

Generative Adversarial Networks With AdaBoost Ensemble Learning for Anomaly Detection in High-Speed Train Automatic Doors

Mingjing Xu, Piero Baraldi^{ID}, Xuefei Lu, and Enrico Zio^{ID}, *Senior Member, IEEE*

Abstract—Due to the scarcity of abnormal condition data in components of transportation systems, only normal condition data are typically used to train models for anomaly detection. One of the main challenges is the difficulty of properly representing the data distribution which is typically non-smooth, high-dimensional and on a manifold. This work develops an anomaly detection model based on an Auto-Encoder (AE) formed by the generator of a Generative Adversarial Network (GAN) and an auxiliary encoder to capture the sophisticated data structure. The reconstruction error of the AE is, then, used as anomaly score to detect anomalies. Additionally, an adaptive noise is added to the data to make easier the GAN optimization, an AdaBoost-based ensemble learning scheme is used to improve detection performance and a new approach for setting the hyperparameters of the AE-GAN model based on the derivation of a lower bound of the Jensen-Shannon divergence between generator and normal condition data distributions is developed. The method has been applied to synthetic and real data collected from automatic doors of high-speed trains.

Index Terms—High-speed train automatic door, anomaly detection, high dimensional time series, manifold distribution, generative adversarial networks, AdaBoost ensemble learning.

I. INTRODUCTION

ANOMALY detection aims at identifying novel and unexpected patterns within the data collected [1]. It plays a critical role in several industrial domains, such as network intrusion detection [2], transportation monitoring [3], video anomalous behavior recognition [4] and anomaly detection

in transportation systems. These latter applications are made possible by the fact that sensors measure a variety of signals for the control operation and monitoring of behavior of critical components of transportation systems can allow improving the efficiency of operation and reducing the cost of maintenance. Anomaly detection approaches are typically categorized as supervised, unsupervised and one-class classification [5]. 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. They typically face the problems of dealing with imbalanced datasets, being abnormal condition data typically rare and of the variability of the operating conditions which causes major modifications to the data distributions. In [6], noise-filtered and under-sampling methods are combined to address the issue of imbalanced data. In [7], the ‘TrAdaBoost’ method, which extends the classical AdaBoost method to deal with the situation in which data distributions change due to variations of operating conditions using a domain transform method is proposed. 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 are sufficiently dissimilar to normal condition patterns to allow discriminating them [8]. On the other hand, in many industrial applications anomalous conditions are rare and changes in operating and environmental conditions cause variations of the measured signals that are larger than the variations caused by the onset of a degradation of a component, at least at the early stages after its occurrence. For this reason, this work considers detection methods based on **one-class classification** [9], which are trained on a dataset containing only normal condition patterns. Examples of classification methods applied to the one-class classification problems are: Support Vector Machines (SVMs) [10], nearest neighbor-based methods [11], statistical-based models [12] and Deep Learning based (DL) [13]. 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 method which integrates Support Vector Data Description (Deep-SVDD) with a deep feature extraction for anomaly detection has been developed in [14] and applied to image benchmark datasets. Nearest neighbors-based methods use properly defined measures of

Manuscript received 16 June 2021; revised 11 January 2022 and 28 June 2022; accepted 8 August 2022. Date of publication 29 September 2022; date of current version 5 December 2022. The work of Mingjing Xu was supported in part by the China Scholarship Council under Grant 201606420061. The work of Piero Baraldi was supported in part by the European project RECET4RAIL and in part by the call BRIC-2018 of the National Institute for Insurance against Accidents at Work—INAIL “Smart maintenance of industrial plants and civil structures by 4.0 monitoring technologies and prognostic approaches—MAC4PRO. The work of Enrico Zio was supported by the call BRIC-2018 of the National Institute for Insurance against Accidents at Work—INAIL “Smart maintenance of industrial plants and civil structures by 4.0 monitoring technologies and prognostic approaches—MAC4PRO. The Associate Editor for this article was M. Zhou. (Corresponding author: Piero Baraldi.)

Mingjing Xu and Piero Baraldi are with the Department of Energy, Politecnico di Milano, 20156 Milan, Italy (e-mail: piero.baraldi@polimi.it).

Xuefei Lu is with the SKEMA Business School, Université Côte d’Azur, 92150 Paris, France.

Enrico Zio is with the Department of Energy, Politecnico di Milano, 20156 Milan, Italy, also with Centre for Research on Risk and Crises (CRC), MINES ParisTech, PSL Research University, 06904 Sophia Antipolis, France, and also with Aramis Srl, 20121 Milan, Italy.

Digital Object Identifier 10.1109/TITS.2022.3203871

dissimilarity among patterns and assume that normal condition data are located in dense neighborhoods, whereas anomalies are far from their closest neighbors [15]. For example, the Auto Associative Kernel Regression (AAKR) method has been used to detect anomalous conditions in an energy production plant in [16]. 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 [16] and [17]. Then, if the reconstruction error exceeds an alarm threshold, the test pattern is identified as abnormal. Statistics-based methods, such as Gaussian Mixture Models (*GMMs*) statistics, construct probabilistic models describing the normal condition patterns. Then, an anomaly is detected if the likelihood of occurrence of the test pattern is lower than a predefined threshold [12]. For example, in [18] a novel multiscale drift detection test is proposed to solve the classification problem when data distributions change over time. In [19], novel outlier detection strategy, based on SetMembership Filtering model is developed to identify measurements corrupted by outliers. A deep generative model stacked with multiple *GMM*-layers has been proposed to detect abnormal events in video surveillance [20]. 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, and the flexibility of designing problem-specific loss functions. In [21], a pairwise Gaussian loss function is developed to address the problem of intra-class compactness and successfully applied to synthetic data. In [22], a full-center loss function is used to improve the separability of features in fraud detection. In [23], a method combining auto-encoders, which extracts nonlinear features, and backpropagation to obtain a fault diagnosis index is developed and applied to a benchmark case. 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 [24]. However, detecting anomalies using conventional deep learning methods, such as *RNNs*, Auto-Encoders and hybrid *DNNs*, can be challenging due to the long-term time dependency and cross correlation among time series [25].

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 [26]. *GANs* have been shown able to learn dataset with complex structure, e.g., spheres or torus manifolds [27], and of reproduce real data distribution for data augmentation [28]. *GAN*-based anomaly detection techniques were first proposed for medical image analysis [29]. In the transport field, a data augmentation method has been developed for synthesizing anomalies of the minority classes in lane detection specifically. A *GAN* is used to learn the distribution of anomalous condition patterns, and generate synthetic anomalies, which are, then, used to train a supervised anomaly detection model [3]. A limitation

of the method is that it cannot be used when abnormal condition patterns are completely missing. In [24], a deep one-class classifier formed by an auto-encoder and a discriminator trained in an adversarial way is developed. In [30], *GAN* and Variational Auto-Encoder(*VAE*) are combined to synthesize auxiliary positive patterns in a problem of predicting lung cancer.

In this context, the objective of the present work is to develop a methodology for detecting anomalies in components behaviour using measurements collected from components of critical systems of transportation systems. To this aim, we develop an **Auto-Encoder aided *GAN* (*AE-GAN*)** which allow associating an anomaly score to the multidimensional signal time series. 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. Finally, to improve the detection performance, an ensemble of anomaly detectors is developed by adapting the AdaBoost ensemble learning scheme. A test pattern is identified as anomalous if the reconstruction of the ensemble of Auto-Encoders error is larger than a certain threshold. Two different *AE-GANs* variants are considered in this work: variant *a*) sets up a dedicated *AE-GAN* for every time slice of the multivariate time series, whereas variant *b*) sets up a single universal *AE-GAN* to be applied to all time slices.

A synthetic case study concerning three complex distributions of normal condition patterns, e.g. Cone, Two Sphere and Bowl distributions, is used to verify the performance of the *AE-GAN*. Then, the developed method is applied to a real-world industrial case study concerning automatic doors of high speed trains. Notice that failures of automatic doors are a cause of unavailability of high-speed train that has recently attracted the attention of the main stakeholders of the transportation systems. The performance of the proposed method has been compared to six state-of-the-art anomaly detection techniques including *OC-SVM*, *AAKR*, *GMM* and deep-learning based *AE*, *VAE*- and Deep *SVDD* algorithms.

The contributions of this work are: 1) a combined framework of *AE* and *GAN* is developed to identify anomalous patterns in the situation, common in transportation systems, in which abnormal condition data are not available and normal condition data are characterized by complex distributions, e.g. with manifold support; 2) the lower bound of Jensen-Shannon (*JS*) divergence is used to guide the setting of the *AE-GAN* hyper-parameters; 3) an adaptive noise is added to the input data in case of non-smooth data distributions; 4) The use of the Adaboost algorithm has allowed to extend the *AE-GAN* to treat multidimensional long-term time series data.

The remaining of the paper is organized as follows: Section II states the problem and illustrates the work objectives; Section III introduces the background and preliminaries of the proposed methodology and Section IV specializes the proposed methodology of anomaly detection for long-term multivariate time series; Section V 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 VI.

II. PROBLEM STATEMENT

We consider N_{nor} components operating in normal conditions. For each component, N_f features related to its health condition are measured during operation. The $N_f \times L$ matrix, \mathbf{X}^r , $r = 1, \dots, N_{nor}$, contains the time series of length L collected during component operation in normal conditions. The aim of this work is to build an anomaly detection model to identify the normal/abnormal health state of a test component given the N_f -dimensional time series \mathbf{X}^{test} measured during its operation. Notice that the situation in which only normal condition data are available is common in many applications of transportation systems involving, for example, safety critical or newly designed components.

III. PRELIMINARIES AND BACKGROUND

A. Generative Adversarial Networks

Let $\mathcal{X} \subseteq \mathbb{R}^{N_x}$ be the space of the training data whose distribution is p_{data} . A GAN consists of a generator and a discriminator, where the generator is a multilayer perceptron aiming at generating patterns from the same distribution of the training data and the discriminator is a multilayer perceptron aiming at providing the probability that a test pattern \mathbf{x} comes from the same data distribution [26].

