal{X}'_{normal} = \left\{ \begin{array}{c} \mathbf{x}'^1(1), \dots, \mathbf{x}'^1(m), \dots, \mathbf{x}'^1(N_m), \dots, \\ \mathbf{x}'^r(1), \dots, \mathbf{x}'^r(m), \dots, \mathbf{x}'^r(N_m), \dots, \\ \mathbf{x}'^{N_{normal}}(1), \dots, \mathbf{x}'^{N_{normal}}(m), \dots, \mathbf{x}'^{N_{normal}}(N_m) \end{array} \right\} \quad (27)$$

where  $\mathbf{x}'^r(m)$  denotes the data from healthy components  $r = 1, \dots, N_{normal}$ . The proposed *AE-GAN* is trained on the data distribution  $\mathcal{X}'_{normal}$  by applying Equations (9)(10)(12)(13); note that the size of set  $\mathcal{X}'_{normal}$  is  $N_{normal} \times N_m$ . Then, the universal optimal generator  $G(\mathbf{z}; \theta_G^*)$  and encoder  $E(\mathbf{x}; \theta_E^*)$  for all time windows can be obtained. Finally, the universal anomaly score function  $\mathcal{A}_b(\mathbf{x}'(m))$  is obtained:

$$\mathcal{A}_b(\mathbf{x}'(m)) = \|\mathbf{x}'(m) - G(E(\mathbf{x}'(m); \theta_E^*); \theta_G^*)\|^2, \quad m = 1, \dots, N_m \quad (28)$$

The *AE-GAN* anomaly detector with *variant b*) is illustrated in Figure 7, where  $A_m$  denotes the anomaly score of the  $m$ -th window.

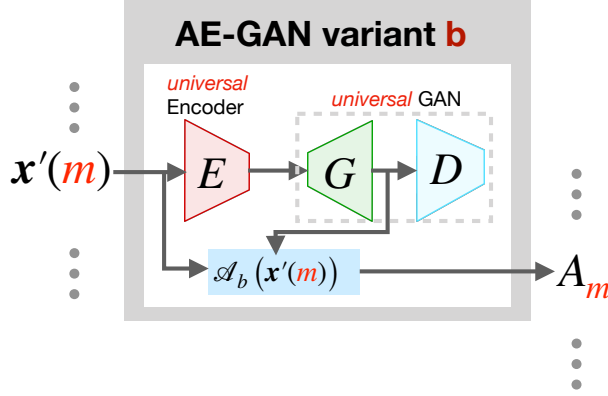


Figure 7: The *AE-GAN* anomaly detector with *variant b*).

The details of obtaining anomaly scores using *AE-GAN* with *variant a*) are contained in *Algorithm 2* and using *AE-GAN* with *variant b*) in *Algorithm 3*. Furthermore, to learn an ensembled anomaly detector based on anomaly scores for time windows, this work adapts the AdaBoost algorithm to the one-class classification. Let function  $h : \mathbf{A} \rightarrow \{-1, 1\}$  denote the base anomaly detector, where  $\mathbf{A} = [A_1, \dots, A_m, \dots, A_{N_m}]^T$  and  $A_m$  is the generic anomaly score at the  $m$ -th time window by using *AE-GAN* with *variant a*) or *b*), and  $-1$  represents normal and  $1$  represents abnormal. Before applying AdaBoost ensemble learning, a validation normal condition dataset  $\{\mathbf{X}^v\}_{v=1, \dots, N_v}$  is required and the anomaly score  $\mathbf{A}^v = [A_1^v, \dots, A_m^v, \dots, A_{N_m}^v]^T$ ,  $v = 1, \dots, N_v$  needs to be calculated for obtaining the anomaly score threshold [16]. The adapted AdaBoost algorithm is showed in *Algorithm 4*, where  $\mathbf{o}^{(m)} = [0, \dots, 1(m-th), \dots, 0]^T$  is a one-hot vector of dimension  $N_m$ , in which the  $m$ -th element is  $1$  whereas all others are  $0$ ,  $\text{Percentile}_c\{\cdot\}$  represents the number in the set that has a probability  $c$  larger than the other numbers: if  $c = 1$ ,  $\text{Percentile}_c\{\cdot\}$  is the maximum in the set, else if  $c = 0$ , it is the minimum in the set. In general, tuning percentile  $c$  can trade off between missed alarms and false alarms: a higher value of  $c$  decreases missed alarms, whereas a lower value of  $c$  decreases false alarms.  $\lambda \sim \mathcal{N}(0, 10^{-10})$  is a small noise term to keep stability in the AdaBoost Ensemble learning, in particular, avoiding  $h_m(\mathbf{A}) = 0$ .

## 5. Case Study

### 5.1. Protocol and Setting

*Performance Metrics.* For simplicity, the normal condition pattern is noted as positive and the abnormal condition pattern is noted as negative: true positive ( $tp$ ) represents correctly classified normal condition patterns, true negative ( $tn$ ) represents correctly classified abnormal condition patterns, false positive ( $fp$ ) represents abnormal condition patterns but misclassified as normal, false negative ( $fn$ ) represents normal condition patterns but misclassified as abnormal. The following performance metrics are used to evaluate the anomaly detection results:

- *Accuracy*, the fraction of correctly classified normal or abnormal condition patterns among all patterns, which is defined by the ratio of the sum of  $tp$  and  $tn$  to the total number of tested patterns ( $tp + tn +$

---

**Algorithm 2:** Training *AE-GAN* for Computing Anomaly Score: *variant a*)

---

**Input:** Normal condition data  $\{\mathbf{X}^r\}_{r=1,\dots,N_{normal}}$ .  
**Output:** Anomaly score  $\mathbf{A}^r = [A_1^r, \dots, A_m^r, \dots, A_{N_m}^r]^T$  for  $r=1, \dots, N_{normal}$  and anomaly score function  $\mathcal{A}_a(\mathbf{x}(m); m)$  for  $m=1, \dots, N_m$ .  
**Initialize:** Scaling factor  $\gamma$  and bias term  $\delta$  for adaptive noise, time window size  $L_W$ .

- 1 Add adaptive noise and obtain  $\mathbf{X}'^r$  by using Equations (20) (21) for  $r = 1, \dots, N_{normal}$
- 2 Normalize  $\mathbf{X}'^r$  into  $\widetilde{\mathbf{X}}^r$  in the range  $[-1, 1]$  by using Equation (22) for  $r = 1, \dots, N_{normal}$   
 /\* Train *AE-GAN* model for each  $m$ -th time window \*/
- 3 **for**  $m = 1, \dots, N_m$  **do**
- 4     Obtain  $\mathbf{x}^r(m)$  by using Equation (23) for  $r = 1, \dots, N_{normal}$ .
- 5     Construct set  $\mathcal{X}_{normal,m}$  by using Equation (24) and assign to  $\mathcal{X}_{normal}$ .
- 6     Initialize  $\theta_D, \theta_G, \theta_E$  by Xavier Uniform initialization method [54].
- 7     **for**  $epoch = 1, \dots, N_{epoch}$  **do**
- 8         **for**  $k = 1, \dots, K$  **do**
- 9             Update  $\theta_D^{(k)}$  by Equation (10).
- 10          Update  $\theta_G$  by Equation (12).
- 11     **for**  $epoch = 1, \dots, N_{epoch}$  **do**
- 12         Update  $\theta_E$  by Equation (15).
- 13     Assign  $\theta_{G_m}^* \leftarrow$  optimized  $\theta_G$ ,  $\theta_{E_m}^* \leftarrow$  optimized  $\theta_E$  and obtain  $\mathcal{A}_a(\mathbf{x}(m); m)$  in Equation (25).  
 /\* Compute Anomaly Scores \*/
- 14 **for**  $r = 1, \dots, N_{normal}$  **do**
- 15     **for**  $m = 1, \dots, N_m$  **do**
- 16         Compute Anomaly Score  $A_m^r = \mathcal{A}_a(\mathbf{x}^r(m); m)$  by Equation (25).

---

---

**Algorithm 3:** Training *AE-GAN* for Computing Anomaly Score: *variant b*)

---

**Input:** Normal condition data  $\{\mathbf{X}^r\}_{r=1,\dots,N_{normal}}$ .  
**Output:** Anomaly score  $\mathbf{A}^r = [A_1^r, \dots, A_m^r, \dots, A_{N_m}^r]^T$  for  $r=1, \dots, N_{normal}$  and anomaly score function  $\mathcal{A}_b(\mathbf{x}'(m); m)$  for  $m=1, \dots, N_m$ .  
**Initialize:** Scaling factor  $\gamma$  and bias term  $\delta$  for adaptive noise, time window size  $L_W$ .

