2018 European Control Conference (ECC)
June 12-15, 2018. Limassol, Cyprus

Generative Adversarial Networks for Unsupervised Fault Detection

Plakias Spyridon[1] and Yiannis S. Boutalis[1]

Abstract— **Fault detection is an important and demanding problem in industry. However, in many applications, abnormal data are difficult to be collected and are not available during training. In this paper, we propose a one class fault detection scheme, based on the unsupervised training of a Generative Adversarial Network (GAN). The generator tries to learn the manifold of normal behavior of the process, while the final decision of fault occurrence is taken from the discriminator. The network architectures of the discriminator and the generator and the hyper-parameters, that are used during training, are crucial for the stability of the GAN model. To ensure system convergence during training and to enhance the accuracy of the unsupervised classifier, a model selection algorithm is used. The latter one evaluates each trained model on a validation set of the normal training dataset, based on a proposed performance metric. Also, a method for evaluating the generating ability of each trained generative model is introduced, based on the reconstruction cost of an Autoencoder (AE). The proposed evaluation method of generative models, is used on the search algorithm, which selects the final classifier model. Finally, the proposed system is tested and compared with One-class SVM (OCSVM) on the industrial benchmark of Tennessee Eastman (TE) process with satisfactory results.**

I. INTRODUCTION

The demand for accurate, reliable and fast fault detection is raising over the past years, due to the increasing complexity and cost of many industrial processes. The techniques, that have been introduced for solving the problem, can be classified into three categories: model based methods [1], signal processing based methods [2] and pattern recognition or classification methods [3]. In model based methods we build a model of the system and we detect the fault by calculating the residuals between model output and real system measurements. The signal processing based methods are trying to resolve the nature of the detected faults by mathematical and statistical operations on the signal of the system. Over the past years many approaches of pattern recognition techniques for fault detection have appeared, attempting to find the best boundary condition between the classes.

In many industrial processes where the abnormal scenario is disastrous and difficult to appear in reality, the set of negative samples is limited. In this case of absent or few abnormal samples, binary classification algorithms are not convenient, so we have to use one-class classification methods (OCC) [4]. The goal of OCC algorithms is to recognize positive samples and so to identify negative ones when the training set is consisting only from positive samples. Several OCC algorithms have been proposed in the literature using Neural Networks, Nearest Neighbors, Decision Trees, Bayesian Classifiers and other. In the literature, a lot of OCC approaches are adopting one class SVM (OCSVM), due to the generalization of Support Vector Machines and their ability to deal with non linearly separable data using the kernel trick [5] [6] [7].

Recently Deep Learning (DL) Architectures have gained much attention in the field of computer vision, natural language processing, automatic speech and audio recognition, machine translation and others [8]. DL networks learn a better representation of the data by stacking multiple nonlinear hidden layers, using at the same time techniques to prevent over-fitting (Dropout) and to accelerate training (Batch Normalization). In this way, DL architectures extract features with different level of representation in each layer of the network, achieving hierarchical representation of learning.

Many researchers have addressed the use of DL architectures for fault and anomaly detection applications. In a recent paper, the problem of supervised (binary) fault detection has been attacked by building high level representation of features using a Sparse Auto-Encoder(sparse AE) [9]. Also, acoustic novelty detection applications have been introduced using AEs [10] [11]. The AE is trained on the dataset of normal acoustic signals and the novelty is detected if a difference metric between the input and the output of the AE exceeds a threshold. Another interesting work using DL architectures, propose an adaptive implementation of 1D Convolutional Neural Networks(CNN) for the real-time motor fault detection [12].

Generative models can estimate the probability distribution function of the training data and by achieving it, can generate samples similar with the ones of the training dataset [13]. The most popular generative methods are Autoregressive models, Variational Auto-Encoders (VAE) and Generative Adversarial Networks(GAN). GAN architectures [14] [15] are a hot topic of DL, especially in the field of image vision, with the production of realistic images. They are composed of two main components, the Generative and the Discriminator. The Generative is trying to create samples that look like the real ones and so to fool the discriminator. On the contrary the discriminator is trying to classify if a sample is real or fake.

