Fusing convolutional generative adversarial encoders for 3D printer fault detection with only normal condition signals

Chuan Li [a,1], Diego Cabrera [a,b,*,1], Fernando Sancho [c], René-Vinicio Sánchez [b], Mariela Cerrada [b], Jianyu Long [d], José Valente de Oliveira [e]

[a] National Research Base of Intelligent Manufacturing Service, Chongqing Technology and Business University, Chongqing 400067, China
[b] GIDTEC, Universidad Politécnica Salesiana, Ecuador
[c] Dpt. of Computer Science and Artificial Intelligence, Universidad de Sevilla, Spain
[d] School of Mechanical and Automotive Engineering, South China University of Technology, Guangzhou, China
[e] Universidade do Algarve, Portugal

Keywords:
Fault detection
3D printers
Condition-based maintenance
Convolutional neural networks
Adversarial learning

A B S T R A C T

Collecting data from mechanical systems in abnormal conditions is expensive and time consuming. Consequently, fault detection approaches based on classical supervised learning working with both normal and abnormal data are not applicable in some condition-based maintenance tasks. To address this problem, this paper proposes Fusing Convolutional Generative Adversarial Encoders (fCGAE) method to create fault detection models from only normal data. Firstly, to obtain an adequate deep feature space, encoder models based on 1D convolutional neural networks are created. Then, these encoders are optimized in an unsupervised way through Bidirectional Generative Adversarial Networks. Finally, the multi-channel features collected from the system are merged with One-Class Support Vector Machine. fCGAE is applied to fault detection in 3D printers, where experimental results in two fault detection cases show excellent generalization capabilities and better performance compared to peer methods.

1. Introduction

3D printing has been attracting much attention as an innovation technology in the industry [1], as well as in medical [2], food [3], and other areas [4]. This is due to the accuracy level achieved by this technology [5], which improves the quality of the obtained products. Of course, the quality that a 3D printer offers depends mainly on the transmission elements that constitute it [6].

Despite the fact that the quality of the components of a 3D printer is the best or not, these elements suffer from wear and tear due to continuous use in the printing process, as in any other machinery [7]. Additionally, the union of the components can be compromised by the loosening of the fastening elements such as nuts, bolts, and join bearings. Those wear and/or loosening cause unintended movements in the printer head along with unwanted vibrations of the machine [8], resulting into loss of print quality, an accelerated deterioration of other components of the printer [9], and possible risk of damage

* Corresponding author at: National Research Base of Intelligent Manufacturing Service, Chongqing Technology and Business University, Chongqing 400067, China.
E-mail address: dcabrera@ups.edu.ec (D. Cabrera).
[1] The two authors contributed equally to this paper.

to the machine or operator safety. Hence, it is critical to estimate the condition of the machine in order to make necessary adjustments or replace defective parts.

Several efforts have focused on fault detection and diagnosis for 3D printers. For example, in [10] a method to combine different sources of information was proposed for estimating the printing quality. The estimation of the machine condition itself was not performed, but only informed if an error occurred during the printing process. In [11], an approach based on the filtering of acoustic emission signal and the Kurtogram extraction of a photoelectric signal was considered for the fault diagnosis in additive manufacturing. In that work, features are extracted considering machinery expert knowledge. With the same application, in [12] fault diagnosis was proposed by using Support Vector Machines (SVM) with signals obtained from an attitude sensor. In [13,14] the 3D printer condition is estimated by Echo State Networks (ESN), while [15] introduced a method improving an extreme learning machine through a modified swarm optimizer, and [16] presented the feature reinforcement for improving the 3D printer condition classification with a success rate in the normal condition of 93.6%. In addition, the comparison results reported 66.7%, 67.5%, 90.1%, 32.0% and 21.3% with SAE+Softmax, ESN, SAE+ESN, SAE+SVM, and SVM, respectively, highlighting the difficulty to correctly detect a fault even with powerful pattern recognition algorithms and supervised learning approachs. [17] introduced a method for unsupervisedly optimizing the reservoir of an ESN and applied it to this problem. And [18] used a sparse autoencoder network (SAE) for fault diagnosis of a delta 3D printer.

Authors from [19] presented transfer SVM for fault diagnosis of 3D printers in light of the lack of data on fault conditions by transferring learning from other domains with more available data, achieving a classification accuracy of 83.79%. [20] introduced the sensor fusion by error merging of SAE to dynamically assess the condition of a 3D printer. These last two works are the only ones that address the problem of lack of data for training classification/regression models for 3D printers with promising results.

More works on fault diagnosis and fault detection have been reported for other machines. Approaches based on stochastic resonance for detection and diagnosis of machinery failures were presented in [21–23]. Techniques for fault diagnosis based on signal analysis were detailed in [24], with emphasis on the extraction of periodic impulses in [25] and analysis in the time–frequency domain in [26]. In addition, a considerable amount of works using artificial intelligence techniques have also been reported. For example, [27,28] compared several approaches based on fuzzy logic for bearing diagnostics, and [29] presented automatic feature extraction by convolutional auto-encoder to deal with feature engineering in helical gears. A more exhaustive review can be found in [30].

All these works do not consider the difficulty of obtaining abnormal data of the machinery to create a classification model through supervised learning, but some other works have also been reported addressing this problem. For example, in [31] an Echo State Network was combined with a Variational Auto-Encoder for the learning of a detection model only from normal condition data of a helical gearbox. [32] presented Generative Adversarial Networks (GAN) as a model able to capture the data manifold in normal condition using its discriminator model. In [33] Categorical Adversarial Auto-Encoder model was presented for data clustering where condition was unknown. [34,35] used One-Class SVM (OCSVM) model for the detection of bearing failures, and [36] introduced an improved SVM for aircraft engine fault detection with imbalanced data.

However, it remains a challenging task to detect failures in new domains with only data from normal condition, where expert knowledge is not available to apply feature engineering-based approaches to determine machinery condition, and where the sources of information are diverse. This paper proposes a way to solve this challenge through fusing Convolutional Generative Adversarial Encoders (fCGAE) applied to fault detection in delta 3D printers. Main contributions of this work include: (i) a Convolutional Generative Adversarial Encoder (CGAE) model that integrates 1D Convolutional Neural Networks (1DCNN) with Bidirectional Generative Adversarial Neural Networks (BiGAN), in order to map raw signals to an informative feature space for fault detection; and (ii) a proposal for the fusion of deep features extracted in a non-supervised CGAE from multi-channel signals, and the creation of a discriminant function from them. The results obtained in fault detection in 3D printers show a better performance of fCGAE compared to approaches reported in the literature using classical feature engineering and state of the art Deep Learning.

The paper is organized as follows. Section 2 presents the details of the proposal. Experiments for the validation of the method are presented in Section 4. The results and comparisons obtained in each fault diagnosis study case are described in Section 5. Finally, Section 6 presents some conclusions and future work.

2. Methodology

2.1. Signal augmentation

The operating condition of a 3D printer can be inferred through measurements of sensors collecting information of different variables. In this research, both 3-axis velocity sensors and 3-axis angle sensors are considered:

$$s(t) = \{s^i(t)\}_{i=1}^{6} \tag{1}$$

where $s^1, s^2, s^3, s^4, s^5, s^6$ denote, respectively, the x, y and z velocity and x, y and z-related angle signals.

By using the window slicing technique for data augmentation in time-series datasets [37], and choosing appropriate values for the length of the slices (τ_l) and the time-steps before the next slicing (τ_s), we can extract K short signals from every original one, obtaining a larger set of smaller signals (that can overlap):

$$s \mapsto \{s_{[k\,\tau_s+1, k\,\tau_s+1+\tau_l]}\}_{k=0}^{K-1} \tag{2}$$

2.2. CGAE-based learning of the condition features

In recent years, a large number of proposals based on deep learning have been reported for feature extraction and representation learning tasks. Their application fields are as diverse as the different architectures proposed for each specific task. In general, however, their power lies in some hierarchical feature extraction process through the stacking of specialized computational layers for pattern recognition and classification.

Although the idea of representing learning through a feature space transformation was conceived previously to Deep Learning and several models from the traditional machine learning use this technique (for example, SVM model and MLP model), one of the credits of Deep Learning is to focus attention on this mechanism as a means of improving models only limited by the available computational resources. In the context of fault diagnosis applications, the efforts are focused on learning the transformation providing the better representation of input signals. For the i-th sensor, we can write this transformation as:

$$x^i = Enc(s^i(t); \phi) \tag{3}$$

where $Enc(\cdot;\phi)$ represents the learned transformation parameterized by ϕ, also called encoder, and x^i is the obtained representation for $s^i(t)$.

With a good representation, building a condition monitoring model can be reduced to approximate the unknown conditional probability distribution of a failure condition y given the representation $x^i, P(Y|X^i)$, and then the resulting failure is selected through a θ-parameterized model with the following criteria:

$$y = \text{argmax}_y P(y|x^i;\theta) \tag{4}$$

If we denote by D^i the training data set composed by tuples (s^i, y) from all the 3D printer conditions to be identified in the i-th sensor, the maximum likelihood estimation (MLE) approach can be used for finding θ. Combining this with Eq. (3), we can write:

$$\theta = \text{argmax}_\theta \prod_{(s^i,y)\in D^i} P(y|Enc(s^i,\phi);\theta) \tag{5}$$

Several proposals have been introduced for tuning ϕ, as autoencoders or Boltzmann machines. In all cases, the encoder is unsupervisedly pre-trained (without using y_m) and then it is fine-tuned for the specific classification task through the MLE criterion.