The generator $G(\mathbf{z}; \theta_G) : \mathcal{Z} \rightarrow \mathcal{X}$ with associated parameters θ_G maps a latent variable \mathbf{z} from the latent space $\mathcal{Z} \subseteq \mathbb{R}^{N_z}$ to the data space of the patterns $\mathcal{X} \subseteq \mathbb{R}^{N_x}$. The entries of the latent variable \mathbf{z} , $\mathbf{z} \in \mathcal{Z}$ are independent among them and follow a standard Gaussian distribution $\mathcal{N}(0, 1)$. The discriminator $D(\mathbf{x}; \theta_D) : \mathcal{X} \rightarrow [0, 1]$ with associated parameters θ_D discriminates whether a test pattern \mathbf{x} belongs to \mathcal{X} (true) or is generated by the generator (fake) 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 the latent variable \mathbf{z} .

B. Auto-Encoders

An Auto-Encoder is a neural network composed of an encoder and a generator, trained to replicate its input data [31]. 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: $E(\mathbf{x}; \theta_E) = f(\mathbf{W}_E \mathbf{x} + \mathbf{b}_E)$, where the parameters $\theta_E = \{\mathbf{W}_E, \mathbf{b}_E\}$ are the weight matrix \mathbf{W}_E of size $N_z \times N_x$

and the offset vector \mathbf{b}_E of dimension N_z . The generator G maps back the resulting latent variable \mathbf{z} into the reconstructed N_x -dimensional vector $\hat{\mathbf{x}}$. Its typical form is similar to E : $G(\mathbf{z}; \theta_G) = f_G(\mathbf{W}_G \mathbf{z} + \mathbf{b}_G)$, where the parameters $\theta_G = \{\mathbf{W}_G, \mathbf{b}_G\}$ are weight matrix of size $N_x \times N_z$ and the offset vector \mathbf{b}_G of dimension N_x ; f_G is 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 $\hat{\mathbf{x}}$: $\mathcal{L}_{rec} = \mathbb{E}_{\mathbf{x} \in \mathcal{X}} \|\mathbf{x} - \hat{\mathbf{x}}\|^2 = \mathbb{E}_{\mathbf{x} \in \mathcal{X}} \|\mathbf{x} - G(E(\mathbf{x}))\|^2$, where $\|\cdot\|$ denotes the L2 norm.

C. AdaBoost Ensemble Learning

AdaBoost is an ensemble learning algorithm which constructs a classifier as a linear combination of several weak classifiers [32]. 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 ensemble classifier $H : \mathcal{X} \rightarrow \{-1, 1\}$ by linearly combining the base classifiers $h_t : \mathcal{X} \rightarrow \{-1, 1\}$:

$$H(\mathbf{x}) = \text{sgn} \left(\sum_{t=1}^T \alpha_t h_t(\mathbf{x}) \right) \quad (2)$$

where h_t denotes the t -th base classifier, α_t the weight assigned to h_t and T the number of base classifiers.

D. Adam Optimization

The Adam optimization algorithm is an extension of the stochastic gradient descent algorithm [33]. It combines the advantages of a) Adaptive Gradient Algorithm (AdaGrad), which uses a per-parameter learning rate to improve its performances on problems with sparse gradients and b) Root Mean Square Propagation (RMSProp), which iteratively adapts the parameter learning rates on the basis of how quickly the gradients of the weights is changing, to improve its performance in non-stationary problems.

IV. ANOMALY DETECTION METHODOLOGY MODEL

A. Base Anomaly Detector With Auto Encoder Aided Generative Adversarial Networks (AE-GAN)

The proposed anomaly detector is based on the use of a GAN to reconstruct the expected signal behavior in normal conditions, from which the reconstruction error can be obtained and used to discriminate normal from abnormal condition patterns. Its development requires: 1) training the GAN model on normal condition patterns for reproducing the distribution of the normal condition data and 2) training the AE to reconstruct the expected signal behavior in normal conditions.

In 1), the GAN is trained to minimize the Jensen Shannon Divergence $JSD(p_G \| p_{\mathcal{X}_{nor}})$, where \mathcal{X}_{nor} denotes the set of patterns collected from components operating in normal

conditions, $p_{\mathcal{X}_{nor}}$ their probability distribution and p_G the generated pattern probability distribution. Notice that if the GAN generator were perfectly trained, then $JSD(p_G \| p_{\mathcal{X}_{nor}})$ would converge to 0 [34]. Let $\theta_G = \{W_G, b_G\}$ and $\theta_D = \{W_D, b_D\}$ denote the generator and discriminator parameters, respectively. Similarly to the AutoEncoder (Section III-B), the discriminator D is formulated as: $D(x; \theta_D) = f_D(W_D x + b_D)$ where W_D is a $N_z \times N_x$ weight matrix, 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 the purpose of anomaly detection, we set $p_{data} = p_{\mathcal{X}_{nor}}$ and p_z as a Gaussian distribution $\mathcal{N}(0, 1)$ of independent variables and we address the minmax problem of Equation (1).

Before the optimization of the generator parameter θ_G , the discriminator parameter $\theta_D^*(\theta_G)$ is set by using a gradient optimization method based on Adam (Section III):

$$\theta_D^{(k)} = \theta_D^{(k-1)} + \eta \cdot \text{Adam}(\nabla_{\theta_D} \mathcal{F}(\theta_D^{(k-1)}, \theta_G); \beta_1, \beta_2) \quad (3)$$

$$\theta_D^*(\theta_G) = \lim_{k \rightarrow \infty} \theta_D^{(k)} \quad (4)$$

where the updating term $\text{Adam}(\nabla_{\theta_D} \mathcal{F}(\theta_D^{(k-1)}, \theta_G); \beta_1, \beta_2)$ is determined by the gradient of the loss function \mathcal{F} with respect to θ_D , and β_1 and β_2 are the control parameters of Adam [33], and η is the learning rate, and $\theta_D^{(k)}$ 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 based on Adam (Section III):

$$\theta_G = \theta_G - \eta \cdot \text{Adam}(\nabla_{\theta_G} \mathcal{F}(\theta_D^*(\theta_G), \theta_G); \beta_1, \beta_2) \quad (5)$$

where the updating term $\text{Adam}(\nabla_{\theta_G} \mathcal{F}(\theta_D^*(\theta_G), \theta_G); \beta_1, \beta_2)$ is determined by the gradient of the loss function \mathcal{F} with respect to θ_G . Note that for each updating step of θ_G (Equation (5)), there are k updating steps of $\theta_D^{(k)}$ (Equation (3)), because $\theta_D^{(k)}$ depends on θ_G .

In 2), to obtain the reconstruction \hat{x} of data x , it is necessary to query its latent variable $z \in \mathcal{Z}$ and, then, to map z into the data space \mathcal{X}_{nor} by using the generator, $\hat{x} = G(z)$. According to [35], the search of z_{optimal} is treated as an optimization task, i.e. $\min_z \|x - G(z; \theta_G^*)\|^2$. In this work, an auxiliary auto-encoder is proposed for efficiently minimizing the reconstruction error:

$$\mathcal{L}_{rec}(x; \theta_E, \theta_G^*) = \mathbb{E}_{x \in \mathcal{X}_{nor}} \|x - G(E(x; \theta_E); \theta_G^*)\|^2 \quad (6)$$

from which we obtain

$$\theta_E^* = \underset{\theta_E}{\text{argmin}} \mathcal{L}_{rec}(x; \theta_E, \theta_G^*) \text{ and } z_{\text{optimal}} = E(x; \theta_E^*) \quad (7)$$

where θ_E^* is the optimal parameter of encoder E . The encoder parameter θ_E is optimized by Adam (Section III):

$$\theta_E = \theta_E - \eta \cdot \text{Adam}(\nabla_{\theta_E} \mathcal{L}_{rec}(x; \theta_E, \theta_G^*); \beta_1, \beta_2). \quad (8)$$

where the updating term $\text{Adam}(\nabla_{\theta_E} \mathcal{L}_{rec}(x; \theta_E, \theta_G^*); \beta_1, \beta_2)$ is determined by the gradient of the loss function \mathcal{L}_{rec} with respect to θ_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_{epoch} [33].

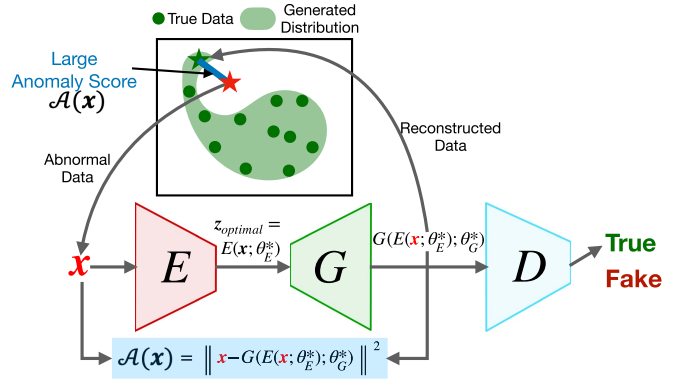


Fig. 1. Overall structure of the anomaly detector.

Note that the Adam is chosen in the GAN and AE because of its competitive performance in the data reconstruction experiments presented in [36] with respect to other possible gradient-based optimization methods (e.g. Stochastic Gradient Descent (SGD), AdaGrad, RMSProp).

The anomaly score function of pattern x is:

$$\mathcal{A}(x) = \|x - \hat{x}\|^2 = \|x - G(E(x; \theta_E^*); \theta_G^*)\|^2 \quad (9)$$

If $\mathcal{A}(x)$ is larger than a threshold value, $A_{\text{threshold}}$, set considering normal condition data: $A_{\text{threshold}} = \max_{x \in \mathcal{X}_{nor}} \mathcal{A}(x)$, then x is anomalous, otherwise, x is normal. The rationale behind the use of a threshold for detecting anomalies is that the distribution of abnormal condition data is expected to be significantly different from that of normal condition data [15]. If the generator were optimal, the optimal latent variable query z_{optimal} corresponding to a pattern in normal condition data, which provides zero reconstruction error, could be identified. In contrast, if x is a pattern collected in abnormal conditions, we would expect a large reconstruction error, because the generator which is trained to reproduce only normal condition data, cannot cover the region of the space corresponding to anomaly data. However, the optimization of z_{optimal} may suffer from large computational burden as one needs to perform the optimization task for each x [37]. Therefore, inspired by [37], we propose to use an auxiliary encoder E to replace the optimization task for efficiently querying the latent variable [37]. Figure 1 shows the overall structure of the developed anomaly detection model.