- 1 Add adaptive noise and obtain  $\mathbf{X}'^r$  by using Equations (20) (21) for  $r = 1, \dots, N_{normal}$
- 2 Normalize  $\mathbf{X}'^r$  into  $\widetilde{\mathbf{X}}^r$  in the range  $[-1, 1]$  by using Equation (22) for  $r = 1, \dots, N_{normal}$
- 3 Obtain  $\mathbf{x}'^r(m)$  by using Equation (26) for  $m = 1, \dots, N_m$ ,  $r = 1, \dots, N_{normal}$ .
- 4 Construct set  $\mathcal{X}'_{normal}$  by using Equation (27) and assign to  $\mathcal{X}_{normal}$ .
- 5 Initialize  $\theta_D, \theta_G, \theta_E$  by Xavier Uniform initialization method [54].
- /\* Train *AE-GAN* model \*/
- 6 **for**  $epoch = 1, \dots, N_{epoch}$  **do**
- 7     **for**  $k = 1, \dots, K$  **do**
- 8         Update  $\theta_D^{(k)}$  by Equation (10).
- 9         Update  $\theta_G$  by Equation (12).
- 10 **for**  $epoch = 1, \dots, N_{epoch}$  **do**
- 11     Update  $\theta_E$  by Equation (15).
- 12 Assign  $\theta_G^* \leftarrow$  optimized  $\theta_G$ ,  $\theta_E^* \leftarrow$  optimized  $\theta_E$  and obtain  $\mathcal{A}_b(\mathbf{x}'(m))$  in Equation (28).
- /\* Compute Anomaly Scores \*/
- 13 **for**  $r = 1, \dots, N_{normal}$  **do**
- 14     **for**  $m = 1, \dots, N_m$  **do**
- 15         Compute Anomaly Score  $A_m^r = \mathcal{A}_b(\mathbf{x}'^r(m))$  by Equation (28).

---



---

**Algorithm 4:** AdaBoost Ensemble Learning for Anomaly Detection

---

**Input:** Anomaly score validation set,  $\mathcal{V} = \{\mathbf{A}^v\}_{v=1,\dots,N_v}$ , weak classifier  $h : \mathbf{A} \rightarrow \{-1, 1\}$ , percentile number  $c$ .  
**Output:** Ensembled classifier  $H(\mathbf{A}) = \text{sgn}\left(\sum_{m=1}^{N_m} \alpha_m \cdot h_m(\mathbf{A})\right)$   
**Initialize:** Weights of validation set  $\mathcal{V}$  anomaly scores  $w_1^{(1)}, w_2^{(1)}, \dots, w_{N_v}^{(1)}$  set to  $\frac{1}{N_v}$ , initial error rate  $\epsilon_m, m=1, \dots, N_m$  set as  $\frac{1}{2}$ .

/\* Train AdaBoost Ensemble model \*/

- 1 **for**  $m = 1, \dots, N_m$  **do**
- 2      $A_{threshold,m} = \text{Percentile}_c\left\{(\mathbf{A}^v)^T \cdot \mathbf{o}^{(m)}\right\}_{v=1,\dots,N_v}$ .
- 3     Obtain classifier  $h_m(\mathbf{A}) = \text{sgn}(\mathbf{A}^T \cdot \mathbf{o}^{(m)} - A_{threshold,m} + \lambda)$ , with  $\lambda$  sampled from  $\mathcal{N}(0, 10^{-10})$ .
- 4     Obtain error rate  $\epsilon_m = \sum_{v, 1=h_m(\mathbf{A}^v)} w_v^{(m)}$ ,  $v = 1, \dots, N_v$ .
- 5     Obtain weights of classifier  $h_m$ ,  $\alpha_m = \frac{1}{2} \ln\left(\frac{1-\epsilon_m}{\epsilon_m}\right)$ .
- 6     Update weights  $w_v^{(m+1)} = w_v^{(m)} e^{\alpha_m \cdot h_m(\mathbf{A}^v)}$ ,  $v = 1, \dots, N_v$ .
- 7     Normalize weights  $w_v^{(m+1)} = \frac{w_v^{(m+1)}}{\sum_{v=1}^{N_v} w_v^{(m+1)}}$ ,  $v = 1, \dots, N_v$ .

---

$fp + fn$ ).

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn}$$

- *Precision*, the fraction of correctly classified normal condition patterns among patterns that are classified as normal, which is defined by the ratio of  $tp$  to the sum of  $tp$  and  $fp$ .

$$Precision = \frac{tp}{tp + fp}$$

- *Recall*, the fraction of correctly classified normal condition patterns among true normal condition patterns, which is defined by the ratio of  $tp$  to the sum of  $tp$  and  $fn$ .

$$Recall = \frac{tp}{tp + fn}$$

- *F-score*, a score that is the harmonic mean of *Precision* and *Recall*; this work uses the balanced *F-score*.

$$F - score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$$

- *Receiver Operating Characteristic (ROC) curve*, the *ROC* curve is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one, and it is created by plotting the true normal rate (*TPR*) against the false positive rate (*FPR*) at various threshold settings.

$$TPR = Recall, FPR = \frac{fp}{fp + tn}$$

- *Area Under the ROC Curve (AUC)*, calculated by using an average of a number of trapezoidal approximations [55].

The range of the performance metrics *Accuracy*, *Precision*, *Recall*, *F-score* and *AUC* for anomaly detection is  $[0, 1]$ , and larger value means better performance.

*Methods considered for the results comparison.* This work compares the base anomaly detector with *AE-GAN* and an AdaBoost ensembled *AE-GAN* with other state-of-the-art anomaly detection methods, such as *OC-SVM*, *AAKR*, *GMM*, *AE* and *AnoGAN*. Firstly, base anomaly detector with *AE-GAN* is compared with *OC-SVM*, *AAKR*, *GMM*, *AE* and *AnoGAN* on the synthetic dataset for verifying the effectiveness. Secondly, AdaBoost ensembled *AE-GAN* with variants *a*) and *b*) is compared with *OC-SVM*, *AAKR*, *GMM* and *AE*, and also, compared with four ensembled approaches based on *OC-SVM*, *AAKR*, *GMM* and *AE* respectively, in which each ensembled approach uses the adapted AdaBoost algorithm (*Algorithm 4*) to learn an ensembled anomaly detector for each time window.

*OC-SVM* for anomaly detection [56] is formulated to estimate the support of a high-dimensional distribution, based on which it can find the margin of normal data distribution and conduct anomaly detection tasks. *OC-SVM* generates a score function,  $f(\mathbf{x}; \theta_{OC-SVM}) = 0.5 \sum_j \hat{\alpha}_j G(\mathbf{x}, \mathbf{x}_j)$  to evaluate patterns, where  $\theta_{OC-SVM} = \{\hat{\alpha}_j\}_j$  denotes the *OC-SVM* parameters set,  $G$  denotes the gram matrix [20]. According to the theory of *OC-SVM* [57], a smaller score indicates that the pattern is more likely abnormal. This work defines  $\mathcal{A}_{OC-SVM}(\mathbf{x}) = -f(\mathbf{x}; \theta_{OC-SVM})$  as the anomaly score function of *OC-SVM*.

*AAKR* for anomaly detection [27] is a reconstruction model, in which the reconstruction  $\hat{\mathbf{x}}_{AAKR}$  is the weighted sum of normal condition patterns and the weight is measured by a radial basis similarity function between test pattern and each normal condition pattern. According to the basic assumption of reconstruction-based anomaly detection [16], abnormal condition patterns have larger reconstruction error than normal condition patterns. This work defines  $\mathcal{A}_{AAKR}(\mathbf{x}) = \|\mathbf{x} - \hat{\mathbf{x}}_{AAKR}\|^2$  as the anomaly score function of *AAKR*.



*GMM* for anomaly detection [58] models the normal condition patterns distribution and the likelihood function can be used to distinguish the abnormal condition patterns, because a small likelihood indicates that the pattern sampled from *GMM* has a smaller probability, which means this pattern is more likely abnormal. Let  $p(\mathbf{x}; \theta_{GMM}) = \sum_{i=1}^k \phi_i \mathcal{N}(\mathbf{x}; \mu_i, \Sigma_i)$  denote the likelihood function, where  $\theta_{GMM} = \{\phi_i, \mu_i, \Sigma_i\}_{i=1, \dots, k}$  denotes the *GMM* parameters set,  $\phi_i$  the component weight,  $\mu_i$  the mean and  $\Sigma_i$  the covariance. This work defines  $\mathcal{A}_{GMM}(\mathbf{x}) = -p(\mathbf{x}; \theta_{GMM})$  as the anomaly score function of *GMM*. The number of components  $k$  is set to 1 for all case studies.