However, this approach cannot be applied for fault detection in real applications because the size of D^i for some of the different conditions of the printer is minimal, or even non-existing, during the building phase of the model. Usually, only a large number of examples from normal condition are available in a real situation.

With this in mind, we present a CGAE model for representing a useful transformation for the fault detection of 3D printers, which is learned only from signals in normal (healthy) condition. It consists of two 1DCNN and one 1D Transpose-Convolutional Neural Network (1DTCNN), both able to deal directly with input time-series with a reduced number of parameters, which decrease the risk of overfitting. 1DCNN is a model composed of a set of convolutional layers, where for a layer l there is an input in the form of a multichannel time-series, $\tilde{s}_l^i(t) \in \mathbb{R}^{T\times N_c}$, to which a set of 1D convolution filters, $h_l^i(t) \times \mathbb{R}^{W,N_c,N_o}$, is applied to produce an output multichannel time-series, $o_l^i(t) \in \mathbb{R}^{T_o\times N_o}$, as follows:

$$o_l^{i,n_o}(t) = \sum_{n_c=1}^{N_c} \tilde{s}_l^{i,n_c}(t) * h_l^{i,n_c,n_o}(t) + b_l^{i,n_o},\ \forall n_o = 1,\ldots,N_o \tag{6}$$

where $\tilde{s}_1^{i,1}(t) = \tilde{s}^i(t)$, and T, T_o, N_c, N_o and W stands for the length of input signal, length of output signal, number of input channels, number of required output channels after the convolution operation, and length of 1D convolution filters, respectively, b_l^{i,n_o} is a bias term applied to adjust an extra displacement in case to be required, and $*$ is the convolution operator with the intrinsic parameters *padding* and *stride* that maintains the input length and downsamples the input signal [38]. All these parameters depend on l and can be different in each layer.

The input $\tilde{s}_l^i(t)$ is obtained from the application of batch normalization (BN) [39] to $s_l^i(t)$, as follows:

$$\bar{s}_l^i(t) = \frac{s_l^i(t) - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{7}$$

$$\tilde{s}_l^i(t) = \eta \bar{s}_l^i(t) + \beta \tag{8}$$

where μ and σ^2 stands for the mean and variance of the signal batch in training time, respectively (or an estimation of these values in testing time), and η and β are the scaling and shifting parameters, which are estimated in training time. ϵ is a

constant close to zero, which is added to avoid the indetermination when $\sigma^2 \to 0$. BN is applied to the inputs of each convolutional layer except the last one to avoid adding extra variance to the output of 1DCNN.

After that, a nonlinear element-wise function $f(\cdot)$ is applied to the output of Eq. (6):

$$\tilde{s}_{l+1}^{i,n_c}(t) = f[o_l^{i,n_o}(t)] \tag{9}$$

where we must take into account that the N_o output channels from l-th layer are the N_c input channels for $(l+1)$-th layer, and T_o can be computed from *padding* and *stride* parameters as:

$$T_o = \begin{cases} \frac{T-W}{stride} + 1, & \text{if } padding = valid \\ \frac{T}{stride}, & \text{if } padding = same \end{cases} \tag{10}$$

where $padding = same$ states for adding samples to the input signal for keeping $T_o = T$ (only when $stride = 1$), and $padding = valid$ states when no sample is added and the output length is mandatorily decremented. By controlling the *stride* parameter, the downsampling of the signal is modified in each layer. This process results in features with a high abstraction level and less time-dependency when more convolutional layers are stacked.

On the other hand, 1DTCNN model performs the inverse process of 1DCNN. It uses the transpose-convolutional layer to up-sampling the input features until they are transformed into time-series. Let $H \in \mathbb{R}^{T_o \times T_o}$ be the orthonormal transformation matrix built with a h filter such that:

$$\tilde{z} = H\tilde{s} \tag{11}$$

where $\tilde{z}$ is an input channel to a transpose-convolutional layer, and it is truth that $\tilde{z} = h * \tilde{s}$. Then, due to the orthonormality of H, the transpose convolution is simply defined as:

$$\tilde{s} = H^\top \tilde{z} \tag{12}$$

Therefore, the transpose-convolution operation over a set of input features $\tilde{z}_l^i \in \mathbb{R}^{T,N_c}$ and output of the l-th transpose-convolutional layer can be defined as:

$$o_l^{i,n_o}(t) = \sum_{n_c=1}^{N_c} (H_l^{i,n_c,n_o})^\top \tilde{z}_l^{i,n_c} + b_l^{i,n_o} \tag{13}$$

$$\tilde{z}_{l+1}^{i,n_c}(t) = g[o_l^{i,n_o}(t)] \tag{14}$$

$$\tilde{z}_1^{i,1} = \tilde{z}^i \tag{15}$$

Again, $\tilde{z}_l^{i,n_c}$ is the result of batch normalization over z_l^{i,n_c}, and a nonlinear function, $g(\cdot)$, is applied to $o_l^i \in \mathbb{R}^{T_o,N_o}$ to obtain the output. As before, the output of the l-th layer is the input for the next one.

As mentioned above, the purpose of using 1DTCNN layers in our fault detection context is to transform from a feature space to the sensor time-series space. Then, typically $T_o > T$ and it can be computed according to the same parameters, *padding* and *stride* as follows:

$$T_o = \begin{cases} (T-1)\,stride + W, & \text{if } padding = valid \\ T\,stride, & \text{if } padding = same \end{cases} \tag{16}$$

The training process of the CGAE model is based on BiGAN as proposed in [40]. It is a 1DCNN built for approximating an optimum vector code $z^i \sim P_{Enc}$ given $s^i \sim P_R$ signal input, i.e $x^i = Enc(s^i)$. Our hypothesis states that if the model is trained only with normal condition examples (with P_R probability distribution), then the resulting codes for signals with similar distribution will occupy a closer place in the feature space than signals measured in a failure condition of the 3D printer.

Additionally to *Enc*, BiGAN is composed by two more models interacting with each other: a generator model (*Gen*), and a discriminator model (*Dis*). *Gen* is a 1DTCNN which receives $z^i \sim P_z$, an input sampled from a known distribution, and transforms it into a signal $s^i \sim P_{Gen}$. *Dis* is a 1DCNN which takes input tuples (s^i, z^i) and returns its result according to the next function:

$$D(s^i, z^i) = \begin{cases} 1, & \text{if} s^i \sim P_R \text{ and } z^i \sim P_{Enc} \\ 0, & \text{if} s^i \sim P_{Gen} \text{ and } z^i \sim P_z \end{cases} \tag{17}$$

Gen must be improved in order to fool *Dis*. *Dis* improves avoiding to be fooled by *Gen* and, at the same time, correctly detects the inputs from P_R. If $\theta_{Dis}, \theta_{Gen}, \theta_{Enc}$ are the parameters of *Dis*, *Gen* and *Enc*, respectively, the cross-entropy loss for its optimization can be determined as:

$$L(\theta_{Dis}, \theta_{Gen}, \theta_{Enc}) = \sum_{j=1}^{B} log P_{Dis}(y_j | s_j^i, z_j^i) \tag{18}$$

where $y_j|s_j^i, z_j^i$ is a known example in the data batch (with size B) with $y_j = 1$ if $s_j^i \sim P_R$ and $z_j^i \sim P_{Enc}$, and $y_j = 0$ otherwise. P_{Dis} distribution can be described as a Bernoulli distribution, and consequently from Eq. (18) we obtain:

$$L(\theta_{Dis}, \theta_{Gen}, \theta_{Enc}) = \sum_{j=1}^{B} log([Dis(s_j^i, z_j^i)]^{y_j}[1 - Dis(s_j^i, z_j^i)]^{1-y_j}) \tag{19}$$

$$= \sum_{j=1}^{B} log[Dis(s_j^i, z_j^i)]^{y_j} + \sum_{j=1}^{B} log[1 - Dis(s_j^i, z_j^i)]^{1-y_j} \tag{20}$$

$$= \sum_{j=1}^{B} y_j logDis(s_j^i, z_j^i) + \sum_{j=1}^{B} (1 - y_j) log[1 - Dis(s_j^i, z_j^i)] \tag{21}$$

Summarizing as expectations:

$$L(\theta_{Dis}, \theta_{Gen}, \theta_{Enc}) = \mathbb{E}_{s^i \sim P_R} logDis(s^i, Enc(s^i)) + \mathbb{E}_{z^i \sim P_z} log[1 - Dis(Gen(z^i), z^i)] \tag{22}$$

Therefore, the improvement of the models can be established as the next minimax optimization problem:

$$\theta_{Dis}, \theta_{Gen}, \theta_{Enc} = \text{argmin}_{\theta_{Gen}, \theta_{Enc}} \text{argmax}_{\theta_{Dis}} L(\theta_{Dis}, \theta_{Gen}, \theta_{Enc}) \tag{23}$$

The optimization (training stage) of Eq. (23) can be performed through the algorithm described in [41] for GAN-based models. The application of this procedure to each D^i signal set (from i-th sensor) results in one Enc^i model capturing the behavior of each i-th sensor under normal condition of the machine.

2.3. Deep encoded features fusion through SVM

Taking only the encoder into account, the optimization of Eq. (23) returns a set of improved deep convolutional networks able to map signals to feature spaces. The next step is to combine those spaces of independent features for each sensor into one that is enriched by the group of sensors. This task will be performed by concatenating the outputs of each Enc^i model, i.e.:

$$X = [Enc^1(\cdot); Enc^2(\cdot); Enc^3(\cdot); Enc^4(\cdot); Enc^5(\cdot); Enc^6(\cdot)] \tag{24}$$

where $Enc^i(\cdot)$ is a 1DCNN specialized in the i-th sensor and X is the enriched feature space.