B. AE-GAN Hyper-Parameter Optimization

Although $JSD(p_G \| p_{\mathcal{X}_{nor}})$ can be used as an actual objective to optimize the GAN architecture, its true value cannot be obtained during GAN training [28]. The generator G provides in output the probability $D(x)$ that x belongs to the distribution of the normal condition data. The label $y(x)$ associated to x is “1” if $x \sim p_{\mathcal{X}_{nor}}(x)$ and “0” if $x \sim p_G(x)$ is 0, then D is trained to minimize the cross entropy loss:

$$\begin{aligned} \mathcal{L}_{BC} &= -\mathbb{E}_{x \sim p_{\mathcal{X}_{nor}}} y(x) \log D(x) + (1 - y(x)) \log (1 - D(x)) \\ &\quad - \mathbb{E}_{x \sim p_G} y(x) \log D(x) + (1 - y(x)) \log (1 - D(x)) \\ &= -[\mathbb{E}_{x \sim p_{\mathcal{X}_{nor}}} \log (D(x)) + \mathbb{E}_{z \sim p_z} \log (1 - D(G(z)))] \end{aligned} \quad (10)$$

Notice that $-\mathcal{L}_{BC}$ is equivalent to the GAN loss $\mathcal{F}(\theta_D, \theta_G)$. According to [28], equation (10) can be written as,

$$\mathcal{L}_{BC} = - \int_{\mathbf{x}} [p_{\mathcal{X}_{nor}}(\mathbf{x}) \log(D(\mathbf{x})) + p_G(\mathbf{x}) \log(1 - D(\mathbf{x}))] d\mathbf{x} \quad (11)$$

For a given generator G , the identification of the optimal discriminator $D^*(\cdot)$ that minimizes \mathcal{L}_{BC} is equivalent to solve the differential equation:

$$\frac{\partial}{\partial D} [p_{\mathcal{X}_{nor}}(\mathbf{x}) \log(D(\mathbf{x})) + p_G(\mathbf{x}) \log(1 - D(\mathbf{x}))] = 0 \quad (12)$$

whose solution is:

$$D^*(\mathbf{x}) = \frac{p_{\mathcal{X}_{nor}}(\mathbf{x})}{p_G(\mathbf{x}) + p_{\mathcal{X}_{nor}}(\mathbf{x})} \quad (13)$$

Then, the JS divergence between p_G and $p_{\mathcal{X}_{nor}}$ is:

$$\begin{aligned} JSD(p_G \| p_{\mathcal{X}_{nor}}) &= \frac{1}{2} KL(p_G \| \frac{p_G + p_{\mathcal{X}_{nor}}}{2}) + \frac{1}{2} KL(p_{\mathcal{X}_{nor}} \| \frac{p_G + p_{\mathcal{X}_{nor}}}{2}) \\ &= \log 2 + \frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_G} \log \left(1 - \frac{p_{\mathcal{X}_{nor}}(\mathbf{x})}{p_G(\mathbf{x}) + p_{\mathcal{X}_{nor}}(\mathbf{x})} \right) \\ &\quad + \frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\mathcal{X}_{nor}}} \log \frac{p_{\mathcal{X}_{nor}}(\mathbf{x})}{p_G(\mathbf{x}) + p_{\mathcal{X}_{nor}}(\mathbf{x})} \end{aligned} \quad (14)$$

By combining equations (13) and (14), we obtain:

$$\begin{aligned} JSD(p_G \| p_{\mathcal{X}_{nor}}) &= \log 2 + \frac{1}{2} [\mathbb{E}_{\mathbf{x} \sim p_G} \log(1 - D^*(\mathbf{x})) \\ &\quad + \mathbb{E}_{\mathbf{x} \sim p_{\mathcal{X}_{nor}}} \log D^*(\mathbf{x})] = \log 2 \\ &\quad + \frac{1}{2} [\mathbb{E}_{\mathbf{z} \sim p_z} \log(1 - D^*(G(\mathbf{z}))) \\ &\quad + \mathbb{E}_{\mathbf{x} \sim p_{\mathcal{X}_{nor}}} \log D^*(\mathbf{x})] = \log 2 + \frac{1}{2} \mathcal{F}(\theta_D^*, \theta_G) \end{aligned} \quad (15)$$

where $\mathcal{F}(\theta_D^*, \theta_G)$ is the GAN loss function of generator G at iteration k . Considering the minimal $\mathcal{F}(\theta_D^*, \theta_G)$, one obtains that $\mathcal{F}(\theta_D^*, \theta_G) \geq \mathcal{F}(\theta_D^k, \theta_G)$ from which is possible to derive $\log 2 + \frac{1}{2} \mathcal{F}(\theta_D^*, \theta_G) \geq \log 2 + \frac{1}{2} \mathcal{F}(\theta_D^k, \theta_G)$, and, therefore:

$$\log 2 + \frac{1}{2} \mathcal{F}(\theta_D^k, \theta_G) = JSD_{LB}(p_G \| p_{\mathcal{X}_{nor}}) \quad (16)$$

In this work, we use JSD_{LB} to monitor the convergence of the GAN training process. In particular, the JS Divergence (JSD) is bounded in the range $[0, \ln 2]$ and when JSD_{LB} becomes close to 0 a good generator G that is able to reproducing the ‘true’ distribution is obtained. Thus, GAN hyperparameters can be further optimized and performance of generator can be quantified without the use of abnormal patterns.

The GAN hyper-parameters include the number of hidden neurons, the number of hidden layers, the size of latent space in generator, N_z , the iteration steps of discriminator per each iteration of generator, k , and the number of epochs, N_{epoch} . 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

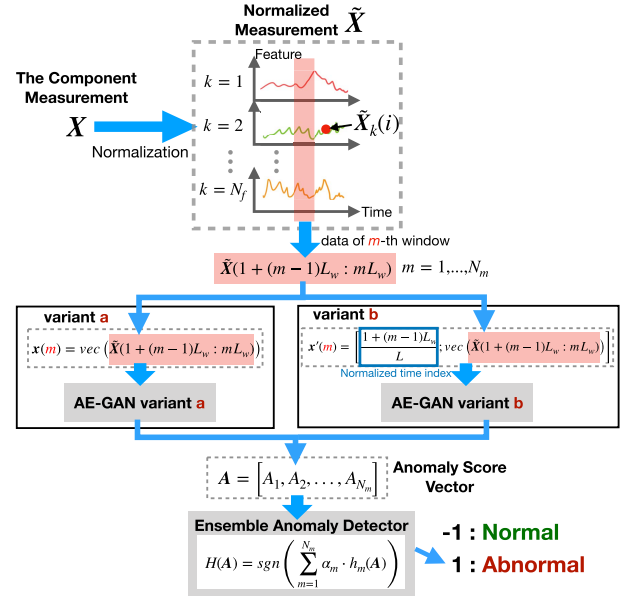


Fig. 2. The flowchart of variants (a) and (b).

perceptrons with the same number of hidden layers, and each hidden layer has the same number of hidden neurons.

C. Ensembled Anomaly Detector by AdaBoost Algorithm

The ensembled anomaly detection method based on AE-GAN is shown in Figure 2. The two main challenges encountered in the real industrial applications are: 1) the densities of data distributions are not smooth and 2) high dimensionality.

With respect to 1), we have found that training GAN on distributions whose densities are not smooth prevents $JSD_{LB}(p_G \| p_{\mathcal{X}_{nor}})$ from converging (Figure 8). To tackle this challenge, we add a normally distributed adaptive noise $\epsilon_k(i)$ to the k -th feature at the i -th time stamp of the r -th healthy components, $\mathbf{X}_k^r(i)$, then we derive:

$$\mathbf{X}_k^{r'}(i) = \mathbf{X}_k^r(i) + \epsilon_k(i) \quad (17)$$

with $\epsilon_k(i) \sim \mathcal{N}(0, \sigma_k(i)^2)$, $k = 1, \dots, N_f$, $i = 1, \dots, L$ and $\sigma_k(i) = \gamma \cdot \text{std}(\{\mathbf{X}_k^r(i)\}_{r=1, \dots, N_{nor}}) + \delta$, where the standard deviation $\sigma_k(i)$ is a variable that changes according to the standard deviation of $\{\mathbf{X}_k^r(i)\}_{r=1, \dots, N_{nor}}$, $\gamma \in (0, +\infty)$ 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 $r = 1, \dots, N_{nor}$.

The rationale of adding an adaptive noise to the data is that if the probability distributions p_G and $p_{\mathcal{X}_{nor}}$ are disjoint manifolds, then the optimal discriminator $D(\mathbf{x}; \theta_D^*(\theta_G)) = 1$ for any true data $\mathbf{x} \in \mathcal{X}_{nor}$, and is 0 for any generated data $G(\mathbf{z})$. Therefore, GAN loss $\mathcal{F}(\theta_D^*(\theta_G), \theta_G)$ will be zero [38], and, as a consequence, the value of $JSD(p_G \| p_{\mathcal{X}_{nor}})$ is equal to $\ln(2) \approx 0.69$. By adding an adaptive noise $\mathcal{N}(0, \sigma^2)$ to the data distribution $p_{\mathcal{X}_{nor}}$, we obtain that p_G and $p_{\mathcal{X}_{nor}}$ are not disjoint and the gradient of $JSD(p_G \| p_{\mathcal{X}_{nor}})$ over θ_G does not vanish during the training of GAN.