*AE* for anomaly detection [59] has been illustrated in Section 3.2 and it is trained on normal data and the reconstruction error  $\|\mathbf{x} - \hat{\mathbf{x}}\|^2$  is used for detecting anomalies according to the basic assumption of reconstruction-based anomaly detection [45, 16]. This work defines  $\mathcal{A}_{AE}(\mathbf{x}) = \|\mathbf{x} - \hat{\mathbf{x}}\|^2$  as the anomaly score function of *AE*. It should be noted that the encoder in *AE* and *AE-GAN* have the same model architecture, and also the generator in *AE* and the generator in *AE-GAN* have the same model architecture. In addition, the Adam optimizer is used to optimize the *AE* loss function, where the two coefficients contained  $\beta_1, \beta_2$  are used for computing running averages of the gradient and its square. The batch size is noted as  $L_{batch}$ . The parameter settings used in *AE*, for example, Adam coefficient  $\beta_1, \beta_2$ , batch size  $L_{batch}$ , learning rate  $\eta$ , number of epoch  $N_{epoch}$ , are the same as in *AE-GAN*.

*AnoGAN* for anomaly detection [37], obtains optimal latent variable  $\mathbf{z}_{optimal}$  of a pattern  $\mathbf{x}$  by minimizing the reconstruction error w.r.t.  $\mathbf{z}$ ; then,  $\|\mathbf{x} - G(\mathbf{z}_{optimal})\|^2$  is used for distinguishing the abnormal condition patterns which have larger reconstruction error than the normal condition patterns, according to the assumption in Section 4.1. This work defines  $\mathcal{A}_{AnoGAN}(\mathbf{x}) = \|\mathbf{x} - G(\mathbf{z}_{optimal})\|^2$  as the anomaly score function of *AnoGAN*. It should be noted that *AnoGAN* shares the same trained *GAN* module with the proposed *AE-GAN*; the parameter settings used in optimizing latent variable  $\mathbf{z}_{optimal}$ , for example, Adam coefficient  $\beta_1, \beta_2$ , learning rate  $\eta$ , number of epochs  $N_{epoch}$ , are the same as those for *AE-GAN* training.

## 5.2. Synthetic Case

*Synthetic Datasets.* This section introduces three synthetic datasets to verify the anomaly detection performance of the proposed base anomaly detector with *AE-GAN* (Section 4.1). Normal condition pattern with different shapes are examined: *Cone*, *Two Gaussian Ball* and *Bowl Manifold* (Figure 8). For all three synthetic datasets, the abnormal condition patterns are uniformly distributed in the space outside the normal condition pattern distribution but inside a 3-D cube with a range of  $[-10, 10]$  for each dimension. For all three datasets, we use normal condition patterns in training set of size 3000, test normal patterns of size 643 and test abnormal condition patterns of size 642.

- *Cone.* The normal condition patterns are obtained by using a cone with bottom radius 2 and height 3 to truncate patterns of a 3-D Gaussian distribution with mean  $[4, 0, 0]$  and variance  $diag(1, 1, 1)$ , in which all patterns inside the Cone are normal.
- *Two Gaussian Ball.* The normal condition patterns are obtained by using two 3-D spheres, whose centers are located at  $[\pm 4, 0, 0]$  and the radius is 2, to truncate patterns in two 3-D Gaussian distributions with mean  $[\pm 4, 0, 0]$  variance  $diag(1, 1, 1)$ , in which all patterns inside these two spheres are normal.
- *Bowl Manifold.* The normal condition patterns are obtained by generating a hemisphere with radius 6 and center point  $[0, 0, 0]$  and randomly sampling points on this hemisphere. The abnormal patterns are not located on this Bowl Manifold surface.

*Implementation details.* The *AE-GAN* contains three sub-networks, namely generator, discriminator and encoder, and each sub-network is implemented by a Multiple Layer Perceptron (*MLP*) neural network with two hidden layers. The *GAN* module is composed by discriminator and generator. The iteration steps of discriminator for each iteration step of generator are set to  $k = 5$ , according to [53, 50]. The *AE-GAN* model architecture is listed in Table 1, where the Latent Space Layer acts as both the output layer of the encoder and the input layer of the generator, and the number of neurons in the Latent Space Layer is

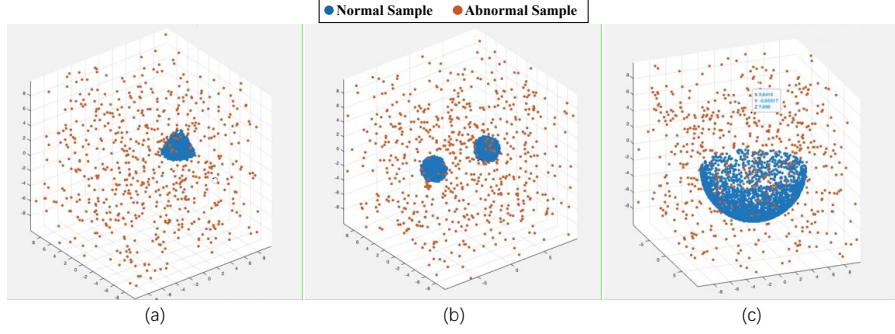


Figure 8: The three synthetic datasets mimic the complicated and challenging datasets in real industrial applications. Blue points are normal condition patterns and red points are abnormal condition patterns. (a) is Cone dataset, (b) is Two Gaussian Ball dataset and (c) is Bowl Manifold dataset.

determined by the specific cases: the number of neurons is set to 2 for Bowl Manifold, and 3 for Cone and Two Gaussian. Because the generator can produce the manifold distribution if the input dimension of the generator is smaller than the output dimension, in particular, when the dimension is 2, the generator can reproduce the Bowl Manifold distribution best (Figure 8c). However, for ‘volumes’ distribution in 3-D space, e.g. Cone and Two Gaussian distribution, the generator can reproduce the distribution best if the input dimension is 3. The Adam optimizer is used to optimize *GAN* (Generator and Discriminator modules) and *AE* (Encoder and Generator), where the learning rate  $\eta = 0.0002$ , the coefficients  $\beta_1 = 0.9, \beta_2 = 0.999$ , the batch size  $L_{batch} = 100$  and the number of epochs  $N_{epoch} = 1000$ . Referring to Equation (17), thresholds in comparing anomaly detection methods, e.g. *OC-SVM*, *GMM*, *AAKR*, *AE*, *AnoGAN*, are set as the maximum of anomaly scores among normal condition patterns in training set. Each dimension of the data is set into  $[-1, 1]$ .

Table 1: *AE-GAN* model architecture. Both Encoder and Generator activate their hidden layers by Rectified Linear Unit (*ReLU*) [60], whereas Discriminator uses Leaky *ReLU* [61] as activation function and the leaky rate is set as 0.2.

Module	Layer	# of Neurons	Activation Function
Encoder	Input Layer	3	
	Hidden Layer #1	50	ReLU
	Hidden Layer #2	50	ReLU
	Latent Space Layer	2 or 3	
Generator	Hidden Layer #1	50	ReLU
	Hidden Layer #2	50	ReLU
	Output Layer	3	Tanh
Discriminator	Input Layer	3	
	Hidden Layer #1	50	Leaky ReLU(0.2)
	Hidden Layer #2	50	Leaky ReLU(0.2)
	Output Layer	1	Sigmoid

In Figure 9, we can see that the *GAN* nearly perfectly reconstructs the distribution of normal condition patterns for very complex distributions such as Cone, Two Gaussian Ball and Bowl Manifold.

According to the anomaly detection results showed in Figure 10 and Table 2, *AE-GAN* has the highest accuracy and F-score in all three synthetic datasets. Although *AAKR* has highest precision score 1, which means zero missed alarms, for all the synthetic datasets, *AAKR* also has the highest false alarm rate. In particular, for Cone dataset, the proposed *AE-GAN* achieves zero false alarms and zero missed alarms, which outperforms all compared methods. Note that although recall score of *OC-SVM*, *GMM* and *AnoGAN* are

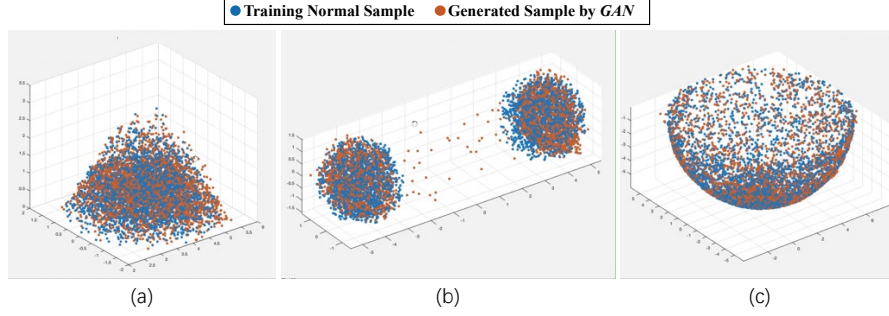


Figure 9: Normal patterns in training set and generated patterns by *GAN* w.r.t. three complex distributions. *a*) is Cone distribution, *b*) is Two Gaussian Ball distribution and *c*) is Bowl Manifold distribution.