For the creation of a decision function, firstly the samples from D^i are projected into X to build the set:

$$D = \bigcup_j \left[Enc^i(s_j^i(t))\right]_{i=1}^{6} \tag{25}$$

Now, a binary decision function that separates the region of the input space containing instances with the distribution of the examples from D can be created as an alternative solution to the problem posed in Eq. (4). To this end, we propose the use of a SVM-based approach. According to [42], f_{SVM} can be defined as:

$$f_{SVM}(x) = sgn\left(\sum_j \alpha_j k(x_j, x) - \rho\right) \tag{26}$$

where sgn takes values $\{-1, 0, 1\}$, α_j is the Lagrange multiplier associated with the support vector x_j ($\alpha_j > 0$) taken from the set D, and ρ is the offset factor of the decision hyperplane in a projection space determined by RBF-kernel:

$$k(x, y) = e^{-\gamma||x-y||^2} \tag{27}$$

where γ is the length scale parameter determining the influence radius in the enriched feature space of the support vectors.

The set of coefficients α of Eq. (26) can be computed using the method developed in [43] by solving the next optimization problem:

$$\alpha = \underset{\alpha}{\text{argmin}} \frac{1}{2} \sum_j \sum_{j'} \alpha_j \alpha_{j'} k(x_j, x_{j'}), s.t. 0 \leqslant \alpha_j \leqslant \frac{1}{v|D|}, \sum_j \alpha_j = 1 \tag{28}$$

where v is the upper bound of abnormal examples in D. And then ρ can be estimated by:

$$\rho = \sum_{j'} \alpha_j k(x_{j'}, x_j) \tag{29}$$

where x_j is any support vector associated with non-upper/lower bound Lagrange multiplier. Finally, v and γ are hyperparameters of the model that will be estimated according to the methods detailed in Section 3.2 or 3.3.

2.4. Overview of the proposed fCGAE approach for 3D printer fault detection

After obtaining the structure of fCGAE (Fig. 1 shows a summary of the process), it is necessary to train the model. $Enc^1, \ldots, Enc^6$ and f_{SVM} can be trained separately, but the outputs from each Enc^i model must be combined into f_{SVM} for data fusion training. The application of fCGAE for fault detection of 3D printers can be summarized as follows:

- *Step 1.* Collect $s^1(t), \ldots, s^6(t)$ from 3-axis velocity sensors and 3-axis angle sensors under normal condition in the 3D printer (one given task).
- *Step 2.* Compute augmented signal sets $D^1, \ldots, D^6$ using Eq. (2).
- *Step 3.* Build $Enc^1, \ldots, Enc^6, Gen^1, \ldots, Gen^6$ and $Dis^1, \ldots, Dis^6$ according to Eqs. (6), (9) (for *Enc* and *Dis*), and Eqs. (13), (14) (for *Gen*), and take as inputs $s^1(t), \ldots, s^6(t), Gen^1(z^1), \ldots, Gen^6(z^6)$ and $(s^1, z^1), \ldots, (s^6, z^6)$, respectively.
- *Step 4.* From the outputs of $Enc^1, \ldots, Enc^6$, build f_{SVM} for data fusion using the input given by Eq. (24).
- *Step 5.* Train $Enc^1, \ldots, Enc^6$ models separately by optimizing Eq. (22), and complete fCGAE training by feeding the results to f_{SVM} for its optimization.
- *Step 6.* Apply trained fCGAE for fault detection by entering new velocity and angle signals into it to obtain the machine condition.

3. Hyperparameters search

Two groups of hyperparameters need to be identified: those from the networks architecture, and those from OCSVM model. For the first group, we propose an heuristic-based approach for minimizing the amount of information lost between layers. For the second group, we evaluate two approaches; (i) cross-validated random search, and (ii) unsupervised search methods. Details are presented in the next paragraphs.

3.1. Networks architecture

The number of layers (L), number of output channels in every layer ($\{N_o^l\}$), kernel lengths ($\{W^l\}$), *stride* and *padding* define the architecture of *Enc*, *Dis*, and *Gen* networks. Their configuration follows a principle of minimum information lost between the network layers: given a raw signal of length T and configuring $stride = 2$ as the minimum downsampling for having translation invariance in each layer, L can be computed for the three networks as:

$$L = \lfloor log_2 T \rfloor \tag{30}$$

As the selected *stride* decreases the signal length to the half, N_o^l compensates this information loss, i.e., for *Enc* and *Dis*:

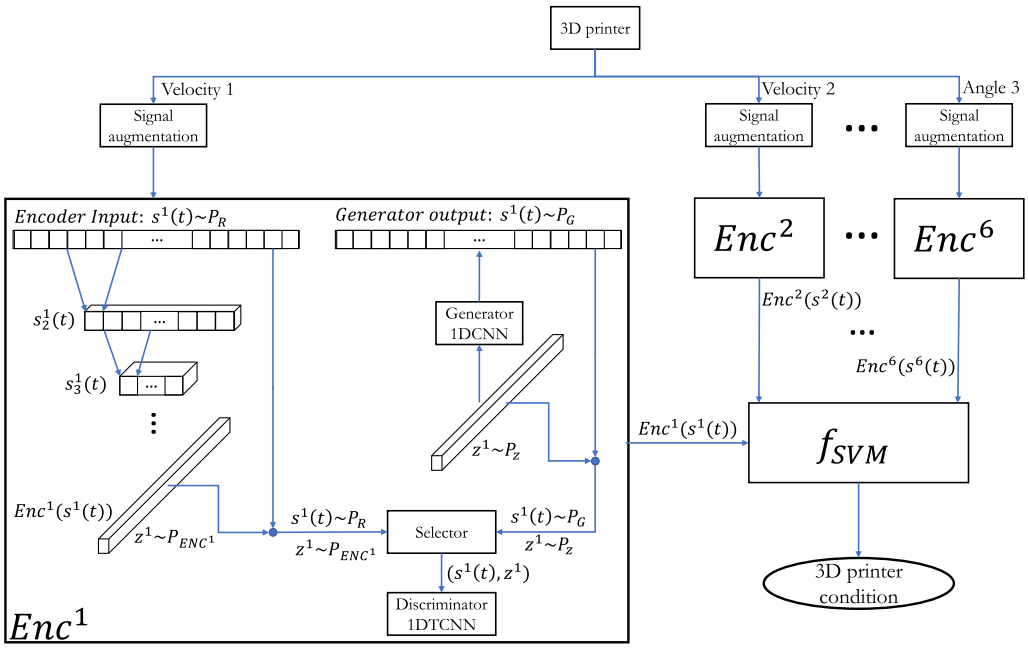


Fig. 1. Structure of fCGAE model for fault detection in a 3D printer.

$$\{N_o^l = 2^l\}_{l=1}^{L-1} \tag{31}$$

and for *Gen*:

$$\{N_o^l = 2^{L-l}\}_{l=2}^{L} \tag{32}$$

Setting *Enc* and *Gen* inverse architectures, N_o^L of *Enc* and N_o^1 of *Gen* are both the same as the dimension of the desired feature space. Furthermore, $N_o^L = 1$ for *Dis*, as requires an adversarial-based approach.

If T_{in}^l is the input signal length of the lth layer in *Enc* and *Dis*, then $padding = same$ if T_{in}^l is even and $l < L$, and $padding = valid$ otherwise.

If W_{max} is the maximum kernel length, then the W^l values for *Enc* and *Dis* must verify:

- if $padding = same$ and W_{max} can be applied at least 2 times in T_{in}^l, then $W^l = W_{max}$.
- if $padding = valid$, then $W^l = W_{max} - 1$. It produces an even input for the next layer.
- In the last layer, $W^L = T_{in}^L$ for computing the last features using the entire input.

W^l value of *Gen* is equal to $W^L - l + 1$ value of *Enc*, and the same inverse configuration must be applied for the *padding* parameter.

3.2. Cross-validated random search

Contrarily to grid search [44], Cross-Validated Random Search method (CVRS) evaluates only N candidates sampled from the hyperparameter search space. This method highly reduces the computational cost of the search process with almost similar results compared with the grid search in Deep Learning applications [45]. CVRS can optimize ν and γ parameters of our approach and also hyperparameters of other machine learning-based proposals. However, it is necessary to point out that CVRS assumes a calibration set available in the training stage with abnormal examples to evaluate the detection model.

CVRS requires a metric to evaluate the hyperparameter set quality. We choose the balanced accuracy since it considers the unbalance that could have the evaluation set in each iteration of the CVRS. Let tp, tn, fp and fn be the true positives, true negatives, false positives, and false negatives, respectively; the balanced accuracy for a binary classifier as $f_{SVM}(\cdot)$ is defined by:

$$\text{balanced accuracy} = \frac{\frac{tp}{tp+fn} + \frac{tn}{tn+fp}}{2} \tag{33}$$

Then, the CVRS procedure adapted for the one-class learning problem is:

1. Select N candidate sets of the search space according to the probability distribution of the hyperparameters.
2. Split the training and calibration sets into K partitions.
3. For $n = 1$ to N:
   (a) Configure the model with the n-th hyperparameter set.
   (b) For $k = 1$ to K:
      i. Train the model with the other $K - 1$ partitions (all but the k-th partition) of the training set.
      ii. Evaluate the model in the k-th partition of the training and calibration sets.
   (c) Compute the average balanced accuracy over the K previous results for the n-th hyperparameter set.
4. Choose the hyperparameter set with the highest average balanced accuracy.
5. Train a model with the entire training set and chosen hyperparameters.
6. Evaluate the model in the test set.