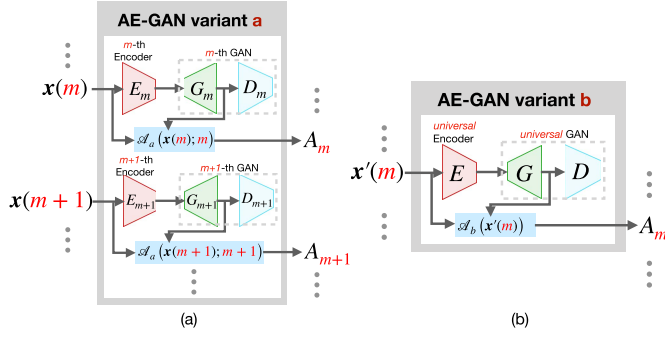


Fig. 3. The AE-GAN anomaly detector with variant (a) and (b).

With respect to 2), many industrial application are characterized by long-term multivariate time series with a large number of features N_f and long sequences of length L , which makes the data dimensionality N_x much larger than the number of the patterns. As a consequence, computation complexity becomes infeasible. To deal with this problem, we propose to use non-overlapped sliding time windows which allows splitting the multivariate time series and treating 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 universal AE-GAN for all time windows. Before the training of AE-GANs and testing for anomaly detection, the training and test data are linearly normalized in the range $[-1, 1]$ according to [28]. In the training phase, the adaptive noise is added to the data, whereas, in the test phase, the adaptive noise is not added to the data.

Variant a). Let $\tilde{X}_k(i)$ be the normalized value of the k -th feature at the i -th time stamp, L_W the size of the time window and $N_m = \text{ceil}(\frac{L}{L_W})$ the number of time windows, the vector $\mathbf{x}(m)$ is defined as $\mathbf{x}(m) = \text{vec}(\tilde{X}(1 + (m-1)L_W : mL_W))$ 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 :

$$\mathcal{X}_{nor,m} = \{\mathbf{x}^1(m), \dots, \mathbf{x}^r(m), \dots, \mathbf{x}^{N_{nor}}(m)\} \quad (18)$$

where $\mathbf{x}^r(m)$ denotes data in the m -th time window collected from healthy components $r = 1, \dots, N_{nor}$. The proposed AE-GAN is trained on the data set $\mathcal{X}_{nor,m}$ for each m -th time window by applying Equations (3)(5)(6); then, the optimal generator $G(\mathbf{z}; \theta_G^*)$ and encoder $E(\mathbf{x}; \theta_E^*)$ for the m -th time window can be obtained. Finally, the anomaly score function of the m -th time window is:

$$\mathcal{A}_a(\mathbf{x}(m); m) = \|\mathbf{x}(m) - G(E(\mathbf{x}(m); \theta_E^*); \theta_G^*)\|^2 \quad (19)$$

The AE-GAN anomaly detector with *variant a)* is illustrated in Figure 3a, where A_m and A_{m+1} denote the anomaly score of the m -th and $m+1$ -th windows, respectively.

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:

$$\mathbf{x}'(m) = \left[\frac{1 + (m-1)L_W}{L}; \text{vec}(\tilde{X}(1 + (m-1)L_W : mL_W)) \right] \quad (20)$$

where symbol ‘;’ represents the vertical concatenation. Note that the dimension of column vector $\mathbf{x}'(m)$ is $N_f \cdot L_W + 1$. This variant constructs the matrix \mathcal{X}'_{nor} of size $N_{nor} \times N_m$ containing all the normal condition patterns, independently from their time window whose generic element (r, m) is $\mathbf{x}'^r(m)$, i.e. the data at m -th window from healthy components $r = 1, \dots, N_{nor}$. The proposed AE-GAN is trained on the data distribution \mathcal{X}'_{nor} by applying Equations (3)(5)(6). Then, the universal optimal generator $G(\mathbf{z}; \theta_G^*)$ and encoder $E(\mathbf{x}; \theta_E^*)$ for all time windows can be obtained and, finally, the universal anomaly score function $\mathcal{A}_b(\mathbf{x}'(m))$ is:

$$\mathcal{A}_b(\mathbf{x}'(m)) = \|\mathbf{x}'(m) - G(E(\mathbf{x}'(m); \theta_E^*); \theta_G^*)\|^2 \quad (21)$$

The training of the AE-GAN anomaly detector with *variant b)* is illustrated in Figure 3b, where A_m denotes the anomaly score of the m -th window.

The details of the training of the anomaly detectors variants *a)* and *b)* are reported in *Algorithm 1*. Notice that the order in which the models are updated are explicitly presented in the Algorithms 1. For *Variant a)*, Lines 7A-10A correspond to the GAN model update, i.e., optimization of GAN objective $\mathcal{F}(\theta_D, \theta_G)$; Lines 11A-12A correspond to the AE model update, i.e. optimization of AE objective $\mathcal{L}_{rec}(\mathbf{x}; \theta_E, \theta_G)$. Similarly, for *variant b)*, Lines 6B-9B correspond to the GAN model update and Lines 10B-11B correspond to the AE model update.

In this work, we further extend the proposed AE-GAN to online detection by adapting the AdaBoost algorithm to the one-class classification. The objective is to develop an ensemble of anomaly detectors producing anomaly scores in the different time windows. 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 [13]. The adapted AdaBoost algorithm is showed in *Algorithm 2*, 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

Algorithm 1 Training AE-GAN Anomaly Detectors: variant a) and b)

Common Part:**Input:** Normal condition data $\{X^r\}_{r=1,\dots,N_{nor}}$.**Initialize:** Scaling factor γ and bias term δ for adaptive noise, time window size L_W .

```

1 Add adaptive noise and obtain  $X^{rr}$  by using Equations (17)
  for  $r = 1, \dots, N_{nor}$ 
2 Normalize  $X^{rr}$  into  $\tilde{X}^r$  in the range  $[-1, 1]$  for
   $r = 1, \dots, N_{nor}$ 
Variant a):
/* Train AE-GAN for each  $m$ -th time window */
3A for  $m = 1, \dots, N_m$  do
4A   Obtain  $x^r(m)$  for  $r = 1, \dots, N_{nor}$ .
5A   Construct set  $\mathcal{X}_{nor,m}$  by using Equation (18) and assign
    to  $\mathcal{X}_{nor}$ .
6A   Initialize  $\theta_D, \theta_G, \theta_E$  by Xavier Uniform initialization
    method [39].
/* Optimize GAN objective */
7A   for  $epoch = 1, \dots, N_{epoch}$  do
8A     for  $k = 1, \dots, K$  do
9A        $\theta_D^{(k)} = \theta_D^{(k-1)} + \eta \cdot \text{Adam}(\nabla_{\theta_D} \mathcal{F}(\theta_D^{(k-1)}, \theta_G); \beta_1, \beta_2)$ 
10A       $\theta_G = \theta_G - \eta \cdot \text{Adam}(\nabla_{\theta_G} \mathcal{F}(\theta_D^{(k)}, \theta_G); \beta_1, \beta_2)$ 
/* Optimize AE objective */
11A   for  $epoch = 1, \dots, N_{epoch}$  do
12A      $\theta_E = \theta_E - \eta \cdot \text{Adam}(\nabla_{\theta_E} \mathcal{L}_{rec}(x; \theta_E, \theta_G^*); \beta_1, \beta_2)$ 
13A   Assign  $\theta_G^* \leftarrow$  optimized  $\theta_G$ ,  $\theta_E^* \leftarrow$  optimized  $\theta_E$  and
    obtain  $\mathcal{A}_a(x(m); m)$  in Equation (19).

```

Variant b):

```

/* Train a universal AE-GAN for all the time windows */
3B Obtain  $x^{rr}(m)$  by using Equation (20) for
   $m = 1, \dots, N_m$ ,  $r = 1, \dots, N_{nor}$ .
4B Construct set  $\mathcal{X}_{nor}$  and assign to  $\mathcal{X}_{nor}$ .
5B Initialize  $\theta_D, \theta_G, \theta_E$  by Xavier Uniform initialization
    method [39].
/* Optimize GAN objective */
6B for  $epoch = 1, \dots, N_{epoch}$  do
7B   for  $k = 1, \dots, K$  do
8B      $\theta_D^{(k)} = \theta_D^{(k-1)} + \eta \cdot \text{Adam}(\nabla_{\theta_D} \mathcal{F}(\theta_D^{(k-1)}, \theta_G); \beta_1, \beta_2)$ 
9B      $\theta_G = \theta_G - \eta \cdot \text{Adam}(\nabla_{\theta_G} \mathcal{F}(\theta_D^{(k)}, \theta_G); \beta_1, \beta_2)$ 
/* Optimize AE objective */
10B for  $epoch = 1, \dots, N_{epoch}$  do
11B    $\theta_E = \theta_E - \eta \cdot \text{Adam}(\nabla_{\theta_E} \mathcal{L}_{rec}(x; \theta_E, \theta_G^*); \beta_1, \beta_2)$ 
12B Assign  $\theta_G^* \leftarrow$  optimized  $\theta_G$ ,  $\theta_E^* \leftarrow$  optimized  $\theta_E$  and obtain
     $\mathcal{A}_b(x'(m))$  in Equation (21).

```

term to keep stability in the AdaBoost Ensemble learning, in particular, avoiding $h_m(A) = 0$.

Considering a AE-GAN network with N_{neu} hidden neurons, the number of weight connections between the input x and the first layer of Encoder E is $N_{neu} \times N_f \times L_w$, and the number of weight connections between the last layer of Generator G and the output $G(z)$ is $N_f \times L_w \times N_{neu}$. As a consequence, the computational complexity of a single AE-GAN is $2\mathcal{O}(N_f \cdot L_w \cdot N_{neu}) + \mathcal{O}(C)$, where C is a constant representing the computational complexity associated to the input and output weight connections. When all the N_m time

Algorithm 2 AdaBoost Ensemble Learning for Anomaly Detection

Input: Anomaly score validation set, $\mathcal{V} = \{A^v\}_{v=1,\dots,N_v}$, weak classifier $h : A \rightarrow \{-1, 1\}$, percentile c .**Output:** Ensembled classifier

$$H(A) = \text{sgn}\left(\sum_{m=1}^{N_m} \alpha_m \cdot h_m(A)\right)$$

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 errorrate $\epsilon_m, m=1, \dots, N_m$ set as $\frac{1}{2}$.

```

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

```

windows are considered, the total computational complexity is $2\mathcal{O}(N_m \cdot N_f \cdot L_w \cdot N_{neu}) + \mathcal{O}(N_m \cdot C)$. Note that $L_w \cdot N_m$ is equal to L and is, therefore, independent from the length of the time window L_W . The second term linearly increases with the number of windows N_m , but the network computational complexity $\mathcal{O}(C)$ is expected to be small when the time windows become small and N_m large.