		N: Normal Sample		A: Abnormal Sample											
		<i>OC-SVM</i>		<i>AAKR</i>		<i>GMM</i>		<i>AE</i>		<i>AnoGAN</i>		<i>AE-GAN (proposed)</i>			
Cone	N	643	8	587	0	643	10	625	6	643	6	643	0	643	0
	A	0	634	56	642	0	632	18	636	0	636	0	642	0	642
	Predicted/True	N	A	N	A	N	A	N	A	N	A	N	A	N	A
Two Gaussian Ball	N	643	4	473	0	643	6	635	4	603	16	642	3	642	3
	A	0	638	170	642	0	636	8	638	40	626	1	639	1	639
	Predicted/True	N	A	N	A	N	A	N	A	N	A	N	A	N	A
Bowl Manifold	N	639	64	545	0	638	122	627	32	639	230	631	23	631	23
	A	4	578	98	642	5	520	16	610	4	412	12	619	12	619
	Predicted/True	N	A	N	A	N	A	N	A	N	A	N	A	N	A

Figure 10: Confusion matrix of anomaly detection result w.r.t. Cone, Two Gaussian Ball and Bowl Manifold datasets.

Table 2: Anomaly detection performance w.r.t. Cone, Two Gaussian Ball and Bowl Manifold datasets.

Dataset	Metric	<i>OC-SVM</i>	<i>AAKR</i>	<i>GMM</i>	<i>AE</i>	<i>AnoGAN</i>	<i>AE-GAN (proposed)</i>
Cone	Accuracy	0.9938	0.9564	0.9922	0.9813	0.9977	<b>1</b>
	Precision	0.9877	<b>1</b>	0.9847	0.9905	0.9954	<b>1</b>
	Recall	<b>1</b>	0.9129	<b>1</b>	0.9720	<b>1</b>	<b>1</b>
	F-score	0.9938	0.9545	0.9923	0.9812	0.9977	<b>1</b>
Two Gaussian Ball	Accuracy	<b>0.9969</b>	0.8677	0.9953	0.9907	0.9564	<b>0.9969</b>
	Precision	0.9938	<b>1</b>	0.9908	0.9937	0.9742	0.9953
	Recall	<b>1</b>	0.7356	<b>1</b>	0.9876	0.9378	0.9984
	F-score	0.9968	0.8477	0.9954	0.9906	0.9556	<b>0.9969</b>
Bowl Manifold	Accuracy	0.9471	0.9237	0.9012	0.9626	0.8179	<b>0.9728</b>
	Precision	0.909	<b>1</b>	0.8395	0.9514	0.7353	0.9648
	Recall	<b>0.9938</b>	0.8476	0.9922	0.9751	<b>0.9938</b>	0.9813
	F-score	0.9495	0.9175	0.9095	0.9631	0.8452	<b>0.9730</b>

1, they have relatively high missed alarm rates and this would be unacceptable for risk-critical industrial applications. For Two Gaussian Ball dataset, although *OC-SVM* and *GMM* have the highest recall score 1, they have more missed alarm than *AE-GAN*. *AE-GAN* also achieves the competitive top-2 highest precision and recall scores. For Bowl Manifold dataset, *OC-SVM* and *AnoGAN* have the highest recall score but they cause many more missed alarms than *AE-GAN*. *AE-GAN* achieves competitive scores on precision (top 2)

and recall (top 3).

In summary, the proposed base anomaly detector with *AE-GAN* has the best anomaly detection performance on the Cone dataset. Also, on the Two Gaussian Ball and Bowl Manifold Datasets, although *AE-GAN* cannot achieve the lowest false alarm and missed alarm, it still has the highest accuracy and F-score, simultaneously, the nearly top-2 precision and recall score, and this means that *AE-GAN* has the best trade-off between false alarms and missed alarms, which is a critical capability for risk-critical applications. Overall, the proposed *AE-GAN* has the best comprehensive anomaly detection performance.

### 5.3. Anomaly Detection for Automatic Door in High-Speed Train

*Real Industrial Dataset.* The real industrial dataset is collected from the automatic door components of high-speed trains. There is a current sensor (recording tractive force) and a decoder sensor (recording position) to record the state during the door opening and closing processes. Due to the different time of duration to operate the door, the sensor records for a fixed time of duration, 855 time units, to ensure that the entire operation process is covered. This real industrial dataset contains 138 components operated on normal condition, and 22 components on fault type A and 33 components on fault type B. The statistics of the dataset are shown in Table 3. This work uses the signals during both the door opening and closing processes to detect whether the component is normal or abnormal; so, the start time of door opening and closing needs to be synchronized to derive a multivariate time series. Figure 11 shows the example signals of normal components, where *a*) is feature #1: open door, current signal, *b*) is feature #2: open door, decoder signal, *c*) is feature #3: close door, current signal and *d*) is feature #4: close door, decoder signal.

Table 3: The automatic door dataset.

Type	Number
Training normal components	100
Validation normal components	20
Test normal components	18
Test components with fault A	22
Test components with fault B	33

This section investigates the effect of the size of the non-overlapped sliding time windows and the effect of adaptive noise.

- *The effect of window size.* This paragraph experiments the effect of different window sizes on the convergence of  $JSD_{LB}(p_G \parallel p_{\mathcal{X}_{normal}})$ . In the experiment, the window size  $L_W$  is set to 1, 3, 5, 50 time steps, and the normal components signals at the time window with a starting time of 400 is used to train the *GAN* (see details in Equations (23)(24)); the size of latent space in *GAN* is set to  $4 \times L_W$ . During the *GAN* training process, *J-S* divergence at each iteration of the generator optimization is recorded (see Figure 12). It is found that when the window size gradually shrinks to 1, the  $JSD_{LB}(p_G \parallel p_{\mathcal{X}_{normal}})$  gradually converges approximately to 0, which further proves that the generator cannot reproduce the high-dimensional distribution when a small set of data is given.
- *The effect of adaptive noise.* This paragraph experiments the effect of adaptive noise on the convergence of  $JSD_{LB}(p_G \parallel p_{\mathcal{X}_{normal}})$ . In the experiment, the normal components signal at time 460 is used to train the *GAN* (see Figure 13b) and the parameters of the adaptive noise that is added to the data are set as  $\gamma = 0.02$ ,  $\delta = 0.001$  (see Equations (20)(21)), and the size of latent space in *GAN* is set to 4 dimensions. During the *GAN* training process, the  $JSD_{LB}(p_G \parallel p_{\mathcal{X}_{normal}})$  at each iteration of the generator optimization is recorded (see Figure 13a). This experiment verifies that after adding adaptive noise to the data distribution with non-smooth density, the original distribution is converted to a smooth distribution, so that  $JSD_{LB}(p_G \parallel p_{\mathcal{X}_{normal}})$  converges to 0 according to Section 4.3.

Notice that the *AE-GAN* activation function is *ReLU* and the batch size  $L_{batch}$  is set to 20 in all cases.