The main drawback of GANs is their instability during training [16]. The balance between the Discriminator and the Generator is crucial to be maintained. So the architecture of the networks and the selection of the hyper-parameters during training is very important. In the literature, several techniques have been proposed that improve GAN stability

[1] S. Plakias and Y. Boutalis are with the Department of Electrical and Computer Engineering, Democritus University of Thrace, 67100 Xanthi, Greece splakias@ee.duth.gr, ybout@ee.duth.gr

during training [17]. Also, ambiguous is the evaluation of Generative Models, and GANs especially, considering that many models in computer vision are evaluated by human inspections.

An unsupervised anomaly detection scheme of medical images based on GAN architecture is presented in [18]. The proposed system detects images related with disease status from a set of unknown images, when the training dataset contains images only of healthy patients. The paper proposes the computation of an anomaly score for each image, based on the linear mapping of the discrimination loss and the residual loss. The final decision of anomaly detection is based on the comparison of the anomaly score with a predefined threshold. To ensure the stability of the GAN during training, they are adopting the hyper-parameters and network architectures of the widely used DCGAN model [15], trained on images of similar size. Furthermore, one question that arises is the selection of the predefined threshold, that is needed for the final decision, as much as the used hyper-parameters for the computation of the anomaly score.

In this paper a GAN architecture is proposed for the unsupervised fault detection. To the best of our knowledge it is the first time that fault detection is addressed using GANs. The final decision about the fault occurrence is taken by the Discriminator. In this way we achieve to avoid the use of unknown thresholds. To establish the stability during training, we practice feature matching for the training of the Generator and several other stability techniques proposed in the literature. Furthermore, we use an evaluation scheme of the performance of the Discriminator, based on a validation set of the original training set. The Discriminator accuracy in fake data is affected by the Generator ability to produce data that are similar to the real ones. To estimate the Generator ability to reproduce the set of normal samples, we propose an evaluation method based on Auto-Encoders (AEs).

The remaining sections are organized as follows. Section II describes the basic components of GANs and a theoretical basis of adversarial training. Also, it presents methods that strengthen the stability of GAN and enhance its generating skill. The proposed method for evaluating the Generative models quantitatively is given in Section III. Further, the search algorithm that compares each trained Discriminator by its capability to detect faults is described in section IV. Remaining Sections depict the application of the GAN model on the Tennessee Eastman benchmark, present the simulation results and their comparison with one class SVM at the same application. Finally, we conclude with conclusions and future work.

II. GENERATIVE ADVERSARIAL NETWORKS

A. Architecture of GAN

Generative Adversarial Networks (Goodfellow et al., [14] [16]) are consisting of two deep neural networks, the Generator and the Discriminator, challenging each other in a zero sum game (figure 1). Generator assignment is to fool the Discriminator by producing samples that look likes the real ones. To achieve this, the Generator tries to learn the

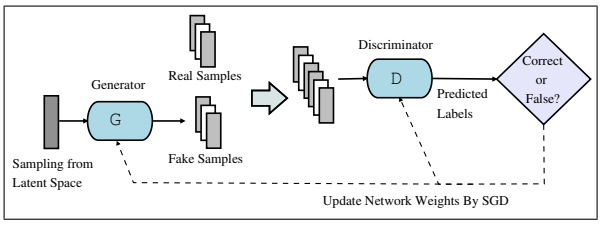

Fig. 1. Architecture of Generative Adversarial Network

distribution D_x of the training data and maps randomly samples z from a predefined latent space to the space of the distribution D_x. On the other hand, Discriminator attempts to distinguish the real samples from the generating ones. The reward of the Generator is increasing with the deceiving of the Discriminator in conflict with the Discriminator reward, which is closely related to the correct classification of real and generating samples.

Both Generator and Discriminator play an antagonistic game in each step of the training process. In particular, a minibatch of samples is produced from the Generator and is presented with an equal size minibatch of real samples to the Discriminator. Then, the discriminator decides about the authenticity of the samples. Finally, in each step of the training, we update the parameters of both Generator and Discriminator, by running simultaneously two stochastic gradient descent algorithms(SGD) and by trying to minimize their cost functions.