V. CASE STUDY

A. Protocol and Setting

Performance Metrics. The metrics of accuracy, precision, recall, balanced F-score and the Area Under the receiver operating characteristic Curve (AUC) are used to evaluate the performance of the proposed method with respect to the anomaly detection task. Specifically, the number of normal condition patterns correctly classified is indicated as true positive (tp), the number of abnormal condition patterns correctly classified as true negative (tn), the number of abnormal condition patterns misclassified as normal as false positive (fp), the number of normal condition patterns misclassified as abnormal conditions as false negative (fn). *Accuracy* is defined as the fraction of correctly classified patterns among all patterns, $Accuracy = \frac{tp+tn}{tp+tn+fp+fn}$. *Precision* is the fraction of normal condition patterns correctly classified among the patterns that are classified as normal, $Precision = \frac{tp}{tp+fp}$. *Recall* is the fraction of normal condition patterns correctly classified among the true normal condition patterns, $Recall = \frac{tp}{tp+fn}$. *balanced F-score* F_1 is the harmonic mean of *Precision*

and Recall, $F_1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$. Receiver Operating Characteristic (ROC) curve is a curve obtained by plotting the recall against the false positive rate (FPR), $FPR = \frac{fp}{fp+tn}$. Area Under the ROC Curve (AUC) is calculated by using an average of a number of trapezoidal approximations [40].

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. The proposed methods are compared with the following state-of-the-art anomaly detection methods: OC-SVM [41], AAKR [17], GMM [42], AE [43], AnoGAN [35], VAE- β [44] and Deep-SVDD [45]. OC-SVM [41] estimates the support of a high-dimensional distribution, and a hypersphere is defined to distinguish the normal and abnormal, the score function is generated by measuring the “distance” between the margin of the hypersphere of normal condition data distribution. The larger the score, the higher probability the pattern is abnormal. AAKR [17] reconstructs a test pattern as a weighted sum of normal condition patterns. The weights are obtained by applying a radial basis function between which measures the similarities between the test pattern and the normal condition patterns. Then, the anomaly score is defined by: $\mathcal{A}_{AAKR}(\mathbf{x}) = \|\mathbf{x} - \hat{\mathbf{x}}_{AAKR}\|^2$ with $\hat{\mathbf{x}}_{AAKR}$ indicating the AAKR reconstruction of \mathbf{x} . GMM [42] is used to model distribution of the normal condition patterns. Then, the anomaly score of a test pattern is defined as $\mathcal{A}_{GMM}(\mathbf{x}) = -p(\mathbf{x}; \theta_{GMM})$ where the likelihood of a test pattern is used. $p(\mathbf{x}; \theta_{GMM}) = \sum_{i=1}^k \phi_i \mathcal{N}(\mathbf{x}; \mu_i, \Sigma_i)$ is the likelihood function, with the GMM parameters ϕ_i indicating the component weight, μ_i indicating the mean and Σ_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 [43] has been illustrated in Section III-B and it is trained on normal condition 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 [13]. This work defines $\mathcal{A}_{AE}(\mathbf{x}) = \|\mathbf{x} - \hat{\mathbf{x}}\|^2$ as anomaly score of AE. It should be noted that the encoder and generators of AE and AE-GAN have the same architecture. Also, the setting of the Adam coefficients β_1, β_2 , batch size L_{batch} , learning rate η , number of epoch N_{epoch} , is the same for AE and AE-GAN. AnoGAN for anomaly detection [35] associates an optimal latent variable $\mathbf{z}_{optimal}$ to a pattern \mathbf{x} by minimizing the reconstruction error. Then, the anomaly score is defined as $\mathcal{A}_{AnoGAN}(\mathbf{x}) = \|\mathbf{x} - G(\mathbf{z}_{optimal})\|^2$. It should be noted that AnoGAN shares the GAN module with the proposed AE-GAN. Therefore, the parameter settings used for optimizing the latent variable $\mathbf{z}_{optimal}$, Adam coefficients β_1, β_2 , learning rate η , number of epochs N_{epoch} , are the same as those for AE-GAN training. VAE- β differs from AE since it encodes the input data into a multivariate latent distribution, which is constrained by the Kullback-Leibler divergence between the parametric posterior and the true posterior [44]. The use of a VAE- β for anomaly detection requires the training of the VAE using normal condition data with the objective of

minimizing the sum of the reconstruction error $\|\mathbf{x} - \hat{\mathbf{x}}\|^2$ and a term equal to $\beta \cdot KL(p_G \| p_{\mathcal{X}_{nor}})$. Then, the anomaly score of a test pattern is defined by $\|\mathbf{x} - \hat{\mathbf{x}}\|^2$. The network architectures (number of neurons and model layers) and hyperparameters setting (batch size L_{batch} , epoch number N_{epoch} , learning rate η and Adam coefficients β_1, β_2) of the developed VAE- β are set equal to that of AE and AE-GAN. Deep-SVDD has been introduced in [45] with the objective of learning a neural network transformation $\phi(\cdot; \mathcal{W})$ from an input space to an output space properly defined in such a way that normal patterns fall within a hypersphere of minimum volume. Then, the anomaly score of a test pattern is defined as the distance between its projection in the output space and the hypersphere center. The hyperparameters of the developed Deep-SVDD have been set equal to those of the encoders of AE and VAE- β .

Section V-B verifies the effectiveness of the proposed AE-GAN anomaly detector with respect to the comparison methods OC-SVM, AAKR, GMM, AE, AnoGAN, VAE- β and Deep-SVDD on a synthetic dataset. Then, in Section V-C, we consider variants a) and b) of the proposed AdaBoost ensemble of AE-GANs and we compare their performances to OC-SVM, AAKR, GMM, AE, VAE- β , Deep-SVDD and six Adaboost ensembles based on Algorithm 2 and the use of OC-SVM, AAKR, GMM, AE, VAE- β and Deep-SVDD as base classifiers respectively, in which each ensemble approach uses the adapted AdaBoost algorithm to learn an ensemble anomaly detector for each time window.

B. Synthetic Case

Three synthetic case studies, which will be referred to as Cone, Two Spheres and Bowl, are considered to verify the anomaly detection performance of the proposed AE-GAN anomaly detector on data that mimic the complexity of real industrial applications. In all three case studies the training set is formed by 3000 normal condition patterns and the test set of 643 normal condition patterns and 642 abnormal condition patterns. Cone patterns are sampled from a 3-D Gaussian distribution with mean [4, 0, 0] and variance $diag(1, 1, 1)$. Then, only the patterns inside a cone of bottom radius 2 and height 3 are kept. Two Spheres patterns are sampled from two 3-D Gaussian distributions with mean $[\pm 4, 0, 0]$ variance $diag(1, 1, 1)$. Then, only patterns inside two 3-D spheres, whose centers are located at $[\pm 4, 0, 0]$ and the radius is 2, are kept. Bowl patterns are uniformly sampled from a hemisphere with radius 6 and center point [0, 0, 0].

In all the three case studies, abnormal condition data are uniformly sampled from a Uniform distribution in the hypercube with a range of $[-10, 10]$ for each dimension. Then, the patterns within the normal condition region are deleted.

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 a discriminator and a generator. According to [38] and [46], during the model training, for each iteration of the generator is followed by 5 iterations of the discriminator. The AE-GAN model architecture is

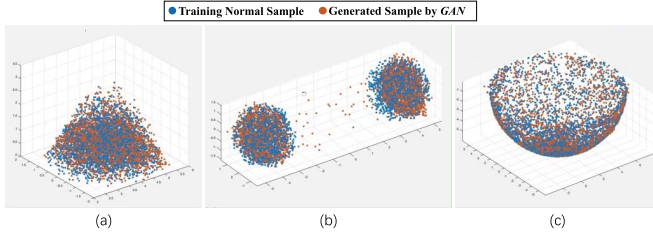


Fig. 4. GAN reconstruction of the normal condition data in the three case studies. (a) Cone, (b) Two Sphere, and (c) Bowl.

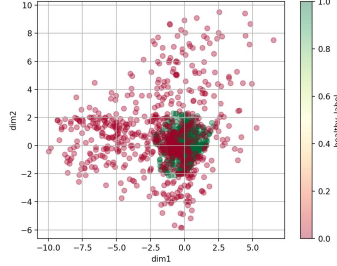


Fig. 5. AE-GAN Latent space visualization for test patterns of Manifold dataset, green denotes normal and red denotes abnormal.

characterized by an encoder, a generator, and a discriminator. They are all made by two hidden layers of 50 neurons. The Latent Space Layer acts as output layer of the encoder and input layer of the generator, and the number of neurons in the Latent Space Layer is 2 for the Bowl, and 3 for the Cone and Two Sphere. Also, both Encoder and Generator activate their hidden layers by Rectified Linear Unit (*ReLU*) [47], whereas Discriminator uses Leaky *ReLU* [48] as activation function and the leaky rate is set as 0.2. The choice of using 2 neurons for the Bowl case study is motivated by the observation that 2 coordinates are enough to describe the normal condition data, whereas 3 coordinates are needed in the other case studies. The Adam optimizer is used to train the GAN (Generator and Discriminator modules) and AE (Encoder and Generator), setting 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 $N_{epoch} = 1000$. The threshold to detect abnormal conditions is set equal to the maximum of the anomaly score among the patterns of the train set for *OC-SVM*, *GMM*, *AAKR*, *AE*, *AnoGAN*.

Figure 4 shows that the GAN provides a satisfactory reconstruction of the distribution of the normal condition patterns in the three case studies.