3.3. Unsupervised search methods

As it was stated before, CVRS requires a calibration set that could be difficult to have in the training stage. This restriction could be lightened if the cross-validation process uses a fixed calibration set for every iteration; in this case, the calibration set size is smaller than the training set (for $K = 10$, the calibration set size is $1/10$ of the training set).

In this work, we also tackle the worst case where a calibration set is not available for the hyperparameters search. This issue is addressed with the approaches proposed by Xiao et al. [46] and Ratsch et al. [47] for tuning γ and ν with only the one-class examples in the training set. That is, if $\{x_i\}_1^m$ be a one-class training set with m examples, then γ can be optimized maximizing the next objective function:

$$f(\gamma) = \frac{2}{m}\sum_{i=1}^{m} e^{-\gamma \min_{j\neq i} ||x_i - x_j||^2} - \frac{2}{m}\sum_{i=1}^{m} e^{-\gamma \max_{j} ||x_i - x_j||^2} \tag{34}$$

The $f(\gamma)$ function can be maximized with any optimization numerical method of scalar functions. In this work, we used Brent's method [48].

The v parameter optimization follows an interactive procedure to maximize the projected distance between the examples classified as $+1$ and -1 by f_{SVM}. Let $inc = \frac{1}{N_{nu}}$ the v increment of each iteration splitting the interval in N_{nu} homogeneous steps, then the procedure is as follow:

1. For $v = inc$ to $1 - inc$, in steps of *Inc.*:
   (a) Train OCSVM with the optimized γ.
   (b) Compute the average distance to the hyperplane of the examples with $f_{SVM} = +1$.
   (c) Compute the average distance to the hyperplane of the examples with $f_{SVM} = -1$.
   (d) Compute the total distance of the two groups by adding the previous distances.
2. Choose v that maximizes the total distance.

4. Experiments

In this section a 3D printer experimental setup is built to collect 3-axis velocity and 3-axis angle measurements and to test the proposed fCGAE technique to fault detection. Comparison with peer approaches are also introduced at the end of the section.

4.1. Delta 3D printer test bed

We tested the performance of this model on the experimental platform for 3D printers developed by our group. The 3D printer (SLD-BL600-6, brand SHILEIDI) to be diagnosed has a delta kinematic configuration with 3 degrees of freedom (Fig. 2). A bipolar step motor each join, which provides movement to a synchronous belt-driven transmission mechanism for the arm displacement. Each arm is composed of two metal rods coupled at their ends by universal join bearings.

A low-cost attitude sensor BWT901 type MEMS (brand WIT), installed at the base of the delta configuration, collects the 3-axis velocity and angle signals. This device includes an analog-to-digital conversion system with a sensitivity in the velocity measurement offered by the sensor manufacturer of 0.05 deg /s and 0.01 deg for the angular measurement. The sampling frequency was set at 100 Hz with time synchronization for both types of measurements. Digital samples were collected with a USB interface.

The signals were obtained under detailed conditions for every case study and captured in the computer while the base of the printer performs a circular movement of 75 mm radius with 20 repetitions in a total time of 324 s. The whole process was repeated 3 times with a uniformly distributed random starting time between 1 and 60 s after finishing the previous signal acquisition. Therefore, we obtained 3 multidimensional time series (3 dimensions for velocity and 3 for angle) with a duration of 32400 samples each.

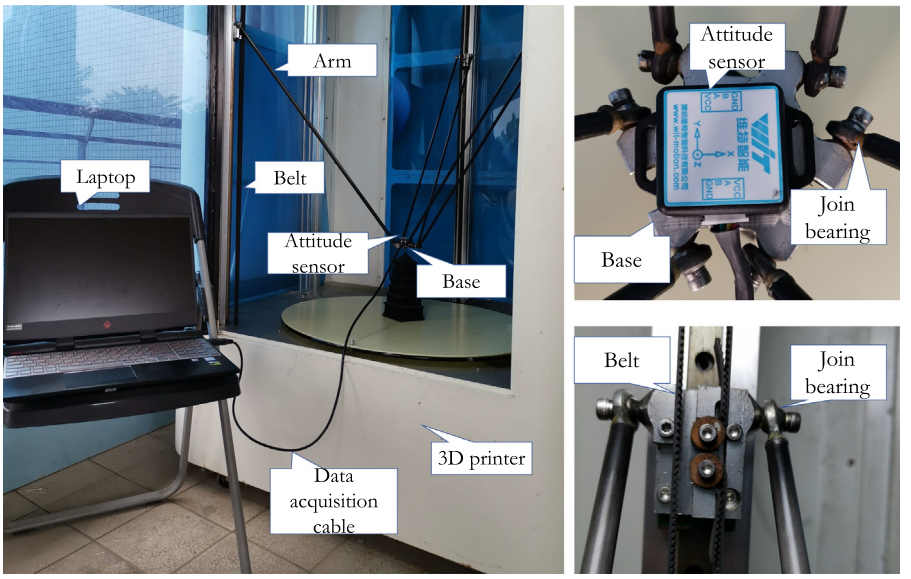
Later, for the practical application and evaluation of our test in a realistic scenario, it has to be performed at a trial stage where the machine makes one circular movement providing data to the model. This trial stage could be included before and/or after the printing working to avoid disturbances during the device operation. A trial-before allows detecting faults that could affect the printing quality. While a trial-after allows detecting faults resulting from a printing process. However, other trial stages could also be implemented in the middle of extensive printing workings, if possible. The real-time monitoring of the 3D printers under a variable operation condition is out of the scope of this work.

4.1.1. Case study 1: Join bearings

A 3D printer fault causes generally a loss in the final product quality. However, the acceptable production tolerances, which are related with the fault severity, depends on the specific application in which the product will be used. In this work, we empirically determined the severity by increasing the fault and inspecting signs of deterioration in the surface, corners and edges of the product. The previous procedure stops when the first signs of quality change appear. Therefore, the resulting failures can be considered as early-stage faults.

The first case study is about join bearing fault detection caused by loosening in the fastening screws, commonly caused by prolonged use of the machine. Products resulting from 3D printers with this failure present surface disturbances as shown in Fig. 3.

Faults can occur in any of the rods associated with an arm, as well as in any of its ends. In this way, 12 faults (2 rods $\times$ 2 ends $\times$ 3 arms) of this type could be configured (plus the normal condition, named *P*) according to their location as shown in Table 1. Each failure was artificially created by loosening of 0.7 mm in the screw (2 turns).



(a) Components of the experimental platform

(b) Base of delta kinematic configuration with coupled attitude sensor (up), and belt-driven transmission mechanism (bottom)

Fig. 2. Test bed of the 3D printer.

4.1.2. Case study 2: Belts

The second case study is fault detection in synchronous belts. This type of failure is also due to long periods of use of the machine, causing an additional stretch in the material of the belts. It produces additional oscillations when high acceleration braking is required, which results in a final product of low quality detail.

To obtain a dataset under this type of failure, 1.5 mm of belt was loosened from its optimal clamping position as shown in Fig. 4. Three fault conditions were obtained, one for each synchronization belt, named as M, N and O for arm 1, 2 and 3, respectively. As in the previous case, the normal condition of the machine is also available.

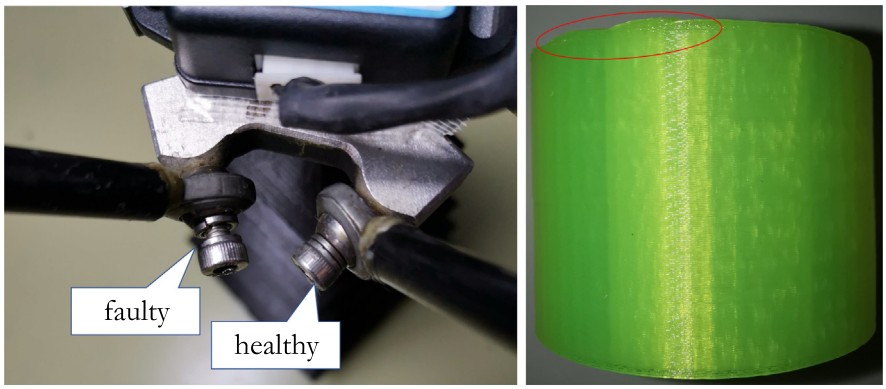
4.2. Experimental setup for comparisons

From the three multidimensional time series with length 32400 obtained under normal condition of the printer, two of them were used for training and one for testing. Then, the process described in SubSection 2.1 was applied to increase the size of the datasets with $\tau_l = 1620$ (32400/20) to obtain time series corresponding to a complete circle, and $\tau_s = 10$. From this, a training dataset with 6158 examples and a test dataset with 3079 examples are obtained under normal condition of the machine. Similarly, two signals of each fault condition were used to create the calibration set and one for the test set after applying them the same procedure.

4.2.1. Feature extraction

The models to be compared to our approach have difficulties in dealing with common representations of signals in domains such as time, frequency and/or time–frequency, therefore, as it was proposed in the methodology presented in [49], the crest factor, shape factor, absolute average amplitude, square root amplitude, kurtosis, variance, clearance factor, impulse indicator, and skewness factor were instead used as condition indicators for each of the 6 available signals in every example represented in the three domains.

There are 32 sub-signals obtained through the wavelet packet decomposition procedure until level 5 for the time–frequency representation, and nine condition indicators were obtained from each of these sub-signals. This results in a features vector of length 9 for time, same for frequency and, finally, another of length 288 for time–frequency domain, resulting in a vector of length 306 for each signal. Then, six vectors (one for each channel of the sensor) were fused into one with 1836 features to build a general representation of each available sample in the dataset.