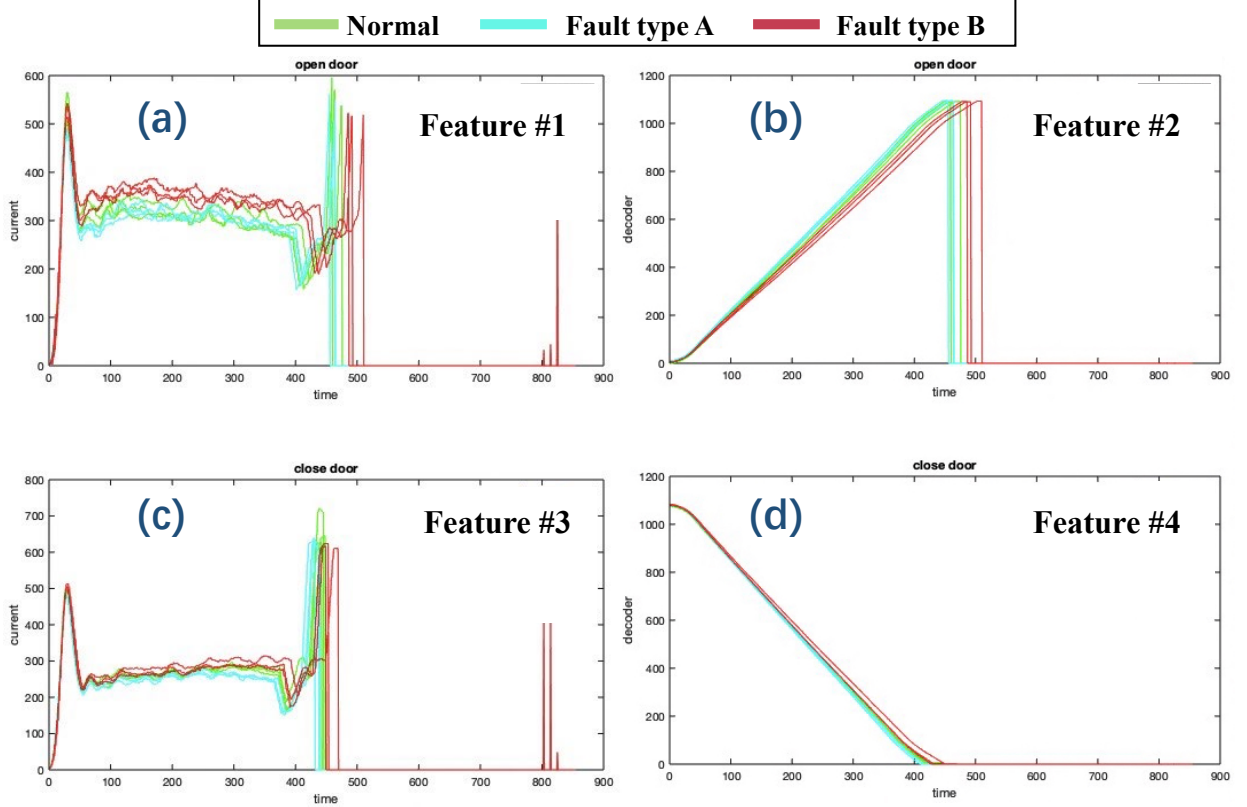


Figure 11: Example signals of normal components with fault type A and fault type B in a real industrial dataset.

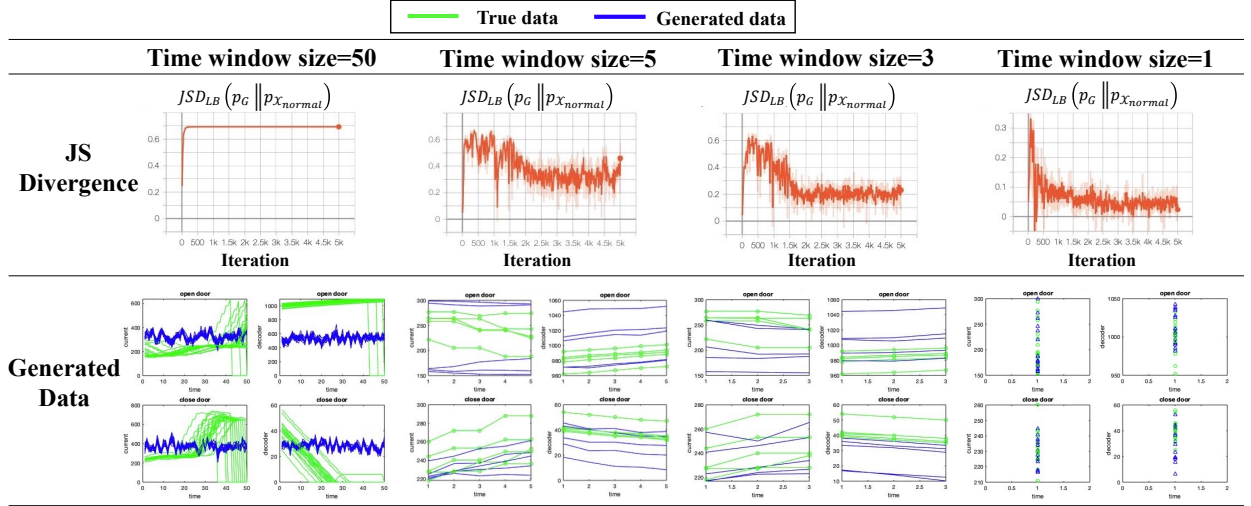


Figure 12: The effect of time window size on the convergence of  $J$ - $S$  divergence and the generated data.

Figure 14 shows the optimization results of the  $AE$ - $GAN$  hyper-parameters (the default initial  $AE$ - $GAN$  hyper-parameters are indicated by the solid blackline):  $N_{epoch} = 1000$ , iteration steps of discriminator for each iteration step of generator,  $k = 5$ , latent space size  $N_z = 4$ , number of hidden layers= 2 and number of

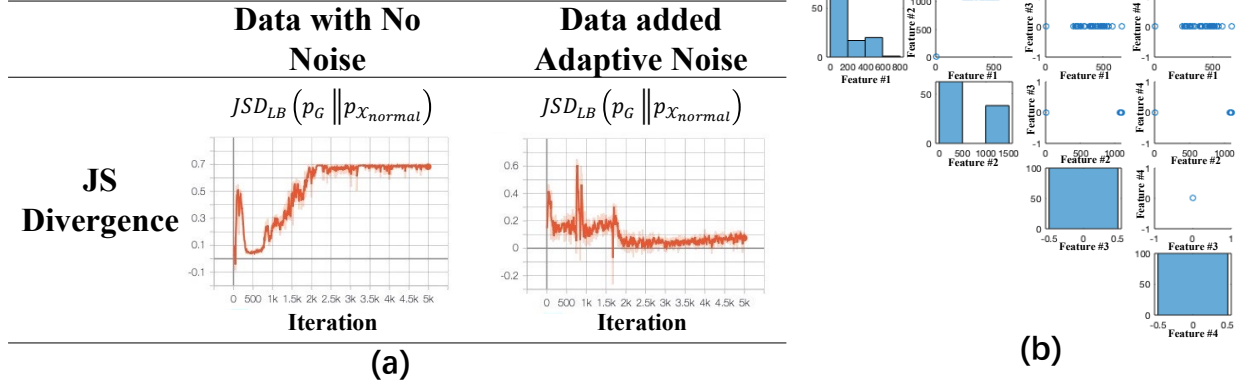


Figure 13: *a)* The effect of adaptive noise on the convergence of  $J$ - $S$  divergence, *b)* the normal component data whose distribution has non-smooth density at time 460.

hidden neurons= 200. The normal components signal at time 400 have been used to train  $AE$ - $GAN$ . Due to the limited computing power, the successive greedy search is used to do the optimization and the order of search is epochs number, iteration steps of discriminator for each iteration step of generator, latent space size, number of hidden layers and hidden neurons number. The optimization objective is  $JSD_{LB}(p_G \parallel p_{\mathcal{X}_{normal}})$  (Section 4.2), After training of  $AE$ - $GAN$  with the optimal hyper-parameters, an example of the generator distribution is shown in Figure 15. We see that the true data distribution can be nearly perfectly reproduced, which satisfies the basic prerequisite of  $GAN$ -based anomaly detection methods, as explained in Section 4.1.

This work compares the proposed  $AE$ - $GAN$  with  $OC$ - $SVM$ ,  $AAKR$ ,  $GMM$  and  $AE$ .  $AnoGAN$  is not compared because it is very computationally intensive when finding optimal latent variable  $\mathbf{z}_{optimal}$  w.r.t. each training and test samples. As for the compared methods, this work uses two strategies. The first is to treat the multivariate time series as the input data sample and obtain the anomaly score (Section 5.1); then, the threshold is set to the maximum value of the anomaly scores among the training normal samples (Equation (17)). The second strategy is similar to the proposed ensembled anomaly detector with  $AE$ - $GAN$ , which uses non-overlapped sliding time windows (size set to 1) to split multivariate time series and treat each time window as a separate data sample for anomaly detection and obtain the anomaly score (Section 5.1); then, it uses the proposed *Algorithm 4* (see Section 4.3) to obtain the ensembled anomaly detection result. The ensembled compared methods are noted as  $OC$ - $SVM$  (*Ens*),  $AAKR$  (*Ens*),  $GMM$  (*Ens*) and  $AE$  (*Ens*).

Table 4: F-score of the proposed  $AE$ - $GAN$  with variants *a)* and *b)*, and comparison methods for the automatic door dataset.