Figure 5 shows the latent space followed by AE-GAN when fed by Manifold patterns. Note that the latent variables corresponding to normal patterns are located nearly to the core of the Normal distribution, whereas the abnormal condition data are widely spread. Since Generator is developed to map patterns of the latent space into normal condition data, the distributions between normal and abnormal are disjoint with the majority of the abnormal conditional patterns that can be distinguished.

TABLE I
DETECTION PERFORMANCE ON THE SYNTHETIC CASE STUDIES

Dataset	Metric	<i>OC-SVM</i>	<i>AAKR</i>	<i>GMM</i>	<i>AE</i>	<i>AnoGAN</i>	<i>VAE-beta</i>	<i>Deep-SVDD</i>	<i>AE-GAN</i> (proposed)
Cone	Accuracy	0.9938	0.9564	0.9922	0.9813	0.9977	0.9953	0.9977	1
	Precision	0.9877	1	0.9847	0.9905	0.9954	1	0.9969	1
	Recall	1	0.9129	1	0.9720	1	0.9907	0.9984	1
	F-score	0.9938	0.9545	0.9923	0.9812	0.9977	0.9953	0.9977	1
Two Gaussian Ball	Accuracy	0.9969	0.8677	0.9953	0.9907	0.9564	0.9930	0.9852	0.9969
	Precision	0.9938	1	0.9908	0.9937	0.9742	0.9953	0.9921	0.9953
	Recall	1	0.7356	1	0.9876	0.9378	0.9907	0.9782	0.9984
	F-score	0.9968	0.8477	0.9954	0.9906	0.9556	0.9930	0.9851	0.9969
Bowl Manifold	Accuracy	0.9471	0.9237	0.9012	0.9626	0.8179	0.9043	0.8996	0.9728
	Precision	0.909	1	0.8395	0.9514	0.7353	0.9981	0.9981	0.9648
	Recall	0.9938	0.8476	0.9922	0.9751	0.9938	0.8099	0.8006	0.9813
	F-score	0.9495	0.9175	0.9095	0.9631	0.8452	0.8942	0.8885	0.9730

Table I reports that the AE-GAN provides the most satisfactory accuracy and F-score in all three case studies. In particular, it achieves zero false and missed alarms on the Cone. In the Two Sphere and Bowl case study, although the proposed AE-GAN method cannot achieve at the same time the smallest false and missed alarm rates, it provides the most satisfactory accuracy, and F-score precision and recall scores are always among the best three. It is, therefore, possible to conclude that the proposed AE-GAN allows obtaining the best trade-off between false and missed alarms, which is the key ability in risk-critical applications such as anomaly detection. Notice that the state-of-the-art *DL* methods (*VAE- β* and *Deep-SVDD*) tend to remarkably underperform with respect to the proposed AE-GAN when applied to the Bowl due to their tendency of being trapped in local minima when applied to highly curved surfaces.

C. 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 (100 used for training, 20 for validating and 18 for testing), and 22 components on fault type A and 33 components on fault type B. 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 6 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.

This section investigates the effect on the performance of the size of the time windows, of the adaptive noise and of using the method in variant *a*) and *b*).

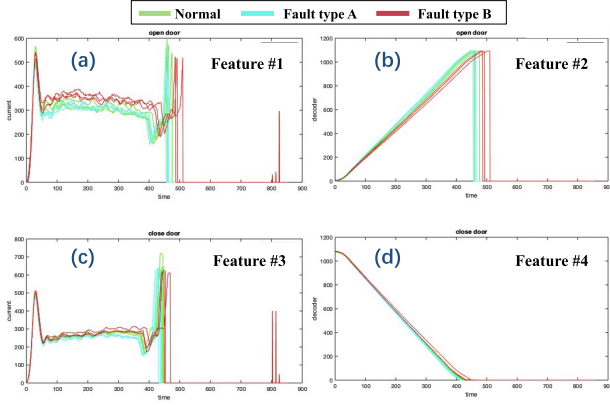
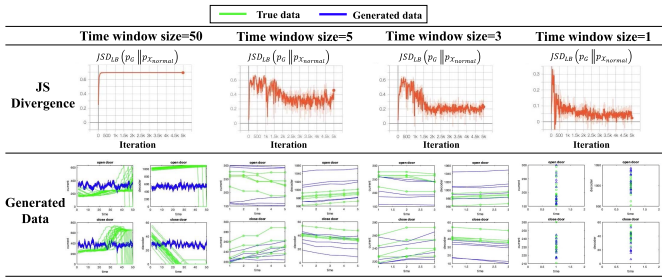


Fig. 6. Examples of signal evaluations in the real industrial dataset.

Fig. 7. J -S divergence and generated data as a function of the time window size.

Effect of the window size. This paragraph experiments the effect of different window sizes on the convergence of $JSD_{LB}(p_G \| p_{\mathcal{X}_{nor}})$. 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 (18)); 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 7). Notice that when the window size is equal to 50, the input dimension becomes 200 which is large in comparison to the number of pattern 100. Figure 7 shows that it causes mode collapse of GAN, which is revealed by the collapsed generated data that is nothing but randomness and the non convergence of JS divergence. On the other side, as the window size gets smaller, $JSD_{LB}(p_G \| p_{\mathcal{X}_{nor}})$ gradually converges close to 0.

Effect of the adaptive noise. This paragraph experiments the effect of adaptive noise on the convergence of $JSD_{LB}(p_G \| p_{\mathcal{X}_{nor}})$. In the experiment, the normal components signal at time 460 is used to train the GAN (see Figure 8b) and the parameters of the adaptive noise that is added to the data are set as $\gamma = 0.02$, $\delta = 0.001$ (see Equations (17)), and the size of latent space in GAN is set to 4. During the GAN training process, the $JSD_{LB}(p_G \| p_{\mathcal{X}_{nor}})$ at each iteration of the generator optimization is recorded (see Figure 8a). The experiment shows that the original data distribution is non-smooth and is the cause of non convergence of GAN. Also, it verifies that after adding adaptive noise to the data distribution with non-smooth density, the original

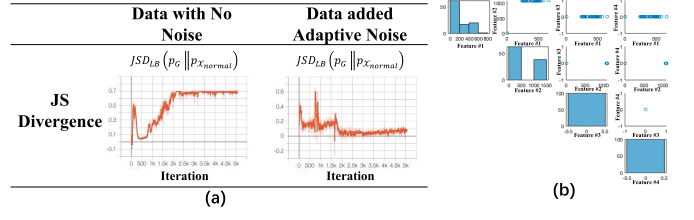
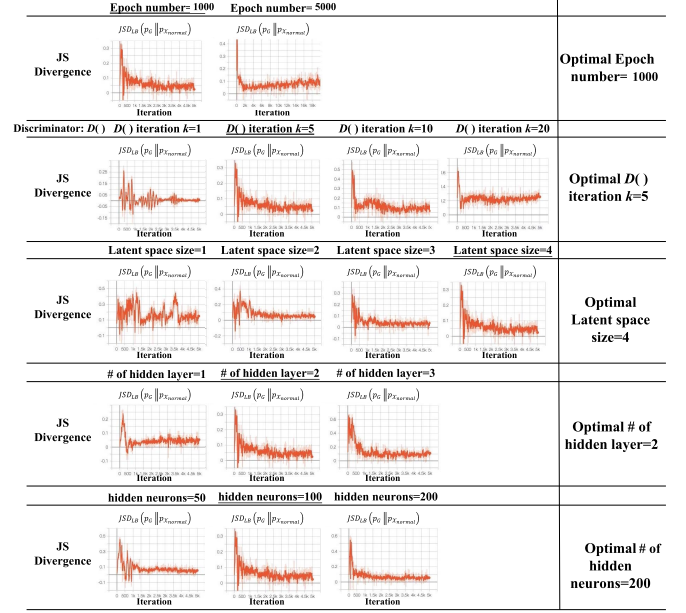
Fig. 8. (a) The effect of adaptive noise on the convergence of J -S divergence, (b) the normal component data whose distribution has non-smooth density at time 460.

Fig. 9. Results of the AE-GAN hyper-parameters optimization.

distribution is converted to a smooth distribution, so that $JSD_{LB}(p_G \| p_{\mathcal{X}_{nor}})$ converges to 0 according to Section IV-C.

Effect of using variant a) and b). The methods reported in Table II and Figure 12 shows that variant a) outperforms variant b). This is due to the fact that variant b), which develops a single universal AE-GAN for all windows, is less specific than variant a), which develops a dedicated AE-GAN for each time window. Notice, however, that variant b) is more computationally efficient than a).

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

Figure 9 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 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

TABLE II
F-SCORE ON THE REAL INDUSTRIAL DATASET

Percentile c	Compared Methods												Ablation Study		Proposed Method	
	<i>OC-SVM</i>	<i>OC-SVM</i>	<i>AAKR</i>	<i>AAKR</i>	<i>GMM</i>	<i>GMM</i>	<i>AE</i>	<i>AE</i>	<i>VAE-β</i>	<i>VAE-β</i>	<i>Deep-SVDD</i>	<i>Deep-SVDD</i>	<i>AE-GAN</i>	<i>AE-GAN</i>	<i>AE-GAN</i>	<i>AE-GAN</i>
		(Ens)		(Ens)		(Ens)		(Ens)		(Ens)		(Ens)	(a-I)	(a-II)	(a)	(b)
100%	0.5952	0.6207	0.4727	0.6452	N/A	0.6333	0.5391	0.6230	0.6972	0.7311	0.6847	0.7311	0.6365	0.7173	0.7312	0.7312
95%	N/A	0.6207	N/A	0.6452	N/A	0.6333	N/A	0.6230	N/A	0.7311	N/A	0.7311	0.6830	0.7226	0.7174	0.7312
90%	N/A	0.5965	N/A	0.6452	N/A	0.6333	N/A	0.6230	N/A	0.7311	N/A	0.7311	0.6758	0.7083	0.7253	0.7312
85%	N/A	0.5965	N/A	0.6557	N/A	0.6333	N/A	0.6230	N/A	0.7311	N/A	0.7311	0.6758	0.6599	0.7191	0.7312
80%	N/A	0.5965	N/A	0.6557	N/A	0.6333	N/A	0.6452	N/A	0.7368	N/A	0.7234	0.6758	0.6599	0.7273	0.7527
75%	N/A	0.5965	N/A	0.6557	N/A	0.6102	N/A	0.5763	N/A	0.7272	N/A	0.7368	0.6758	0.6666	<u>0.7750</u>	0.7692
70%	N/A	0.5283	N/A	0.6555	N/A	0.6332	N/A	0.5574	N/A	0.7254	N/A	0.7234	0.6758	0.6796	0.7692	0.6750
65%	N/A	0.4906	N/A	0.6333	N/A	0.5614	N/A	0.5246	N/A	0.7207	N/A	0.7499	0.6547	0.6728	0.6761	0.5854
60%	N/A	0.5091	N/A	0.6102	N/A	0.6000	N/A	0.5246	N/A	0.7304	N/A	0.7318	0.6258	0.6666	0.6364	0.5500