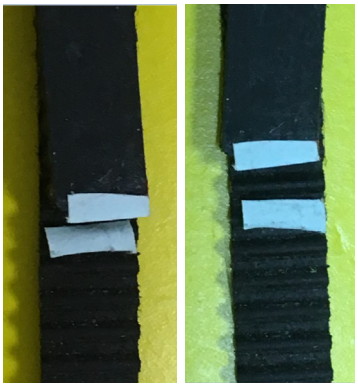


(a) Loosening in the fastening screws (b) Resulting product with a faulty 3D printer

Fig. 3. Example of fault in a join bearing.

Table 1
Simulated faulty conditions on join bearings.

Fault code	Arm	End	Rod
A	1	Base	Left
B	1	Base	Right
C	1	Rail	Left
D	1	Rail	Right
E	2	Base	Left
F	2	Base	Right
G	2	Rail	Left
H	2	Rail	Right
I	3	Base	Left
J	3	Base	Right
K	3	Rail	Left
L	3	Rail	Right



(a) Healthy belt (b) Artificial faulty belt

Fig. 4. Example of fault in a synchronization belt.

4.2.2. Compared approaches

As in our proposal, all the compared approaches use a model for the learning of condition features followed by a detection function. The hyperparameters of both elements were selected using the CVRS described in Section 3.2 with 128 candidates

and 10 folds. This configuration highly decreases the variance in similar applications of hyperparameter search in Deep Learning-based models [45].

The compared methods can be grouped according to condition features learning in: Traditional Machine Learning, Deep Learning. and Adversarial (our proposal) approaches. More details about the methods and their specific hyperparameters are provided below.

In addition to OCSVM, also the isolation Forest method (iForest) [50] has been evaluated as an alternative for the decision function for all the approaches. Table 2 summarizes the probability distribution sampled for each hyperparameter candidates of OCSVM and iForest in the CVRS.

Traditional Machine Learning

The first two comparison approaches were the robust OCSVM [51], and the anomaly isolation using iForest [52]. For both models, the inputs were the condition indicators previously presented. For OCSVM, a decision function is constructed according to Eq. (26) from the modeling of the normal behavior profile of the machine.

On the other hand, iForest seeks to isolate an example under fault conditions under the hypothesis that it requires fewer nodes in a set of random trees to be isolated than compared to those required by the examples in normal conditions. Like OCSVM, the random trees are created only with examples from normal condition.

Deep Learning

The next comparative approach is based on *Auto-Encoders*. This method has recently been evaluated for anomaly detection tasks [53]. Its architecture consists of two connected neural networks, an *encoder* and a *decoder*. The decoder input is the encoder output, commonly called *code*. This two networks are optimized together to minimize the error between the decoder output and the encoder input. Then, the encoder is optimized to map its input to an optimal code from which its input can be regenerated with the decoder (inverse function). Under this premise, the encoder is able to capture the data distribution in the code space.

From this idea, a fault detection model can be created with the following steps:

1. Optimize an autoencoder with the condition indicators of the training dataset.
2. Discard the decoder and map the training dataset with the optimized encoder.
3. Create a decision function based on OCSVM or iForest.
4. Evaluate the condition indicators-based new examples by mapping them to the code space with the encoder and evaluate the decision function to assess the 3D printer condition.

By following these steps and using two most popular techniques for optimizing autoencoder (denoising Auto-Encoder [54], dAE, and sparse Auto-Encoder [55], sAE), four encoder models were created: dAE + OCSVM (dAESVM), sAE + OCSVM (sAESVM), dAE + iForest (dAEForest) and sAE + iForest (sAEForest). In addition to the detection function hyperparameters, CVRS also tunes the hyperparameters of these autoencoder-based models listed in Table 3 with their probability distributions. Adam algorithm [56] optimized these models with a base learning rate and the number of epochs of 0.0002 and 1000, respectively. Adam is popular for training Deep Learning models due to adaptively compute an individual learning rate for each hyperparameter in the model, providing greater robustness in its selection.

Adversarial approach

To evaluate the incidence of multisensory fusion, three more models were created from our proposal: fusing only the velocity signals (fCGAE1), fusing only the angle signals (fCGAE2), and fusing both velocity and angle signals (fCGAE3). Additionally, the incidence of replacing the SVM-based deep features fusion with anomaly isolation based on iForest was also evaluated, obtaining three more models (named as fCGAE1F, fCGAE2F and fCGAE3F). In addition to the previous models whose decision function hyperparameters are tuned by CVRS, we also evaluate the worse situation where a calibration set is not available. For this, we consider fCGAE3u as fCGAE3 but replacing CVRS by the unsupervised methods for tuning γ and ν from Section 3.3.

In all cases, 1DCNN and 1DTCNN architectures were configured with the procedure of Section 3.1 that resulted in the architectures described in Tables 4–6 for *Enc*, *Gen* and *Dis* models, respectively.

The activation function of each layer is the leaky ReLU [57]. However, last layers for *Enc*, *Gen* and *Dis* models used linear activation functions. The input of *Enc* was a raw signal with shape 1620×1. Assuming a feature space of 100 dimensions, the input for *Gen* and output of *Enc* is a vector with shape 1×100. For the input of *Dis*, firstly the output of *Enc* was projected with a fully connected neural layer to a shape of 1620×1, and then it was joined to the input raw signal of the *Gen* output as it corresponded. The result was an input with shape 1620×2 for the *Dis* model.

Adam optimization algorithm [56] was set, as before, for training the models with a learning rate and number of epochs of 0.0002 and 1000, respectively. For these cases, the batch size was set to the maximum allowed value in the GPU GEFORCE GTX 1080 used for training the models (10 layers for *Gen*, *Enc* and *Dis*) as suggested in [58], i.e., 128 examples.

Table 2
Probability distributions of OCSVM and iForest hyperparameters.

Detection function	Hyperparameter	distribution	min value	max value	type
OCSVM	γ	*loguniform*	0.001	0.5	continuous
	ν	*loguniform*	0.001	0.5	continuous
iForest	contamination	*loguniform*	0.001	0.5	continuous
	features(%)	*uniform*	1	100	continuous
	estimators	*uniform*	1	500	discrete

Table 3
Probability distributions of AE hyperparameters.

Hyperparameter	distribution	min value	max value	type	values
layers	*loguniform*	1	10	discrete	increments of 1
neurons	*uniform*	10	2000	discrete	increments of 10
corruption level (dAE)	*loguniform*	0.0	1.0	continuous	–
sparse regularization (sAE)	*uniform*	0.0	1.0	continuous	–
activation function	*uniform*	–	–	discrete	*ReLU*, *LeakyReLU*, *tanh*, *sigmoid*
batch size	*uniform*	–	–	discrete	32, 64, 128, 256
normalization	*uniform*	–	–	discrete	standard, [0,1], [-1,1]

Table 4
Architecture of 1DCNN for *Enc* model.

No.	kernel length/*stride*	output channels	output shape	*padding*	Normalized?
1	4/2	2	810×2	*same*	*yes*
2	4/2	4	405×4	*same*	*yes*
3	3/2	8	202×8	*valid*	*yes*
4	4/2	16	101×16	*same*	*yes*
5	3/2	32	50×32	*valid*	*yes*
6	4/2	64	25×64	*same*	*yes*
7	3/2	128	12×128	*valid*	*yes*
8	4/2	256	6×256	*same*	*yes*
9	4/2	512	3×512	*same*	*yes*
10	3/2	100	1×100	*valid*	*no*

Table 5
Architecture of 1DTCNN for *Gen* model.

No.	kernel length/*stride*	output channels	output shape	*padding*	Normalized?
1	3/2	512	3×512	*valid*	*yes*
2	4/2	256	6×256	*same*	*yes*
3	4/2	128	12×128	*same*	*yes*
4	3/2	64	25×64	*valid*	*yes*
5	4/2	32	50×32	*same*	*yes*
6	3/2	16	101×16	*valid*	*yes*
7	4/2	8	202×8	*same*	*yes*
8	3/2	4	405×4	*valid*	*yes*
9	4/2	2	810×2	*same*	*yes*
10	4/2	1	1620×1	*same*	*no*

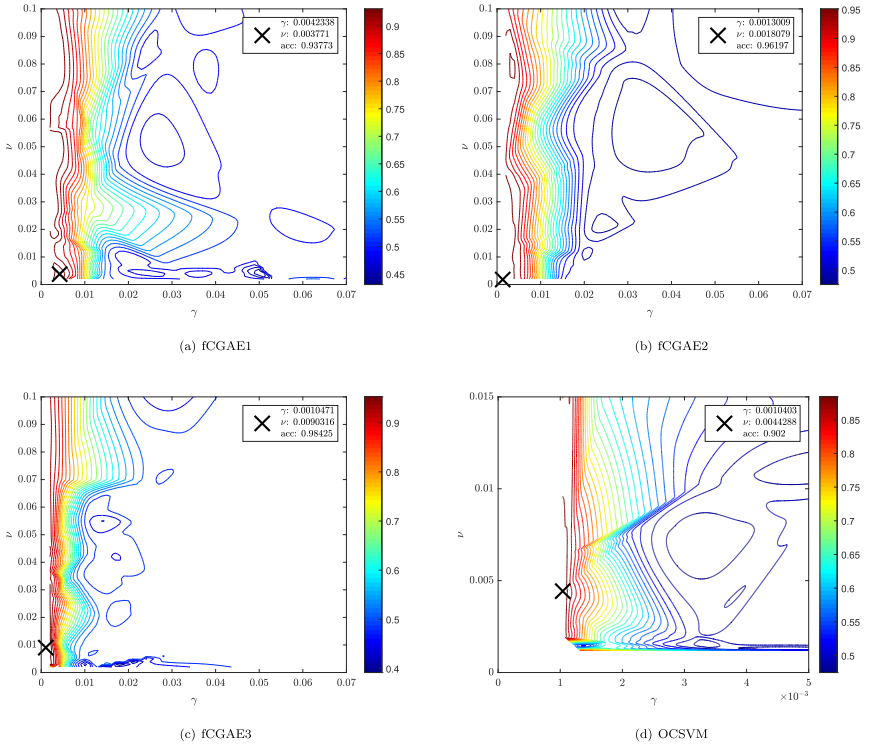
5. Results and discussion

5.1. Hyperparameters

Figs. 5 and 6 highlight the best hyperparameter configuration in the landscapes of the search spaces for the join bearing and belt study cases, respectively. These 3D landscapes can be created uniquely for models depending on two hyperparameters (γ and ν in our models), and they were generated by the 128 hyperparameter candidates sampled for the CVRS. Although the search range is [0.001, 0.5] for both hyperparameters, we restrict the area in the figures to only show the most interesting part, where most of the changes around the optimum can be appreciated. The landscapes illustrate a non-linear search space mostly depending on γ, meanwhile near the optimum values can be obtained for an extended ν range.