Percentile $c$	Compared Methods								Proposed Method	
	$OC$ - $SVM$	$OC$ - $SVM$ ( <i>Ens</i> )	$AAKR$	$AAKR$ ( <i>Ens</i> )	$GMM$	$GMM$ ( <i>Ens</i> )	$AE$	$AE$ ( <i>Ens</i> )	$AE$ - $GAN$ ( <i>a</i> )	$AE$ - $GAN$ ( <i>b</i> )
100%	0.5952	0.6207	0.4727	0.6452	N/A	0.6333	0.5391	0.6230	<b>0.7312</b>	<b>0.7312</b>
95%	N/A	0.6207	N/A	0.6452	N/A	0.6333	N/A	0.6230	0.7174	<b>0.7312</b>
90%	N/A	0.5965	N/A	0.6452	N/A	0.6333	N/A	0.6230	0.7253	<b>0.7312</b>
85%	N/A	0.5965	N/A	0.6557	N/A	0.6333	N/A	0.6230	0.7191	<b>0.7312</b>
80%	N/A	0.5965	N/A	0.6557	N/A	0.6333	N/A	0.6452	0.7273	<b>0.7527</b>
75%	N/A	0.5965	N/A	0.6557	N/A	0.6102	N/A	0.5763	<b>0.7750</b>	0.7692
70%	N/A	0.5283	N/A	0.6555	N/A	0.6332	N/A	0.5574	<b>0.7692</b>	0.6750
65%	N/A	0.4906	N/A	0.6333	N/A	0.5614	N/A	0.5246	<b>0.6761</b>	0.5854
60%	N/A	0.5091	N/A	0.6102	N/A	0.6000	N/A	0.5246	<b>0.6364</b>	0.5500



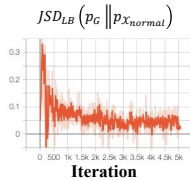
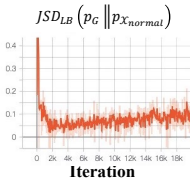
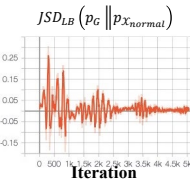
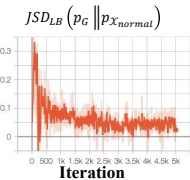
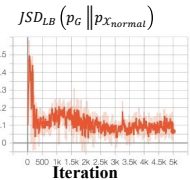
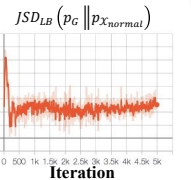
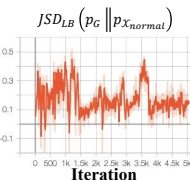
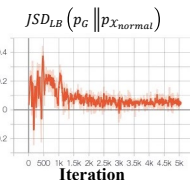
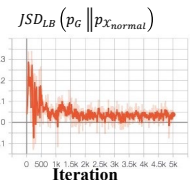
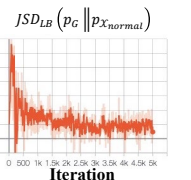
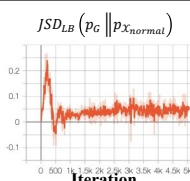
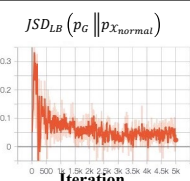
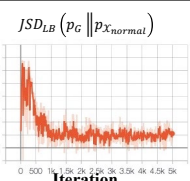
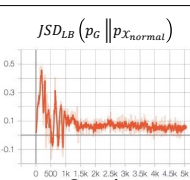
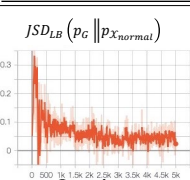
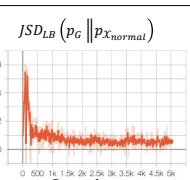
Epoch number= 1000		Epoch number= 5000		Optimal Epoch number= 1000	
JS Divergence					
	Discriminator: $D()$				
	$D()$ iteration $k=1$	$D()$ iteration $k=5$	$D()$ iteration $k=10$	$D()$ iteration $k=20$	Optimal $D()$ iteration $k=5$
JS Divergence					
	Latent space size=1    Latent space size=2    Latent space size=3    Latent space size=4				Optimal Latent space size=4
JS Divergence					
	# of hidden layer=1    # of hidden layer=2    # of hidden layer=3				Optimal # of hidden layer=2
JS Divergence					
	hidden neurons=50    hidden neurons=100    hidden neurons=200				Optimal # of hidden neurons=200
JS Divergence					

Figure 14: Results of the AE-GAN hyper-parameters optimization.

Table 4 reports the comparison of anomaly detection results. Different anomaly detection results are obtained by adjusting the percentile  $c$  in *Algorithm 4*, and the proposed AE-GAN with both variants *a*) and *b*) achieves the best result for a variety of percentile values  $c$ . For variant *a*), best F-score 0.7750 is obtained when  $c$  is 75%, for variant *b*), best F-score 0.7527 is obtained when  $c$  is 85%. Overall, variant *a*) is better than *b*), because variant *b*) introduces discrete time into the data space (see Equation (26)), which makes it difficult for the generator, which has a continuous data space, to fit the data space containing discrete times. Note that *OC-SVM*, *AAKR*, *AE* have results only when  $c = 100\%$ , since a certain anomaly score threshold is set, and *GMM* has no results because the high dimensionality of data makes matrix computation infeasible.

By using the proposed improved AdaBoost ensemble learning (*Algorithm 4*), the F-score and *AUC* is boosted for nearly all the compared methods. Another advantage of *Algorithm 4* is that it can automatically filter the task-related features in data by assigning different weights to the base anomaly detector (see

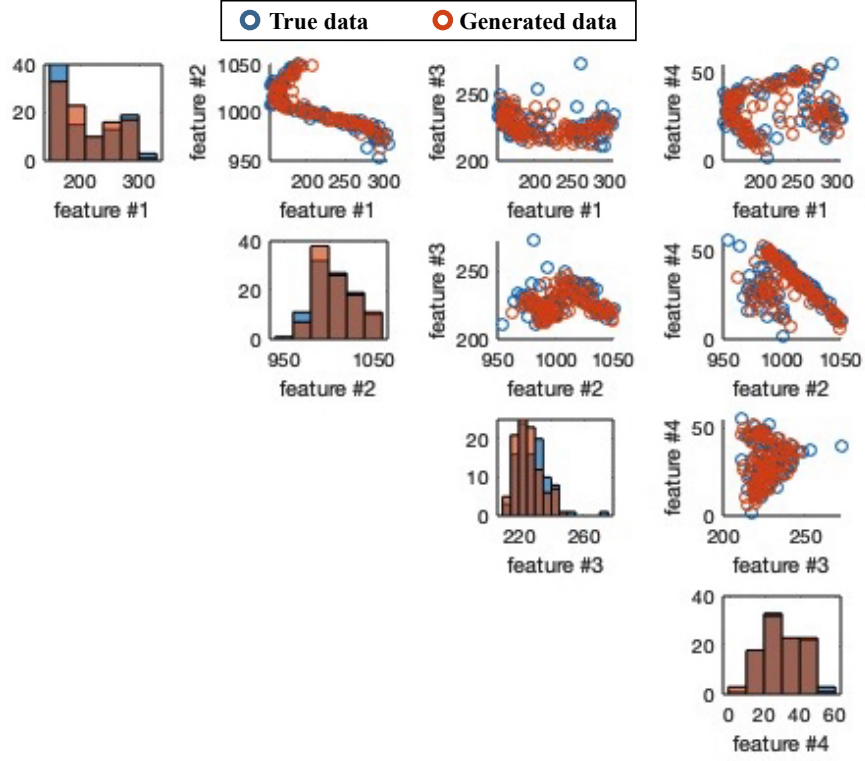


Figure 15: An example of true data distribution and the generated data distribution produced by the optimal *AE-GAN*. The true data comes from the normal components signal at time 400.

Figure 16). This can be confirmed by the findings in Figure 16, as we observe that the weights of the base anomaly detectors suddenly drop at time 500, when, interestingly, the value of the original signal (Figure 11) becomes a constant, which means that the signal after time 500 is irrelevant to component health monitoring.