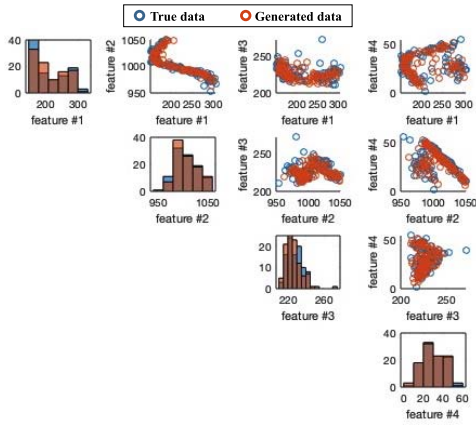


Fig. 10. An example of true data distribution and the generated data distribution produced by the optimal *AE-GAN*. The true data comes from the normal components signal at time 400. The experiment shows that the true data distribution can be nearly perfectly reproduced, which satisfies the basic prerequisite of *GAN*-based anomaly detection methods.

and hidden neurons number. The optimization objective is $JSD_{LB}(p_G \| p_{\mathcal{X}_{nor}})$ (Section IV-B). Observing Figure 9 one can notice that: *i*) a large epoch number, e.g. larger than 1000, can degrade the *JS* divergence; *ii*) too small k makes *GAN* training unstable and large value $k > 5$ degrade the *JS* divergence; *iii*) the larger the latent size, the more stable the *GAN* training; *iv*) the number of hidden layer has less impact on *GAN* training than k and the latent size; *v*) larger hidden neuron number brings better performance. After training of *AE-GAN* with the optimal hyper-parameters, an example of the generator distribution is shown in Figure 10.

This work compares the proposed *AE-GAN* with *OC-SVM*, *AAKR*, *GMM*, *AE*, *VAE-β* and *Deep-SVDD*. *AnoGAN* is not compared because it is very computationally intensive when finding optimal latent variable $z_{optimal}$ w.r.t. each training and test patterns. As for the compared methods, this work uses two strategies. The first is to treat the multivariate time series as the input data pattern and obtain the anomaly score (Section V-A); then, the threshold (in Section IV) is set to the maximum value of the anomaly scores among the training normal patterns.

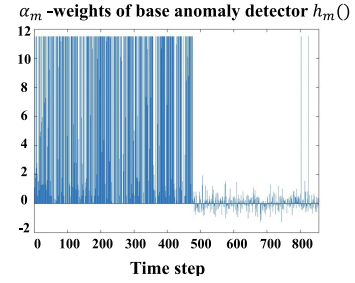


Fig. 11. Example of base anomaly detectors weight α_m of, e.g. *AE-GAN* variant *a*), at each time step. We observe that the weights of the base anomaly detectors suddenly drop at time 500, when, interestingly, the value of the original signal (Figure 6) becomes a constant, which means that the signal after time 500 is irrelevant to component health monitoring.

The second strategy is similar to the proposed ensemble 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 pattern for anomaly detection and obtain the anomaly score (Section V-A); then, it uses the proposed *Algorithm 2* (see Section IV-C) to obtain the ensemble anomaly detection result. The ensemble compared methods are referred to as *OC-SVM (Ens)*, *AAKR (Ens)*, *GMM (Ens)*, *AE (Ens)*, *VAE-β (Ens)* and *Deep-SVDD (Ens)*.

Table II compares the anomaly detection results considering different values of percentile c in *Algorithm 2*. Notice that variants *a*) and *b*) of the proposed *AE-GAN* achieve the best performances for the majority of the considered percentiles. The best F-score (0.7750) is obtained by variant *a*) when the percentile c is set equal to 75%, whereas the best F-score of variant *b*) (0.7527) is obtained when c is 85%. Overall, variant *a*) is better performing than variant *b*), since this latter introduces discrete time into the data space (see Equation (20)), which makes it difficult for the generator, which has a continuous data space, to fit the data space containing discrete times. Notice that *GMM* has not been applied to this dataset given the large dimensionality of the data which makes matrix computation unfeasible and the methods not based on ensemble of classifiers (*OC-SVM*, *AAKR*, *AE*, *VAE-β*,

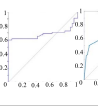
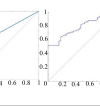
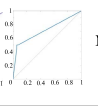
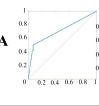
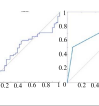
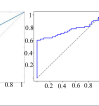
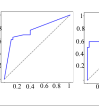
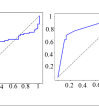
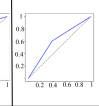
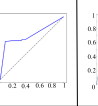
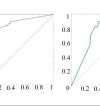
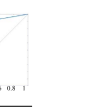

Metric	Compared Methods												Ablation Study		Proposed Method	
	<i>OC-SVM</i>	<i>OC-SVM (Ens)</i>	<i>AAKR</i>	<i>AAKR (Ens)</i>	<i>GMM</i>	<i>GMM (Ens)</i>	<i>AE</i>	<i>AE (Ens)</i>	<i>VAE-β</i>	<i>VAE-β (Ens)</i>	<i>Deep-SVDD</i>	<i>Deep-SVDD (Ens)</i>	<i>AE-GAN (a-I)</i>	<i>AE-GAN (a-II)</i>	<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	0.7517	0.7614	0.6512	0.7942	0.6117	0.7149	0.8289	0.7244

Fig. 12. Real industrial dataset: ROC curve and AUC of the proposed AE-GAN with variants (a) and (b), and of the methods used for the comparison. The x axes report the false positive rates and the y axes the true positive rates. In the ablation study, AE-GAN(a-I) is the experiment applying AE-GAN variant (a) without adding adaptive noise, AE-GAN(a-II) is the experiment applying AE-GAN variant (a) with the mode collapsed generator using window size 50 (see Fig. 7).

Deep-SVDD) which provide in output a single pattern class for each test pattern, and, therefore do not allow computing the percentiles of the output distribution.

By using the proposed improved AdaBoost ensemble learning (Algorithm 2), the F-score and AUC is boosted for nearly all the compared methods Table II. In order to obtain the comprehensive anomaly detection performance, we look at the ROC curve adjusting the percentile c and obtaining the AUC (Figure 12), which shows that the proposed AE-GAN variant (a) outperforms any other compared methods. Additionally, the ablation studies AE-GAN (a-I, a-II) whose results are reported in Figure 12 prove the effectiveness of the proposed strategies: adding adaptive noise has a more significant impact on the performance than optimizing AE-GAN hyperparameters. The interpretation is that the proposed method can address the real industrial data difficulties, including complex, non-smooth and manifold distributions.

To deeply analyze the performance of the proposed method, we construct two test sets from the industrial dataset. Test Set 1 contains 18 normal condition patterns and 22 anomalous patterns of fault class A and Test Set 2 contains the same 18 normal condition patterns and 33 anomalous patterns of only fault class B. The AUC on Test Set 1 is 0.6304 for AE-GAN(a) and 0.5407 for AE-GAN(b), on Test Set 2 is 0.9474 for AE-GAN(a) and 0.8469 for AE-GAN(b), which shows 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 condition data, whereas the data distribution of fault type B is clearly distinguishable from normal condition data.

Advantages: i) Differently from the state-of-the-art methods, the JS divergence approximation introduced in this work allows assessing the performance of the GAN and thus optimize AE-GAN anomaly detector hyperparameters without the use of true anomalous data; ii) The combination of the AE-GAN with the Adaboost algorithm allows automatically filtering the task-related features by assigning different weights to the base anomaly detectors and providing interpretable results; iii) The proposed method is capable of simultaneously addressing the following challenges of real industrial anomaly detection problems: treating long-term time series with complex, manifold and non smooth data distribution.

Disadvantages: i) More hyperparameters to tune during optimization of GAN; ii) Although, the inference time is competitive, the computational effort needed to train the AE-GAN is larger.

VI. 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 high-speed train automatic doors where abnormal condition 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 condition 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) *high 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.

REFERENCES

- [1] A. A. Cook, G. Misirli, and Z. Fan, "Anomaly detection for IoT time-series data: A survey," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6481–6494, Jul. 2020.
- [2] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Comput. Secur.*, vol. 28, nos. 1–2, pp. 18–28, Feb. 2009.
- [3] H. Kim, J. Park, K. Min, and K. Huh, "Anomaly monitoring framework in lane detection with a generative adversarial network," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 3, pp. 1603–1615, Mar. 2020.
- [4] W. Luo et al., "Video anomaly detection with sparse coding inspired deep neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 3, pp. 1070–1084, Mar. 2021.
- [5] A. Belhadi, Y. Djenouri, G. Srivastava, D. Djenouri, A. Cano, and J. C.-W. Lin, "A two-phase anomaly detection model for secure intelligent transportation ride-hailing trajectories," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4496–4506, Jul. 2021.