A brief mathematical formulation of GANs training follows. We consider that G corresponds to the Generator mapping of random samples z of the latent space Z to generating samples of the input space X, with parameters θ^G. Furthermore, we denote as D the decision function of the Discriminator, with parameters θ^D. More precisely, if x denotes an unknown sample in the input of the Discriminator, then $D(x)$ is the estimated probability of the latter, that x comes from the training data rather than the Generator.

The cost function of the Discriminator J^D is the standard cross-entropy function between the discriminator output and the actual labels (eq. 1).

$$\begin{aligned} J^D(\theta^D,\theta^G) = &-\frac{1}{2}\mathbf{E}_{x\sim p_{data}} log(D(x)) \\ &-\frac{1}{2}\mathbf{E}_{z\sim P(z)} log(1-D(G(x))) \end{aligned} \tag{1}$$

On the contrary, the cost function of the Generator J^G tries to maximize the log probability of false decisions of the Discriminator (eq. 2).

$$J^G(\theta^G,\theta^D) = -\frac{1}{2}\mathbf{E}_{z\sim P(z)} log(D(G(x))) \tag{2}$$

$P(z)$ is usually a predefined normal or uniform distribution with fixed latent dimensionality. A reader, who is interested about the sampling of latent space in GAN framework, may refer to [19].

B. Stability of GAN Training

The main weakness of GAN architecture is its instability during training. We see from equations (1) and (2) that the cost functions of both networks contain terms that do not belong to them but are controlled by the other competitive network. So, the training of the GAN architecture is more likely to be characterized as a game between two competitive agents than an optimization problem. In this case, the solution is not a local minimum of an objective cost function, but the Nash equilibrium of the game. In terms of game theory, the Nash equilibrium is a state where each agent have no benefit of changing his strategy. In the case of GAN training, the Nash equilibrium is a state where we have local minimum of J^D with respect to θ^D and simultaneously local minimum of J^G with respect to θ^G.

Also, another problem that drives the system to instability is Mode Collapse. It is characterized as the situation where Generator produces outputs that are identical. In most cases the probability distribution of training data is compound and it is composed of several peaks where each peak corresponds to a subset of the data. While Generator is trying to fool the Discriminator, it can learn to sample data from only a particular peak of the data distribution, inducing mode collapse.

There are several approaches that boost the generating performance of the GAN model and enhance its stability during training, by addressing the problem of mode collapse. Feature matching [17] is a technique that alters the objective function of the Generator, by using the output of an intermediate layer of the Discriminator. More particular, the expected values of the Discriminator features of the real samples are matched to those of the generating samples. The new objective of the Generator is

$$J^G = \|\mathbf{E}_{z\sim P(z)} f(G(x)) - \mathbf{E}_{x\sim p_{data}} f(x)\|^2 \tag{3}$$

where f denote the output of an intermediate layer of the Discriminator. Experimental results show the effectiveness of feature matching in cases where the GAN training has convergence problems.

Another technique for stabilizing the GAN architecture is the addition of instant noise to real and generating samples before their entrance in the Discriminator. This approach is tested using a GAN framework for image super-resolution and shows that boosts stability [20]. Also, dropout has been applied to the Generator for image-to-image translations problems [21]. The use of dropout increases the regularization of the Generator, addressing the problem of over-fitting, and helps the convergence during training.

III. Evaluation of GAN models based on AEs

Another important open problem of GAN framework is their quantitative evaluation. In the initial paper that introduced the GAN architecture [14], a nearest neighbor method from the set of generated samples is used for evaluating the model. Human inception [22] and the classification performance of the Generator [15] is applied for addressing the same problem. Also, [23] compares two GAN models, by forcing them to play one competitive game against each other.