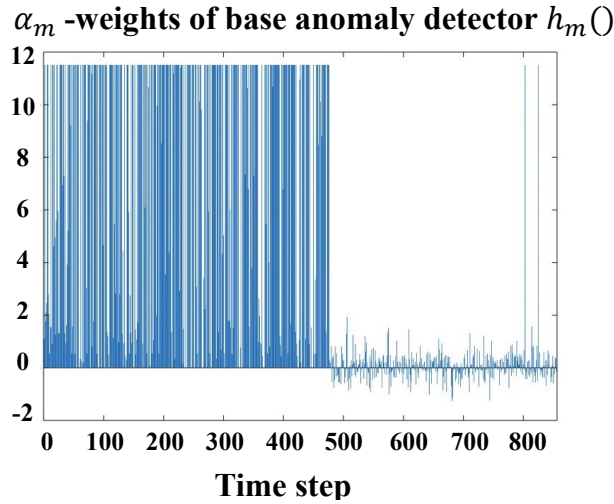


Figure 16: Example of base anomaly detectors weight  $\alpha_m$  of, e.g. *AE-GAN* variant *a*), at each time step.

In order to obtain the comprehensive anomaly detection performance, we look at the *ROC* curve adjusting



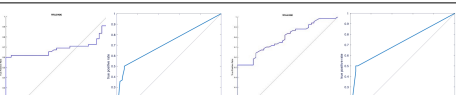
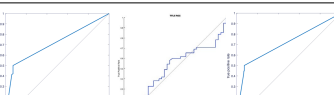
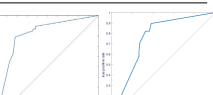
	Compared Methods								Proposed Method	
Metric	<i>OC-SVM</i>	<i>OC-SVM (Ens)</i>	<i>AAKR</i>	<i>AAKR (Ens)</i>	<i>GMM</i>	<i>GMM (Ens)</i>	<i>AE</i>	<i>AE (Ens)</i>	<i>AE-GAN (a)</i>	<i>AE-GAN (b)</i>
ROC Curve					N/A					
AUC	0.6811	0.7146	0.7601	0.7301	N/A	0.7182	0.5041	0.6981	<b>0.8289</b>	0.7244

Figure 17: *ROC* curve and *AUC* of the proposed *AE-GAN* with variants *a*) and *b*), and compared methods for real industrial dataset. Note that x axis denotes false positive rate, y axis denotes true positive rate.

the percentile  $c$  and obtaining the *AUC* (Figure 17), which shows that the proposed *AE-GAN* variant *a*) outperforms any other compared methods. The interpretation of this result is that the real industrial data is very complex: similar to the investigation of the synthetic case study, it contains a distribution with non-smooth density, manifold distribution, which will make the compared methods unable to model the real industrial data distribution.

Table 5: Number of components in Test Set 1 and Test Set 2.

Test Set 1		Test Set 2	
#Test normal components	#Test components with fault A	#Test normal components	#Test components with fault B
18	22	18	33

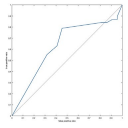
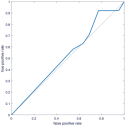
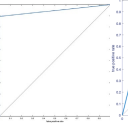
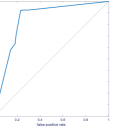
Metric	Test Set 1		Test Set 2	
	<i>AE-GAN (a)</i>	<i>AE-GAN (b)</i>	<i>AE-GAN (a)</i>	<i>AE-GAN (b)</i>
<b>ROC Curve</b>				
<b>AUC</b>	<b>0.6304</b>	0.5407	<b>0.9474</b>	0.8469

Figure 18: *ROC* curve and *AUC* of the proposed *AE-GAN* with variants *a*) and *b*) for Test Sets 1 and 2. Note that x axis denotes false positive rate, y axis denotes true positive rate.

To deeply analyze the performance of the proposed method, we construct two test sets from the industrial dataset. Test Set 1 contains normal condition patterns and only anomalous patterns of fault class A and Test Set 2 contains normal condition patterns and anomalous patterns of only fault class B, as reported in Table 5. The *ROC* curve and *AUC* on Test Sets 1 and 2 show that the proposed method can better detect the anomalies of fault class B than A. The reason is that, according to the original data, data distribution of fault type A is almost the same with the distribution of normal data, whereas the data distribution of fault type B is clearly distinguishable with normal data.

## 6. Conclusion

In this paper, an AdaBoost ensembled *AE-GAN* anomaly detection method based on the use of *GAN* and AdaBoost ensemble learning has been proposed for the industrial case where abnormal data is not available. For obtaining the anomaly score, e.g. reconstruction error, the latent variable corresponding to the data pattern in *GAN* needs to be queried and we propose to embed an auxiliary encoder in front of the generator to avoid local optimal solutions for data with manifold distribution. Furthermore, we derive the lower bound of Jensen-Shannon divergence between generator distribution and normal data distribution to optimize the *AE-GAN* hyperparameters. To overcome real industrial challenges, like 1) *the densities of data distributions are not smooth* and 2) *the curse of dimensionality*, we propose to add adaptive noise on data and adapt the AdaBoost algorithm to integrate *AE-GAN* base anomaly detectors which treat each time window separately for anomaly detection. Extensive experiments are conducted on both synthetic and real industrial data sets, which demonstrate that the proposed ensembled *AE-GAN* anomaly detection method outperforms state-of-the-art anomaly detection methods for long-term multivariate time series.

## Acknowledgment

This project has received funding from the Shift2Rail Joint Undertaking under the European Union's Horizon 2020 research and innovation programme under grant agreement No 101015423. The information in this document is provided "as is", and no guarantee or warranty is given that the information is fit for any particular purpose. The content of this document reflects only the author's view – the Shift2Rail Joint Undertaking is not responsible for any use that may be made of the information it contains. The users use the information at their sole risk and liability. Mingjing Xu gratefully acknowledges the financial support from the China Scholarship Council (No. 201606420061).

## References

- [1] A. Cook, G. Misirli, Z. Fan, Anomaly Detection for IoT Time-Series Data: A Survey, *IEEE Internet of Things Journal* (2020). doi:10.1109/jiot.2019.2958185.
- [2] C. M. Rocco S., E. Zio, A support vector machine integrated system for the classification of operation anomalies in nuclear components and systems, *Reliability Engineering and System Safety* (2007). doi:10.1016/j.res.2006.02.003.
- [3] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, E. Vázquez, Anomaly-based network intrusion detection: Techniques, systems and challenges, *Computers and Security* (2009). doi:10.1016/j.cose.2008.08.003.
- [4] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, S. C. de Alvarenga, A survey of intrusion detection in Internet of Things (2017). doi:10.1016/j.jnca.2017.02.009.
- [5] T. Fuse, K. Kamiya, Statistical Anomaly Detection in Human Dynamics Monitoring Using a Hierarchical Dirichlet Process Hidden Markov Model, *IEEE Transactions on Intelligent Transportation Systems* (2017). doi:10.1109/TITS.2017.2674684.
- [6] H. Kim, J. Park, K. Min, K. Huh, Anomaly Monitoring Framework in Lane Detection With a Generative Adversarial Network, *IEEE Transactions on Intelligent Transportation Systems* 1 (c) (2020) 1–13. doi:10.1109/tits.2020.2973398.
- [7] W. Luo, W. Liu, D. Lian, J. Tang, L. Duan, X. Peng, S. Gao, Video Anomaly Detection With Sparse Coding Inspired Deep Neural Networks, *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019). doi:10.1109/tpami.2019.2944377.
- [8] H. Sarmadi, A. Karamodin, A novel anomaly detection method based on adaptive Mahalanobis-squared distance and one-class kNN rule for structural health monitoring under environmental effects, *Mechanical Systems and Signal Processing* (2020). doi:10.1016/j.ymssp.2019.106495.
- [9] Z. Shi, A. Chehade, A dual-lstm framework combining change point detection and remaining useful life prediction, *Reliability Engineering & System Safety* 205 (2021) 107257.
- [10] S. Tolo, X. Tian, N. Bausch, V. Becerra, T. Santhosh, G. Vinod, E. Patelli, Robust on-line diagnosis tool for the early accident detection in nuclear power plants, *Reliability Engineering & System Safety* 186 (2019) 110–119.
- [11] D. M. Hawkins, Identification of Outliers, 1980. doi:10.1007/978-94-015-3994-4.
- [12] H. Jeong, B. Park, S. Park, H. Min, S. Lee, Fault detection and identification method using observer-based residuals, *Reliability Engineering & System Safety* 184 (2019) 27–40.
- [13] L. Zhang, J. Lin, R. Karim, An angle-based subspace anomaly detection approach to high-dimensional data: With an application to industrial fault detection, *Reliability Engineering & System Safety* 142 (2015) 482–497.
- [14] A. A. Jimenez, C. Q. G. Muñoz, F. P. G. Márquez, Dirt and mud detection and diagnosis on a wind turbine blade employing guided waves and supervised learning classifiers, *Reliability Engineering & System Safety* 184 (2019) 2–12.