- [6] Q. Kang, X. Chen, S. Li, and M. Zhou, "A noise-filtered under-sampling scheme for imbalanced classification," *IEEE Trans. Cybern.*, vol. 47, no. 12, pp. 4263–4274, Dec. 2017.
- [7] L. Zheng, G. Liu, C. Yan, C. Jiang, and M. Li, "Improved TrAdaBoost and its application to transaction fraud detection," *IEEE Trans. Computat. Social Syst.*, vol. 7, no. 5, pp. 1304–1316, Oct. 2020.
- [8] Z. Ghafoori, S. M. Erfani, J. C. Bezdek, S. Karunasekera, and C. Leckie, "LN-SNE: Log-normal distributed stochastic neighbor embedding for anomaly detection," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 4, pp. 815–820, Apr. 2020.
- [9] Y. Xiao, H. Wang, L. Zhang, and W. Xu, "Two methods of selecting Gaussian kernel parameters for one-class SVM and their application to fault detection," *Knowl.-Based Syst.*, vol. 59, pp. 75–84, Mar. 2014.
- [10] C. M. Rocco and E. Zio, "A support vector machine integrated system for the classification of operation anomalies in nuclear components and systems," *Rel. Eng. Syst. Saf.*, vol. 92, no. 5, pp. 593–600, May 2007.
- [11] H. Sarmadi and 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," *Mech. Syst. Signal Process.*, vol. 140, Jun. 2020, Art. no. 106495.
- [12] L. Li, R. J. Hansman, R. Palacios, and R. Welsch, "Anomaly detection via a Gaussian mixture model for flight operation and safety monitoring," *Transp. Res. C, Emerg. Technol.*, vol. 64, pp. 45–57, Mar. 2016.
- [13] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Comput.*, vol. 22, pp. 949–961, Jan. 2019.
- [14] Z. Zhang and X. Deng, "Anomaly detection using improved deep SVDD model with data structure preservation," *Pattern Recognit. Lett.*, vol. 148, pp. 1–6, Aug. 2021.
- [15] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, 2009.
- [16] D. Yang, A. Usynin, and J. W. Hines, "Anomaly-based intrusion detection for SCADA systems," in *Proc. 5th Int. Top. Meeting Nucl. Plant Instrum., Control Hum. Mach. Interface Technol.*, 2006, pp. 12–16.
- [17] P. Baraldi, F. Di Maio, P. Turati, and E. Zio, "Robust signal reconstruction for condition monitoring of industrial components via a modified auto associative kernel regression method," *Mech. Syst. Signal Process.*, vols. 60–61, pp. 29–44, Aug. 2015.
- [18] X. Wang, Q. Kang, M. Zhou, L. Pan, and A. Abusorrah, "Multiscale drift detection test to enable fast learning in nonstationary environments," *IEEE Trans. Cybern.*, vol. 51, no. 7, pp. 3483–3495, Jul. 2021.
- [19] L. Zou, Z. Wang, H. Geng, and X. Liu, "Set-membership filtering subject to impulsive measurement outliers: A recursive algorithm," *IEEE/CAA J. Autom. Sinica*, vol. 8, no. 2, pp. 377–388, Feb. 2021.
- [20] Y. Feng, Y. Yuan, and X. Lu, "Learning deep event models for crowd anomaly detection," *Neurocomputing*, vol. 219, pp. 548–556, Jan. 2017.
- [21] Y. Qin, C. Yan, G. Liu, Z. Li, and C. Jiang, "Pairwise Gaussian loss for convolutional neural networks," *IEEE Trans. Ind. Informat.*, vol. 16, no. 10, pp. 6324–6333, Oct. 2020.
- [22] Z. Li, G. Liu, and C. Jiang, "Deep representation learning with full center loss for credit card fraud detection," *IEEE Trans. Computat. Social Syst.*, vol. 7, no. 2, pp. 569–579, Apr. 2020.
- [23] J. Qian, L. Jiang, and Z. Song, "Locally linear back-propagation based contribution for nonlinear process fault diagnosis," *IEEE/CAA J. Autom. Sinica*, vol. 7, no. 3, pp. 764–775, May 2020.
- [24] M. Sabokrou, M. Fathy, G. Zhao, and E. Adeli, "Deep end-to-end one-class classifier," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 2, pp. 675–684, Feb. 2021.
- [25] M. N. Kurt, Y. Yilmaz, and X. Wang, "Real-time nonparametric anomaly detection in high-dimensional settings," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 7, pp. 2463–2479, Jul. 2021.
- [26] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.
- [27] J. Wu, Z. Zhao, C. Sun, R. Yan, and X. Chen, "Fault-attention generative probabilistic adversarial autoencoder for machine anomaly detection," *IEEE Trans. Ind. Informat.*, vol. 16, no. 12, pp. 7479–7488, Dec. 2020.
- [28] Y. Chen, Y. Lv, and F. Wang, "Traffic flow imputation using parallel data and generative adversarial networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 4, pp. 1624–1630, Apr. 2020.
- [29] T. Schlegl, P. Seeböck, S. M. Waldstein, G. Langs, and U. Schmidt-Erfurth, "F-AnoGAN: Fast unsupervised anomaly detection with generative adversarial networks," *Med. Image Anal.*, vol. 54, pp. 30–44, May 2019.
- [30] J. Li, Y. Tao, and T. Cai, "Predicting lung cancers using epidemiological data: A generative-discriminative framework," *IEEE/CAA J. Autom. Sinica*, vol. 8, no. 5, pp. 1067–1078, May 2021.
- [31] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.
- [32] H.-J. Xing and W.-T. Liu, "Robust AdaBoost based ensemble of one-class support vector machines," *Inf. Fusion*, vol. 55, pp. 45–58, Mar. 2020, doi: [10.1016/j.inffus.2019.08.002](https://doi.org/10.1016/j.inffus.2019.08.002).
- [33] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent.*, 2015, pp. 1–15.
- [34] S. M. Arjovsky and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. 34th Int. Conf. Mach. Learn.*, Sydney, NSW, Australia, 2017, pp. 214–223.
- [35] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," in *Proc. Int. Conf. Inf. Process. Med. Imag.* Cham, Switzerland: Springer, 2017, pp. 146–157.
- [36] D. Soydaner, "A comparison of optimization algorithms for deep learning," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 34, no. 13, Dec. 2020, Art. no. 2052013.
- [37] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros, "Generative visual manipulation on the natural image manifold," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 597–613.
- [38] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," 2017, *arXiv:1701.04862*.
- [39] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.
- [40] P. Grzegorzewski and E. Mrówka, "Trapezoidal approximations of fuzzy numbers—Revisited," *Fuzzy Sets Syst.*, vol. 158, no. 7, pp. 757–768, 2007.
- [41] R. Perdisci, G. Gu, and W. Lee, "Using an ensemble of one-class SVM classifiers to harden payload-based anomaly detection systems," in *Proc. 6th Int. Conf. Data Mining (ICDM)*, Dec. 2006, pp. 488–498.
- [42] A. Basharat, A. Gritai, and M. Shah, "Learning object motion patterns for anomaly detection and improved object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1–8.
- [43] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Proc. 2nd Workshop Mach. Learn. Sensory Data Anal. (MLSDA)*, 2014, pp. 4–11.
- [44] I. Higgins *et al.*, "beta-VAE: Learning basic visual concepts with a constrained variational framework," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Toulon, France, 2016.
- [45] L. Ruff *et al.*, "Deep one-class classification," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4393–4402.
- [46] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled generative adversarial networks," 2016, *arXiv:1611.02163*.
- [47] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, 2011, pp. 315–323.
- [48] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, 2013, vol. 30, no. 1, pp. 1–3.



Mingjing Xu received the Ph.D. degree from the Department of Energy, Politecnico di Milano, Milan, Italy. He has published articles in journals *Engineering Applications of Artificial Intelligence* and *Energy*, and he is the author and coauthor of six papers on international journals and conferences. His research interests include the development of artificial intelligence techniques for fault diagnostics and prognostics of industrial components. He has been awarded the Best Student Poster in the European Safety and Reliability Conference, ESREL 2019, Hannover, Germany, and he has been awarded the Best Oral Presentation in a session in the International Conference on System Reliability and Safety, ICSRS 2019, Rome, Italy.



Piero Baraldi received the B.S. degree in nuclear engineering and the European Ph.D. degree in radiation science and engineering from the Politecnico di Milano, Italy, in 2002 and 2006, respectively.

He has been a Full Professor of nuclear engineering with the Department of Energy, Politecnico di Milano since September 2021. He is the coauthor of two books and more than 200 papers on international journals and proceedings of international conferences. His main research efforts are currently devoted to the development of methods and

techniques for system health monitoring, fault diagnostics, prognostics, and maintenance. He has been an invited Keynote Lecturer at the plenary sessions of the European Safety and Reliability Conference, ESREL 2014, Wroclaw, Poland; the 2016 Prognostics and System Health Management Conference, Chengdu, China; and the 4th International Conference on System Reliability and Safety (ICSRS 2019), Rome, Italy. He has been invited to present four tutorials at international conferences.



Xuefei Lu received the Ph.D. degree in statistics from Bocconi University, Italy.

She has been working as a Post-Doctoral Fellow with the Politecnico di Milano and a Lecturer with The University of Edinburgh. In 2021, she joined the SKEMA Business School as an Assistant Professor. Her research interests include statistical machine learning and uncertainty quantification.



Enrico Zio (Senior Member, IEEE) received the first Ph.D. degree from the Politecnico di Milano, Italy, in 1996, and the second Ph.D. degree in probabilistic risk assessment from MIT, in 1998. He is currently a Full Professor with the Centre for Research on Risk and Crises (CRC), École de Mines, ParisTech, PSL University, France; a Full Professor and the President of the Alumni Association at the Politecnico di Milano; a Distinguished Guest Professor at Tsinghua University, Beijing, China; an Adjunct Professor with the City University of Hong Kong,

Beihang University, Beijing, China, and Wuhan University, China; and the Co-Director of the Center for RELiability and Safety of Critical Infrastructures (CRESCI) and the sinofrench laboratory of Risk Science and Engineering (RISE), Beihang University. His research focuses on the modeling of the failure repair maintenance behavior of components and complex systems; the analysis of their reliability, maintainability, prognostics, safety, vulnerability, resilience and security characteristics; and the development and use of Monte Carlo simulation methods, artificial techniques, and optimization heuristics. He is the author or coauthor of seven books and more than 300 articles on international journals, the Chairman and the Co-Chairman of several international conferences, an associate editor of several international journals, and a referee of more than 20.