Table 6
Architecture of 1DCNN for *Dis* model.

No.	kernel length/*stride*	output channels	output shape	*padding*	Normalized?
1	4/2	2	810×2	*same*	*yes*
2	4/2	4	405×4	*same*	*yes*
3	3/2	8	202×8	*valid*	*yes*
4	4/2	16	101×16	*same*	*yes*
5	3/2	32	50×32	*valid*	*yes*
6	4/2	64	25×64	*same*	*yes*
7	3/2	128	12×128	*valid*	*yes*
8	4/2	256	6×256	*same*	*yes*
9	4/2	512	3×512	*same*	*yes*
10	3/2	1	1×1	*valid*	*no*



(a) fCGAE1 (b) fCGAE2

(c) fCGAE3 (d) OCSVM

Fig. 5. Landscape of the search space in join bearing study case.

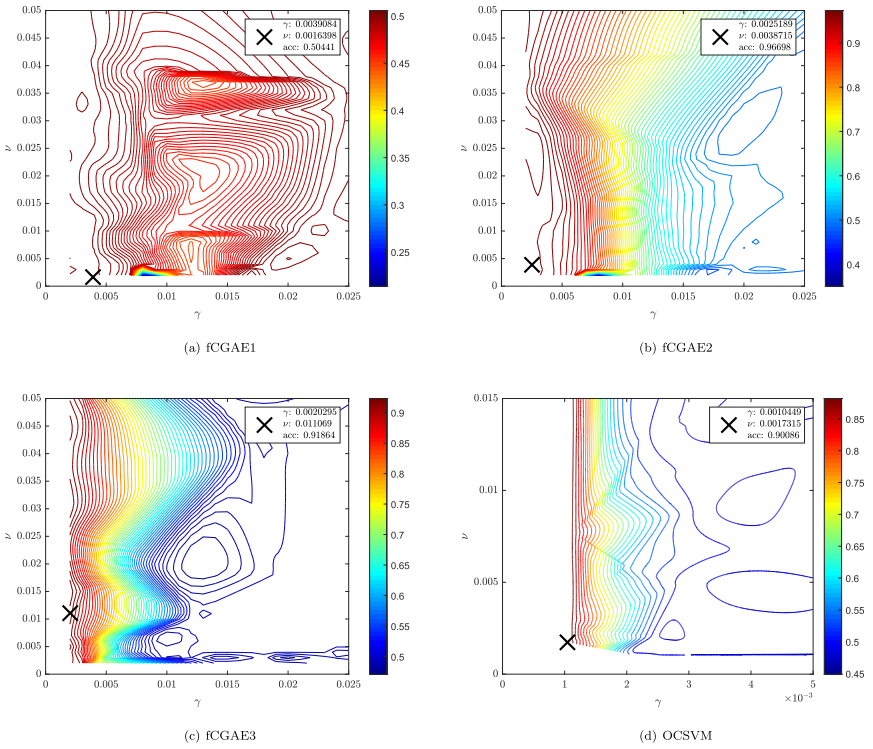
Concretely, the highest accuracy values are obtained with a small γ, suggesting a simple decision boundary (possibly linear or near-linear). Contrarily, the balanced accuracy of the fCGAE1 landscape remains in a low value independent on ν and γ.

Table 7 shows the architectures of the Deep Learning-based models found by the CVRS. Although the number of layers can be extended to 10, in most cases the selected number is lower than 4, thus reducing the model complexity and avoiding data overfitting in the learning process. $[0, 1]$ normalization was the winner in all the cases, something congruent with the domain of the leaky ReLU activation function.

Tables 8 and 9 summarize the hyperparameter search results through the CVRS with OCSVM and iForest decision function, respectively. Accuracy and margin columns together represent a confidence interval at 95% of confidence level over the mean of the balanced accuracies obtained in the cross-validation. The results show that the optimal values of ν are close to zero, which reinforces our fundamental knowledge that the training examples belong to the normal condition, and there are no examples that were initially misclassified. This suggests the healthy/faulty data points are highly separable in the feature space.

Also, Table 8 presents hyperparameter configuration for fCGAE3u obtained with the unsupervised methods. Contrarily to the other approaches, fCGAE3u has the same ν and γ for both study cases since the unsupervised search methods only depend on the training set with normal condition signals.

With iForest decision function, dAEForest and sAEForest have a low mean balanced accuracy for both study cases. These models also have high contamination evidencing a correlation between these two values. Then, we can conclude that the



(a) fCGAE1 (b) fCGAE2

(c) fCGAE3 (d) OCSVM

Fig. 6. Landscape of the search space in belt study case.

Table 7
Model architectures selected by CVRS of deep learning-based approaches.

	Bearing				Belt			
Hyperparameter	dAESVM	sAESVM	dAEForest	sAEForest	dAESVM	sAESVM	dAEForest	sAEForest
layers	4	3	5	4	3	2	3	3
neurons	[1290, 870,550, 300]	[1610, 570,380]	[1030, 770, 620520, 400]	[1500, 1390,990, 800]	[1250, 1100,920]	[1590, 1450]	[1680, 1540,1120]	[1350, 1260,1200]
corruption level (dAE)	0.01	–	0.011	–	0.01	–	0.012	–
sparse regularization (sAE)	–	0.12	–	0.11	–	0.11	–	0.1
activation function	leaky ReLU	leaky ReLU	leaky ReLU	leaky ReLU	leaky ReLU	leaky ReLU	leaky ReLU	leaky ReLU
batch size	128	256	128	256	128	256	128	256
normalization	[0, 1]	[0, 1]	[0, 1]	[0, 1]	[0, 1]	[0, 1]	[0, 1]	[0, 1]

networks of these models are not as good as those from the other models to extract features of the normal condition or the input features. The same is observed for fCGAE1F, but only in the belt study case. It suggests that velocity signals have less information about faults in belt.

5.2. Performance for case study 1

Comparative results of the success rate (detection rate) for the SVM-based models are presented in Table 10. The best normal condition detection rate of the models derived from our approach is obtained with fCGAE2 with a value of 97.99% compared to 93.44% and 96.98% obtained with fCGAE1 and fCGAE3, respectively. However, the worst detection rate with fCGAE1 is 77.59% in condition *H*, and 52.22% for fCGAE2 in condition *C*, while fCGAE3 has a perfect failure detection rate. This shows that fusing velocity and angle signals contributes to the detection of other types of faults that are not easily reflected by using only one type of signal. Clearly, this fusion has greater advantages at the cost of a minimal decrease in the detection rate of *P*. fCGAE3u obtains the performance closest to fCGAE3 with a detection rate of 90.16% for *P* condition and perfect failure detection. It is important to point out that the hyperparameters of this model were unsupervisedly

Table 8
Hyperparameter search results for models with a OCSVM decision function.

Case of study	model	γ	ν	accuracy(%)	margin(%)
Bearing	fCGAE1	0.0042	0.0038	93.77	1.88
	fCGAE2	0.0013	0.0018	96.20	2.56
	fCGAE3	0.0010	0.0090	98.42	1.88
	fCGAE3u	0.0024	0.0110	–	–
	OCSVM	0.0010	0.0044	90.20	3.03
	dAESVM	0.0052	0.0101	82.64	2.15
	sAESVM	0.0058	0.0297	82.15	1.96
Belt	fCGAE1	0.0039	0.0016	50.44	2.02
	fCGAE2	0.0025	0.0039	96.70	3.24
	fCGAE3	0.0020	0.0111	91.86	3.70
	fCGAE3u	0.0024	0.0110	–	–
	OCSVM	0.0010	0.0017	90.09	2.89
	dAESVM	0.0060	0.0085	81.56	4.77
	sAESVM	0.0058	0.0832	81.46	4.27

Table 9
Hyperparameter search results for models with a iForest decision function.

Case of study	model	contamination	features(%)	estimators	accuracy(%)	margin(%)
Bearing	fCGAE1F	0.0924	38.72	183	90.85	2.67
	fCGAE2F	0.0109	22.93	194	95.20	1.71
	fCGAE3F	0.0046	44.03	186	98.68	1.23
	iForest	0.0175	8.26	196	96.88	0.90
	dAEForest	0.2327	79.49	161	73.14	6.26
	sAEForest	0.2280	66.17	129	72.33	6.28
Belt	fCGAE1F	0.3014	8.58	3	50.53	5.00
	fCGAE2F	0.0522	58.23	177	94.94	4.64
	fCGAE3F	0.0605	36.79	121	87.92	2.28
	iForest	0.0460	8.85	171	94.88	2.89
	dAEForest	0.1799	54.25	197	74.89	7.74
	sAEForest	0.1615	17.25	187	73.32	4.06

Table 10
Success rate of SVM-based models applied to join bearing faults.