- [15] C. Duan, V. Makis, C. Deng, A two-level bayesian early fault detection for mechanical equipment subject to dependent failure modes, *Reliability Engineering & System Safety* 193 (2020) 106676.
- [16] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, K. J. Kim, A survey of deep learning-based network anomaly detection, *Cluster Computing* (2019) 1–13.
- [17] P. Baraldi, F. Di Maio, M. Rigamonti, E. Zio, R. Seraoui, Unsupervised clustering of vibration signals for identifying anomalous conditions in a nuclear turbine, *Journal of Intelligent & Fuzzy Systems* 28 (4) (2015) 1723–1731.
- [18] Z. Ghafoori, S. M. Erfani, J. C. Bezdek, S. Karunasekera, C. Leckie, LN-SNE: Log-Normal Distributed Stochastic Neighbor Embedding for Anomaly Detection, *IEEE Transactions on Knowledge and Data Engineering* (2020). doi:10.1109/TKDE.2019.2934450.
- [19] H. J. Shin, D. H. Eom, S. S. Kim, One-class support vector machines - An application in machine fault detection and classification, *Computers and Industrial Engineering* (2005). doi:10.1016/j.cie.2005.01.009.
- [20] Y. Xiao, H. Wang, L. Zhang, W. Xu, Two methods of selecting gaussian kernel parameters for one-class svm and their application to fault detection, *Knowledge-Based Systems* 59 (2014) 75–84.
- [21] L. Li, R. J. Hansman, R. Palacios, R. Welsch, Anomaly detection via a Gaussian Mixture Model for flight operation and safety monitoring, *Transportation Research Part C: Emerging Technologies* (2016). doi:10.1016/j.trc.2016.01.007.
- [22] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, K. J. Kim, A survey of deep learning-based network anomaly detection, *Cluster Computing* 22 (1) (2019) 949–961.
- [23] S. M. Erfani, S. Rajasegarar, S. Karunasekera, C. Leckie, High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning, *Pattern Recognition* (2016). doi:10.1016/j.patcog.2016.03.028.
- [24] T. Shon, J. Moon, A hybrid machine learning approach to network anomaly detection, *Information Sciences* (2007). doi:10.1016/j.ins.2007.03.025.
- [25] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey (2009). doi:10.1145/1541880.1541882.
- [26] D. Yang, A. Usynin, J. W. Hines, Anomaly-based intrusion detection for scada systems, in: 5th intl. topical meeting on nuclear plant instrumentation, control and human machine interface technologies (npic&hmit 05), 2006, pp. 12–16.
- [27] P. Baraldi, F. Di Maio, P. Turati, E. Zio, Robust signal reconstruction for condition monitoring of industrial components via a modified auto associative kernel regression method, *Mechanical Systems and Signal Processing* 60 (2015) 29–44.
- [28] R. Laxhammar, G. Falkman, Online learning and sequential anomaly detection in trajectories, *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2014). doi:10.1109/TPAMI.2013.172.
- [29] H. Ozkan, F. Ozkan, S. S. Kozat, Online Anomaly Detection Under Markov Statistics With Controllable Type-I Error, *IEEE Transactions on Signal Processing* (2016). doi:10.1109/TSP.2015.2504345.
- [30] Y. Feng, Y. Yuan, X. Lu, Learning deep event models for crowd anomaly detection, *Neurocomputing* (2017). doi:10.1016/j.neucom.2016.09.063.
- [31] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, N. V. Chawla, A Deep Neural Network for Unsupervised Anomaly Detection and Diagnosis in Multivariate Time Series Data, *Proceedings of the AAAI Conference on Artificial Intelligence* (2019). arXiv:1811.08055, doi:10.1609/aaai.v33i01.33011409.
- [32] Y. Li, L. Yan, J. Wang, H. and Jin, Anomaly Detection of Time Series With Smoothness-Inducing Sequential Variational Auto-Encoder, *IEEE Transactions on Neural Networks and Learning Systems* (2020).
- [33] E. A. Mohammad Sabokrou, Mahmood Fathy, Guoying Zhao, Deep End-to-End One-Class Classifier, *IEEE Transactions on Neural Networks and Learning Systems* (2020). doi:10.1109/TNNLS.2020.2979049.
- [34] M. N. Kurt, Y. Yilmaz, X. Wang, Real-Time Nonparametric Anomaly Detection in High-Dimensional Settings, *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020). arXiv:1809.05250, doi:10.1109/tpami.2020.2970410.
- [35] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: *Advances in neural information processing systems*, 2014, pp. 2672–2680.
- [36] J. Wu, Z. Zhao, C. Sun, R. Yan, X. Chen, Fault-Attention Generative Probabilistic Adversarial Autoencoder for Machine Anomaly Detection, *IEEE Transactions on Industrial Informatics* (2020). doi:10.1109/tii.2020.2976752.
- [37] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, G. Langs, Unsupervised anomaly detection with generative adversarial networks to guide marker discovery, in: *International conference on information processing in medical imaging*, Springer, 2017, pp. 146–157.
- [38] T. Schlegl, P. Seeböck, S. M. Waldstein, G. Langs, U. Schmidt-Erfurth, f-AnoGAN: Fast unsupervised anomaly detection with generative adversarial networks, *Medical Image Analysis* (2019). doi:10.1016/j.media.2019.01.010.
- [39] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, S.-K. Ng, Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks, in: *International Conference on Artificial Neural Networks*, Springer, 2019, pp. 703–716.
- [40] G. E. Hinton, R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, *Science* (2006). doi:10.1126/science.1127647.
- [41] H.-J. Xing, W.-T. Liu, Robust AdaBoost based ensemble of one-class support vector machines, *Information Fusion* 55 (August 2019) (2019) 45–58. doi:10.1016/j.inffus.2019.08.002. URL <https://doi.org/10.1016/j.inffus.2019.08.002>
- [42] T. Hastie, S. Rosset, J. Zhu, H. Zou, Multi-class adaboost, *Statistics and its Interface* 2 (3) (2009) 349–360.
- [43] D. P. Kingma, J. L. Ba, Adam: A method for stochastic optimization, in: *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015. arXiv:1412.6980.
- [44] S. Martin Arjovsky, L. Bottou, Wasserstein generative adversarial networks, in: *Proceedings of the 34 th International Conference on Machine Learning*, Sydney, Australia, 2017.
- [45] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey (2009).
- [46] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, A. A. Efros, Generative visual manipulation on the natural image manifold, in: *European conference on computer vision*, Springer, 2016, pp. 597–613.

- [47] P. Bojanowski, A. Joulin, D. Lopez-Paz, A. Szlam, Optimizing the latent space of generative networks, arXiv preprint arXiv:1707.05776 (2017).
- [48] A. Radford, L. Metz, S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, arXiv preprint arXiv:1511.06434 (2015).
- [49] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets (2014) 2672–2680.
- [50] L. Metz, B. Poole, D. Pfau, J. Sohl-Dickstein, Unrolled generative adversarial networks, arXiv preprint arXiv:1611.02163 (2016).
- [51] T. D. Luu, J. Fadili, C. Chesneau, Sampling from non-smooth distributions through langevin diffusion, *Methodology and Computing in Applied Probability* (2020) 1–29.
- [52] R. Liu, B. Yang, E. Zio, X. Chen, Artificial intelligence for fault diagnosis of rotating machinery: A review, *Mechanical Systems and Signal Processing* 108 (2018) 33–47.
- [53] M. Arjovsky, L. Bottou, Towards principled methods for training generative adversarial networks, arXiv preprint arXiv:1701.04862 (2017).
- [54] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.
- [55] P. Grzegorzewski, E. Mrówka, Trapezoidal approximations of fuzzy numbers—revisited, *Fuzzy Sets and Systems* 158 (7) (2007) 757–768.
- [56] R. Perdisci, G. Gu, W. Lee, Using an ensemble of one-class svm classifiers to harden payload-based anomaly detection systems, in: *Sixth International Conference on Data Mining (ICDM’06)*, IEEE, 2006, pp. 488–498.
- [57] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, J. C. Platt, Support vector method for novelty detection, in: *Advances in neural information processing systems*, 2000, pp. 582–588.
- [58] A. Basharat, A. Gritai, M. Shah, Learning object motion patterns for anomaly detection and improved object detection, in: *2008 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2008, pp. 1–8.
- [59] M. Sakurada, T. Yairi, Anomaly detection using autoencoders with nonlinear dimensionality reduction, in: *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, 2014, pp. 4–11.
- [60] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 315–323.
- [61] A. L. Maas, A. Y. Hannun, A. Y. Ng, Rectifier nonlinearities improve neural network acoustic models, in: *Proc. icml*, Vol. 30, 2013, p. 3.