In this work, we approach the evaluation of GAN models by using another successful architecture of DL, AutoEncoders (AEs) [24]. An AE is a feed forward neural network which tries to reconstruct its input. More particular, at first place an AE maps the input space to a hidden representation (encoding) and consequently the latent mapping is transfered back to the initial space (decoding). The objective function, which we try to minimize, is the reconstruction cost of the input. In that way, AEs learn a better representation of the data by using as output the hidden layer.

Also, AEs learn the manifold structure of the training data. As a consequence, someone can assume that a batch of samples, belonging to the same manifold of training data or near it, will have small reconstruction error when it will be applied to the AE.

In our approach we calculate the ratio between two reconstruction costs of the same AE. Initially, using the Generator of the trained GAN model, we produce samples with size equal to the original sample size. Then we train an AE, using as training data the set of generating samples and we estimate the reconstruction cost C^{fake} at the end of the training. Finally, we input the original data to the same AE, without training, and we calculate the corresponding reconstruction cost C^{real}. The evaluation grade of the Generator g^{eval} is given by the following equation.

$$g^{eval} = \frac{C^{fake}}{C^{real}} \tag{4}$$

We assume that a GAN model that produces samples more likely to the real ones, will have bigger evaluation grade g^{eval}. If the Generator produces samples that are close to the real ones, then the batch of generating data will belong to the same manifold, or near the manifold of real data. The AE is trained with the generating data. So, the ratio of the reconstruction cost of the generating samples to the reconstruction cost of the real samples, will indicate the generating ability of the model.

Furthermore, the proposed method check if mode collapse has been appeared. If the generating samples fall into one mode, then the AE is trained with data originated from this particular region. Hence, the reconstruction error of AE with the original data take large values and the evaluation ratio become small.

Consider N Generative Adversarial Networks $G_i, i = 1..N$, trained at the same dataset. Then, based on the proposed evaluation method, the Generator G_j produces samples more likely to the training data where $j = argmax(g_i^{eval})$. The algorithmic procedure for calculating the evaluation grade of each GAN model is shown in algorithm 1.

IV. Hyper-Parameter Selection of GAN model

The hyper-parameters of the model are crucial for the GAN stability and affect the classification performance of the Discriminator during the fault detection. The architectures of both Generator and Discriminator, the number of hidden

Algorithm 1 Calculate the evaluation grade g^{eval} of GAN model based on AE

1. Train the GAN model using the training dataset of real samples
2. Generate a batch of fake samples with equal size as the batch of real ones
3. Train an AE with training dataset the batch of fake samples and estimate C^{fake}
4. Calculate the reconstruction cost of AE for the set of real samples C^{real}, without training
5. Return the ratio $\frac{C^{fake}}{C^{real}}$

layers, the number of nodes in each layer and the dimension of the latent space can influence the training, driving the system into states that are not desirable. While training the Generator and the Discriminator in parallel, one common breakdown is the collapse of the power balance between the two networks, causing one to overpower other. In such a situation, the system does not improve as a whole and so the adversarial training breaks.

To overcome the above problem, we use a grid search algorithm as a model hyper-parameter optimization technique on a validation set of the normal training dataset. The classification ability of the Discriminator on normal data can be easily checked, using the unseen, during training, dataset of real examples. On the other hand, we don't have available data from the abnormal class. So, we use samples generated from the Generator to test the classification ability of the Discriminator on abnormal data. However, we must take into account the capability of each Generator to produce samples similar to real ones. Therefore, we calculate the normalized evaluation grade of each GAN model, and as final classification accuracy on fake examples, we use the product of normalized evaluation grades with the classification accuracies on generated samples, for checking the detection ability of each model for abnormal data. The final decision for the model selection is taken by calculating the g-mean accuracy of each model by the equation $g^{mean} = \sqrt{a^+ a^-}$ where a^+ and a^- is the classification accuracy on normal and abnormal data respectively. The above process of model selection is described by algorithm 2.