Condition	fCGAE1	fCGAE2	fCGAE3	fCGAE3u	OCSVM	dAESVM	sAESVM
P	93.44	97.99	96.98	90.16	75.61	84.38	81.84
A	100.0	100.0	100.0	100.0	100.0	98.77	98.73
B	94.06	100.0	100.0	100.0	100.0	73.30	72.04
C	100.0	52.22	100.0	100.0	100.0	80.16	85.25
D	99.90	100.0	100.0	100.0	100.0	97.43	95.13
E	100.0	100.0	100.0	100.0	100.0	80.32	77.85
F	96.82	100.0	100.0	100.0	100.0	61.64	83.79
G	87.40	100.0	100.0	100.0	100.0	80.22	76.19
H	77.59	100.0	100.0	100.0	100.0	96.04	96.59
I	80.74	100.0	100.0	100.0	100.0	94.02	96.62
J	95.97	100.0	100.0	100.0	100.0	85.90	95.78
K	86.68	100.0	100.0	100.0	100.0	68.33	73.34
L	94.38	100.0	100.0	100.0	100.0	54.47	59.60

calibrated without a calibration set with signals in failure conditions, showing that it is applicable under restricted data availability. OCSVM presents almost 25% of false alarms increasing the maintenance cost in a real situation. On the other hand, dAESVM and sAESVM models have some failure detection rates close to 50% with acceptable performance in the detection of *P* condition. This implies that almost half of the examples of faulty conditions have been classified as healthy (normal) condition, which is not acceptable in a fault detection system.

Table 11 shows the detection rate for iForest-based models. As before, the advantages of fusing multi-channel signals are appreciated in the fCGAE3F model. In this model also the detection rate of *P* increases at the expense of its decrease in *C* and *I* to 91.43% and 68.30%, respectively. Likewise, a general decrease in the failure detection rates is appreciated for the fCGAE1F and fCGAE2F models. This highlights the advantages of using SVM versus iForest for fault detection in 3D printers.

The above becomes even more evident when analyzing the results obtained with the iForest model. Although the detection rate has been increased for *P*, the failure detection rates have been in general decreased up to reach unacceptable values

Table 11
Success rate of iForest-based models applied to joint bearing faults.

Condition	fCGAE1F	fCGAE2F	fCGAE3F	iForest	dAEForest	sAEForest
P	87.40	98.80	99.58	94.84	72.39	73.69
A	100.0	100.0	100.0	100.0	98.67	96.07
B	94.80	100.0	100.0	100.0	73.47	57.29
C	99.97	6.72	91.43	100.0	80.77	62.68
D	99.32	100.0	100.0	100.0	95.78	87.33
E	100.0	100.0	100.0	59.89	78.17	60.41
F	96.30	100.0	99.06	100.0	51.19	64.05
G	87.11	100.0	100.0	84.77	82.10	68.40
H	78.17	100.0	100.0	100.0	94.74	82.59
I	78.89	86.72	68.30	97.43	90.16	82.04
J	92.17	100.0	100.0	98.57	90.58	72.23
K	86.29	100.0	100.0	90.81	67.26	53.07
L	93.57	100.0	100.0	99.61	57.65	43.46

such as 59.89% in the *E* condition. This indicates that SVM can generate better discrimination functions in complex representation spaces.

5.3. Performance for case study 2

Tables 12 and 13 show the detection rate for models based on SVM and iForest, respectively, for case study 2. Again, the best results are obtained by combining the two types of signals (six channels), independently whether SVM or iForest is used. The disadvantage of using only velocity signals (fCGAE1 and fCGAE1F) becomes more evident, where the performance decreases for the detection of all the faults of the case study. These results present even worse performance than autoencoder based models, that used features extracted from both velocity and angle signals.

As previously, fCGAE3u has an excellent performance considering the data restrictions it was created with. It is in third position in the best performances, surpassing all iForest-based models and only behind of fCGAE3 and fCGAE2.

To sum up, overall performance for the two study cases gives advantage to fCGAE3. Therefore, greater generalization capacity is appreciated for models that use SVM. Results from dAEForest and sAEForest in conjunction with dAESVM and sAESVM for both cases show that the obtained feature space is not informative enough to detect some faults for 3D printer. fCGAE3u gives a useful alternative in the worse data restriction scenario.

6. Conclusions

This work has proposed a new method for the construction of a fault detection model for 3D printers from the fusion of 3-axis velocity and angle signals obtained exclusively from normal condition. After the signal acquisition phase, a procedure for increasing the number of examples was applied over the healthy training dataset. With this augmented data, the construction of six 1DCNN encoder models was carried out with a BiGAN-based approach, and their outputs were fused into an OCSVM model optimized to create a normal condition profile with the available data. The resulting model was able to correctly discriminate new examples as healthy or faulty condition.

The proposed approach has been evaluated on two cases in 3D printers: fault detection of 12 different join bearings, and fault detection in 3 synchronous bands. Additionally, the method was compared with multiple methods reported in literature for anomaly detection, and also with the resulting models without fusing different type signals. The results allow us to conclude the following points:

- 1DCNN type encoders created with a GAN-based approach allow to build a deep feature space from raw signals without any pre-processing that is informative enough for the fault detection task, surpassing feature engineering-based approaches and autoencoders.
- The feature space created by the encoders can be used by models that characterize the normal condition (SVM) as well as by models that characterize the anomaly (iForest), with acceptable results in both cases.
- The fusion of multi-channel signals gives the resulting model a better generalization capacity for the detection of various faults, compared to that obtained from models of a single type of signals.
- A detection model with excellent performance can be created without a calibration dataset using the proposed heuristic for the network architectures together unsupervised approaches for tuning hyperparameters.

As future work, the exploration of the feature space is proposed for the identification of the type of failure and the possible assessment of its severity. Also, although this work is devoted to including an additional trial stage for detecting faults, which is feasible in most of the 3D printing applications, the extension of the proposed method to real-time 3D printer monitoring is an interesting topic to be explored in the future. This extension process should consider the intrinsic complexity added by non-stationary time series measured in a 3D printer working under variable operating conditions.

Table 12
Success rate of SVM-based models applied to belt faults.

Condition	fCGAE1	fCGAE2	fCGAE3	fCGAE3u	OCSVM	dAESVM	sAESVM
P	94.02	95.81	92.82	90.16	75.38	80.55	78.56
M	14.16	100.0	100.0	100.0	100.0	79.05	80.68
N	1.20	92.30	100.0	100.0	100.0	87.04	88.37
O	2.34	100.0	100.0	100.0	100.0	100.0	100.0

Table 13
Success rate of iForest-based models applied to belt faults.

Condition	fCGAE1F	fCGAE2F	fCGAE3F	iForest	dAEForest	sAEForest
P	70.09	90.19	90.09	89.90	79.96	82.14
M	31.83	100.0	100.0	95.88	69.99	70.87
N	22.15	95.41	99.97	100.0	80.09	75.35
O	22.09	100.0	100.0	100.0	100.0	100.0

CRediT authorship contribution statement

Chuan Li: Supervision, Writing - review & editing, Resources, Conceptualization, Funding acquisition. **Diego Cabrera:** Conceptualization, Methodology, Software, Validation, Investigation, Writing - original draft. **Fernando Sancho:** Supervision, Writing - review & editing, Conceptualization. **René-Vinicio Sánchez:** Resources, Funding acquisition, Writing - review & editing, Project administration. **Mariela Cerrada:** Resources, Funding acquisition, Writing - review & editing, Project administration. **Jianyu Long:** Resources, Funding acquisition, Writing - review & editing, Project administration. **José Valente de Oliveira:** Conceptualization, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The work was sponsored in part by GIDTEC Research Group of Universidad Politécnica Salesiana, the National Natural Science Foundation of China (51775112, 71801046), the National Key R&D Program (2016YFE0132200), the MoST Science and Technology Partnership Program (KY201802006), the Chongqing Natural Science Foundation (cstc2019jcyj-zdxmX0013), and the CTBU Project (KFJJ2018107, KFJJ2018075).