V. Unsupervised Fault Detection of Tennessee Eastman Process

A. Description of Tennessee Eastman (TE) process dataset

The TE process is a simulation model of a chemical industrial system described by [25]. The TE process dataset is extensively used from researchers as a fault detection benchmark. Beside the produced data of normal condition, the model can generate samples from 21 faulty situations. Both normal and faulty samples have 52 variables and are produced at a sampling time of 3 min. The training and testing set of the TE process benchmark consist of 480 and 960 samples for the normal class, restrictively. Also, the testing set for each faulty condition contains 960 samples,

Algorithm 2 Model Selection of GAN for unsupervised classification

1. Construct the grid of hyper-parameters
2. Train N GAN models, where each model corresponds to a combination of hyper-parameters (N is the number of hyper-parameter combinations)
3. For each trained model i, calculate the evaluation grade g_i^{eval} using algorithmic procedure 1
4. Calculate the normalized evaluation grade of each model with the equation $g_i^{neval} = \frac{g_i^{eval}}{\max_i g_i^{eval}}$
5. Find the accuracy of the Discriminator on real samples a_i^+ for each model i.
6. Find the accuracy of the Discriminator on fake samples a_i^- for each model i.
7. The accuracy on fake samples for each model is updated as $a_i^- = a_i^- * g_i^{neval}$
8. The g-mean of each model calculated by the equation $g_i^{mean} = \sqrt{a_i^+ a_i^-}$
9. Finally, select the model j where $j = \text{argmin}_i\, g_i^{mean}$

TABLE I
Evaluation of GAN models with search algorithm

Model	*Generator Nodes*	*Discriminator Nodes*	*Sampling dimension*	*Evaluating g-mean*	*Testing g-mean*
1	400	600	500	.882	73.76%
2	**400**	**600**	**1000**	**.953**	**75.04%**
3	400	800	500	.806	74.66%
4	400	800	1000	.939	75.10%
5	400	1000	500	.835	73.42%
6	400	1000	1000	.922	75.64%
7	500	600	500	.664	69.93%
8	500	600	1000	.567	68.08%
9	500	800	500	.734	73.39%
10	500	800	1000	.623	71.69%
11	500	1000	500	.739	73.39%
12	500	1000	1000	.489	66.04%
13	600	600	500	.687	62.32%
14	600	600	1000	.683	64.98%
15	600	800	500	.695	62.33%
16	600	800	1000	.703	58.40%
17	600	1000	500	.712	62.56%
18	600	1000	1000	.743	67.37%

where the fault occurs after 8 h. So, the first 160 samples are normal and the remaining 800 correspond to faulty states.

For the unsupervised training of both GAN and OCSVM models, we combine the sets that contain only normal data and as a result we have 1440 samples in the final produced training set. Also, the original testing sets for each fault condition of the TE process benchmark are used for the evaluation of the models during the test phase.

B. Training and Evaluation of GANs model

Both the Discriminator and Generator of GANs model have 5 hidden layers, same numbers of nodes in each of them and leaky rectified linear units (Leaky ReLU) as activation functions. The output layer of the Generator has linear mappings while the softmax function is used in the Discriminator's output.

TABLE II
FAULT DETECTIONS AND FALSE ALARMS RATES ON TE PROCESS

	Fault Detection Rates		False Alarm Rates		g-means	
Fault	GAN model	OCSVM	GAN model	OCSVM	GAN model	OCSVM
1	**99.625%**	99.5%	3.125%	**0.625%**	98.24%	**99.44%**
2	98.5%	98.5%	1.25%	**0%**	98.62%	**99.25%**
3	**10.375%**	7.625%	**7.5%**	10.625%	**30.98%**	26.11%
4	**56.25%**	50.375%	3.75%	**0.625%**	**73.58%**	70.75%
5	**32.375%**	30.5%	3.75%	**0.625%**	**55.82%**	55.05%
6	100%	100%	0%	0%	100%	100%
7	**100%**	99.625%	1.875%	**0%**	99.06%	**99.81%**
8	**97.875%**	97.375%	1.25%	**0%**	98.31%	**98.68%**
9	**8.625%**	7.125%	**9.375%**	21.875%	**27.96%**	23.59%
10	50.875%	**53.25%**	0.625%	**0%**	71.10%	**72.97%**
11	**58%**	54.75%	2.5%	**0.625%**	**75.20%**	73.76%
12	**98.75%**	98.625%	**4.375%**	13.125%	**97.17%**	92.56%
13	**95%**	94.875%	1.875%	1.875%	**96.55%**	96.49%
14	100%	100%	1.875%	**0%**	99.06%	**100%**
15	12.5%	**14%**	2.5%	**0%**	34.91%	**37.42%**
16	34.375%	**36.375%**	23.75%	**20.625%**	51.20%	**53.73%**
17	**91.125%**	87.25%	1.875%	**0%**	**94.56%**	93.41%
18	**90.375%**	90.125%	1.875%	**0%**	94.17%	**94.93%**
19	**11.875%**	3.75%	0.625%	0.625%	**34.35%**	19.30%
20	**58.375%**	52.75%	0%	0%	**76.40%**	72.63%
21	**49.875%**	41.875%	**5.625%**	6.25%	**68.61%**	62.66%
Average	**64.51%**	62.78%	3.77%	**3.69%**	**75.04%**	73.45%

During the training, Discriminator is supplied with different mini-batches for real and for fake samples and is trained twice for each epoch. In the hidden layers of the Generator, we adopt dropout regularization with probability of selected nodes equal to 0.5. Also, we prevent over-fitting in the Discriminator by adding Gaussian noise to its inputs and we use feature matching for improving the stability of the GAN model.

We are sampling randomly, for the input of the Generator, from a normal distribution with zero mean and standard deviation equal to 1. During the training process we use the Adam optimizer [26]. The AE, which is used for the evaluation of the Generator ability, has the simplest form with one hidden layer of 52 nodes and with the regularization form of weight decay ($C = 0.1$). By using equal number of nodes in the hidden layer and in the input layer of the AE and weight decay loss during training, we avoid over-fitting and under-fitting problems.

The hyper-parameters, that are selected using the described search algorithm of section IV, are the numbers of nodes in the hidden layers of the Generator and the Discriminator and the dimension of the latent space, that acts as input to the Generator. During the search algorithm, the parameter set $\{400,500,600\}$ is used for the nodes of the Generator, $\{600,850,1100\}$ for the nodes of the Discriminator and $\{500,1000\}$ for selecting the dimension of latent space.

The validation set is constructed by taking the first 500 samples of the training dataset and by using them during the learning process of the GAN models. The remaining samples are used for the calculation of the classification accuracy of Discriminators on normal data and so for the estimation of the evaluating g-mean of each model.

The search procedure, as table I shows, selects the model 2 with the largest g-mean during the evaluation process. The selected model has 400 and 600 nodes in the hidden layers of the Generator and the Discriminator, respectively and latent space of dimension 1000. The last column of table I shows the g-mean accuracy on testing data for each model. From the same table, we observe that high values of the g-mean during the evaluation process correspond to models with better ability to detect fault occurrences. So we conclude that the stability of the unsupervised fault detection scheme is improved and that its performance is enhanced by the practice of the evaluation scheme for the selection of the final model.

Finally, we build an OCSVM model to examine in contrast with the GAN model. During the training of OCSVM model, we set the proportion of expected outliers in the training data to 0.01. We select this small value for the fraction of the outliers, to ensure small false alarm rates during the fault detection. Also, as kernel we use the radial basis function with parameter gamma fixed to 0.01. The training data are scaled to a fixed range between the values 0 and 1.

C. Testing Results of Selected Model

The fault detection rates and false alarm rates of the trained model and OCSVM algorithm on the TE process benchmark are shown in table II. We notice that in most cases the fault detection rates of the GAN model are higher than that given by using the OCSVM algorithm, causing higher average rate of fault detection of GAN over OCSVM. On the contrary, despite the fact that in most cases the rates of false alarms of OCSVM are lower than that of GAN model, their average values are at the same level. For comparison reasons, we calculate the g-means of both algorithms (table II) and we notice that the GAN model obtains higher level of g-mean values than the OCSVM algorithm.