References

[1] T. Sathish, M. Vijayakumar, A.K. Ayyangar, Design and fabrication of industrial components using 3d printing, Materials Today: Proceedings 5 (2018) 14489–14498.
[2] Z. Zahedi-Tabar, S. Bagheri-Khoulenjani, H. Mirzadeh, S. Amanpour, 3d in vitro cancerous tumor models: Using 3d printers, Medical Hypotheses 124 (2019) 91–94.
[3] B. Pérez, H. Nykvist, A.F. Brøgger, M.B. Larsen, M.F. Falkeborg, Impact of macronutrients printability and 3d-printer parameters on 3d-food printing: A review, Food Chemistry 287 (2019) 249–257.
[4] M. Yampolskiy, A. Skjellum, M. Kretzschmar, R.A. Overfelt, K.R. Sloan, A. Yasinsac, Using 3d printers as weapons, International Journal of Critical Infrastructure Protection 14 (2016) 58–71.
[5] C. Balletti, M. Ballarin, F. Guerra, 3d printing: State of the art and future perspectives, Journal of Cultural Heritage 26 (2017) 172–182.
[6] T.D. Ngo, A. Kashani, G. Imbalzano, K.T. Nguyen, D. Hui, Additive manufacturing (3d printing): A review of materials, methods, applications and challenges, Composites Part B: Engineering 143 (2018) 172–196.
[7] A. Bewoor, S. Kulkarni, Interoperability of international standards, condition monitoring methods and research models for bearing fault: an integrated approach, Procedia Manufacturing 22 (2018) 982–989.
[8] A. Khadersab, S. Shivakumar, Parametric vibration analysis of rotating machinery, Materials Today: Proceedings 5 (2018) 25688–25696.
[9] Y. Lu, Y. Wang, Monitoring temperature in additive manufacturing with physics-based compressive sensing, Journal of Manufacturing Systems 48 (2018) 60–70.
[10] B. Li, L. Zhang, L. Ren, X. Luo, 3d printing fault detection based on process data, in: Proceedings of 2018 Chinese Intelligent Systems Conference, Springer, Singapore, 2018, pp. 385–396.
[11] J.K. Yoon, D. He, B.V. Hecke, A phm approach to additive manufacturing equipment health monitoring, fault diagnosis, and quality control, in: Annual Conference of the Prognostics and Health Management Society, 2014.
[12] K. He, Z. Yang, Y. Bai, J. Long, C. Li, Intelligent fault diagnosis of delta 3d printers using attitude sensors based on support vector machines, Sensors 18 (2018) 1298.
[13] S. Zhang, K. He, D. Cabrera, C. Li, Y. Bai, J. Long, Transmission condition monitoring of 3d printers based on the echo state network, Applied Sciences 9 (2019) 3058.
[14] D. Xiang, L. Jianyu, L. Chuan, C. Diego, Z. Shaohui, Intelligent fault diagnosis of 3d printers based on reservoir computing, International Journal of Performability Engineering 15 (2019) 3171.

[15] J. Long, Y. Hong, S. Zhang, D. Cabrera, J. Zhong, Improving extreme learning machine by a level-based learning swarm optimizer and its application to fault diagnosis of 3d printers, International Journal of Performability Engineering 15 (2019) 2972.
[16] S. Zhang, Z. Sun, C. Li, D. Cabrera, J. Long, Y. Bai, Deep hybrid state network with feature reinforcement for intelligent fault diagnosis of delta 3-d printers, IEEE Transactions on Industrial Informatics 16 (2020) 779–789.
[17] J. Long, S. Zhang, C. Li, Evolving deep echo state networks for intelligent fault diagnosis, IEEE Transactions on Industrial Informatics (2019), 1–1.
[18] J. Long, Z. Sun, C. Li, Y. Hong, Y. Bai, S. Zhang, A novel sparse echo autoencoder network for data-driven fault diagnosis of delta 3-d printers, IEEE Transactions on Instrumentation and Measurement (2019) 1–10.
[19] J. Guo, J. Wu, Z. Sun, J. Long, S. Zhang, Fault diagnosis of delta 3d printers using transfer support vector machine with attitude signals, IEEE Access 7 (2019) 40359–40368.
[20] S. Zhang, Z. Sun, J. Long, C. Li, Y. Bai, Dynamic condition monitoring for 3d printers by using error fusion of multiple sparse auto-encoders, Computers in Industry 105 (2019) 164–176.
[21] Z. Qiao, Y. Lei, N. Li, Applications of stochastic resonance to machinery fault detection: A review and tutorial, Mechanical Systems and Signal Processing 122 (2019) 502–536.
[22] Z. Li, B. Shi, A piecewise nonlinear stochastic resonance method and its application to incipient fault diagnosis of machinery, Chinese Journal of Physics 59 (2019) 126–137.
[23] X. Zhang, J. Wang, Z. Liu, J. Wang, Weak feature enhancement in machinery fault diagnosis using empirical wavelet transform and an improved adaptive bistable stochastic resonance, ISA Transactions 84 (2019) 283–295.
[24] Z. German-Sallo, G. Strnad, Machinery fault diagnosis using signal analysis, Procedia Manufacturing 32 (2019) 585–590.
[25] J. Li, X. Yao, H. Wang, J. Zhang, Periodic impulses extraction based on improved adaptive VMD and sparse code shrinkage denoising and its application in rotating machinery fault diagnosis, Mechanical Systems and Signal Processing 126 (2019) 568–589.
[26] L. Wang, Z. Liu, Q. Miao, X. Zhang, Time–frequency analysis based on ensemble local mean decomposition and fast kurtogram for rotating machinery fault diagnosis, Mechanical Systems and Signal Processing 103 (2018) 60–75.
[27] C. Li, J.L.V. de Oliveira, M.C. Lozada, D. Cabrera, V. Sanchez, G. Zurita, A systematic review of fuzzy formalisms for bearing fault diagnosis, IEEE Transactions on Fuzzy Systems (2018), 1–1.
[28] C. Li, M. Cerrada, D. Cabrera, R.V. Sanchez, F. Pacheco, G. Ulutagay, J.V. de Oliveira, A comparison of fuzzy clustering algorithms for bearing fault diagnosis, Journal of Intelligent & Fuzzy Systems 34 (2018) 3565–3580.
[29] D. Cabrera, F. Sancho, C. Li, M. Cerrada, R.-V. Sánchez, F. Pacheco, J.V. de Oliveira, Automatic feature extraction of time-series applied to fault severity assessment of helical gearbox in stationary and non-stationary speed operation, Applied Soft Computing 58 (2017) 53–64.
[30] 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.
[31] D. Cabrera, F. Sancho, M. Cerrada, R.-V. Sánchez, F. Tobar, Echo state network and variational autoencoder for efficient one-class learning on dynamical systems, Journal of Intelligent & Fuzzy Systems 34 (2018) 3799–3809.
[32] S. Plakias, Y.S. Boutalis, Exploiting the generative adversarial framework for one-class multi-dimensional fault detection, Neurocomputing 332 (2019) 396–405.
[33] H. Liu, J. Zhou, Y. Xu, Y. Zheng, X. Peng, W. Jiang, Unsupervised fault diagnosis of rolling bearings using a deep neural network based on generative adversarial networks, Neurocomputing 315 (2018) 412–424.
[34] J. Saari, D. Strömbergsson, J. Lundberg, A. Thomson, Detection and identification of windmill bearing faults using a one-class support vector machine, SVM, Measurement 137 (2019) 287–301.
[35] M. Zeng, Y. Yang, S. Luo, J. Cheng, One-class classification based on the convex hull for bearing fault detection, Mechanical Systems and Signal Processing 81 (2016) 274–293.
[36] P.-P. Xi, Y.-P. Zhao, P.-X. Wang, Z.-Q. Li, Y.-T. Pan, F.-Q. Song, Least squares support vector machine for class imbalance learning and their applications to fault detection of aircraft engine, Aerospace Science and Technology 84 (2019) 56–74.
[37] Z. Cui, W. Chen, Y. Chen, Multi-scale convolutional neural networks for time series classification, arXiv:1603.06995v4 (2016).
[38] V. Dumoulin, F. Visin, A guide to convolution arithmetic for deep learning, arXiv:1603.07285v2 (2018).
[39] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, arXiv:1502.03167v3 (2015).
[40] J. Donahue, P. Krähenbühl, T. Darrell, Adversarial feature learning, arXiv:1605.09782 (2017).
[41] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial networks, arXiv:1406.2661v1 (2014).
[42] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, J. Platt, Support vector method for novelty detection, (2000).
[43] B. Schölkopf, J.C. Platt, J. Shawe-Taylor, A.J. Smola, R.C. Williamson, Estimating the support of a high-dimensional distribution, Neural Computation 13 (2001) 1443–1471.
[44] J. Wainer, G. Cawley, Nested cross-validation when selecting classifiers is overzealous for most practical applications, arXiv:1809.09446v1 (2018).
[45] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, Journal of Machine Learning Research 13 (2012) 281–305.
[46] 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.
[47] G. Ratsch, S. Mika, B. Scholkopf, K.-R. Muller, Constructing boosting algorithms from SVMs: an application to one-class classification, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (2002) 1184–1199.
[48] R.P. Brent, An algorithm with guaranteed convergence for finding a zero of a function, The Computer Journal 14 (1971) 422–425.
[49] C. Li, R.-V. Sanchez, G. Zurita, M. Cerrada, D. Cabrera, R.E. Vásquez, Multimodal deep support vector classification with homologous features and its application to gearbox fault diagnosis, Neurocomputing 168 (2015) 119–127.
[50] F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation forest, in: 2008 Eighth IEEE International Conference on Data Mining, IEEE, 2008.
[51] Y. Xiao, H. Wang, W. Xu, J. Zhou, Robust one-class SVM for fault detection, Chemometrics and Intelligent Laboratory Systems 151 (2016) 15–25.
[52] G.A. Susto, A. Beghi, S. McLoone, Anomaly detection through on-line isolation forest: An application to plasma etching, in: 2017 28th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC), IEEE, 2017.
[53] A. Borghesi, A. Bartolini, M. Lombardi, M. Milano, L. Benini, Anomaly detection using autoencoders in high performance computing systems, arXiv:1811.05269v1 (2018).
[54] P. Vincent, H. Larochelle, Y. Bengio, P.-A. Manzagol, Extracting and composing robust features with denoising autoencoders, in: Proceedings of the 25th International Conference on Machine Learning – ICML, ACM Press, 2008.
[55] F. Li, J.M. Zurada, W. Wu, Sparse representation learning of data by autoencoders with regularization, Neural Network World 28 (2018) 133–147.
[56] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980v9 (2014).
[57] B. Xu, N. Wang, T. Chen, M. Li, Empirical evaluation of rectified activations in convolutional network, arXiv:1505.00853v2 (2015).
[58] S.L. Smith, P.-J. Kindermans, C. Ying, Q.V. Le, Don't decay the learning rate, increase the batch size, arXiv:1711.00489v2 (2017).