Furthermore, we observe the difficulty of both models to recognize the faults occurrences for faults 3,9,15,19. This is expected due to the following reason. The abnormal signals of these faults have similar statistical characteristics with the corresponding signal of normal behavior [27], [28]. Therefore, we assume that the detection of these particular faults is problematic, especially in one class classification tasks.

VI. Conclusions & Future Work

In this paper we introduce the use of a Generative Adversarial Network for one class fault detection, a demanding and urgent problem. During the training of the GAN model, the Generator attempts to learn the distribution of normal data and so to generate samples similar to the real ones. On the other hand, the Discriminator tries to distinguish between real samples and produced ones, learning at the same time a decision function between normal and abnormal data. The Discriminator plays the role of the final judge. Also, we consider the problem of instability during training proposing the use of a search algorithm for selecting the most reliable model of a set of candidates. In the fitness evaluation of each model we have to consider the generating ability of the corresponding Generator. So, we also propose the use of an Autoencoder for the evaluation of the generating ability of each GAN model. The model is tested and compared with one class SVM on TE process benchmark with satisfactory results. Future work can be applied with the testing of the system on more benchmarks, checking at the same time the correlation of the proposed evaluation criteria of the competitive models during the search algorithm with their fault detection accuracy on unknown samples.

References

[1] S. X. Ding, *Model-based Fault Diagnosis Techniques: Design Schemes, Algorithms, and Tools*, 1st ed. Springer Publishing Company, Incorporated, 2008.

[2] F. Gustafsson, "Statistical signal processing approaches to fault detection," *Annual Reviews in Control*, vol. 31, no. 1, pp. 41 – 54, 2007.

[3] K. Chen, C. Huang, and H. Jinliang, "Fault detection, classification and location for transmission lines and distribution systems: A review on the methods," vol. 1, pp. 25–33, 04 2016.

[4] S. S. Khan and M. G. Madden, "One-class classification: Taxonomy of study and review of techniques," *CoRR*, vol. abs/1312.0049, 2013. [Online]. Available: http://arxiv.org/abs/1312.0049

[5] Y. Xiao, H. Wang, W. Xu, and J. Zhou, "Robust one-class svm for fault detection," *Chemometrics and Intelligent Laboratory Systems*, vol. 151, no. Supplement C, pp. 15 – 25, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169743915003056

[6] H. J. Shin, D.-H. Eom, and S.-S. Kim, "One-class support vector machines-an application in machine fault detection and classification," *Comput. Ind. Eng.*, vol. 48, no. 2, pp. 395–408, Mar. 2005. [Online]. Available: http://dx.doi.org/10.1016/j.cie.2005.01.009

[7] S. Yin, X. Zhu, and C. Jing, "Fault detection based on a robust one class support vector machine," *Neurocomputing*, vol. 145, no. Supplement C, pp. 263 – 268, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S092523121400681X

[8] L. Deng and D. Yu, "Deep learning: Methods and applications," *Found. Trends Signal Process.*, vol. 7, no. 3–4, pp. 197–387, Jun. 2014. [Online]. Available: http://dx.doi.org/10.1561/2000000039

[9] F. Lv, C. Wen, Z. Bao, and M. Liu, "Fault diagnosis based on deep learning," in *2016 American Control Conference (ACC)*, July 2016, pp. 6851–6856.

[10] E. Principi, F. Vesperini, S. Squartini, and F. Piazza, "Acoustic novelty detection with adversarial autoencoders," in *2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017*, 2017, pp. 3324–3330. [Online]. Available: https://doi.org/10.1109/IJCNN.2017.7966273

[11] E. Marchi, F. Vesperini, S. Squartini, and B. Schuller, "Deep recurrent neural network-based autoencoders for acoustic novelty detection," *Intell. Neuroscience*, vol. 2017, pp. 4–, Jan. 2017. [Online]. Available: https://doi.org/10.1155/2017/4694860

[12] T. Ince, S. Kiranyaz, L. Eren, M. Askar, and M. Gabbouj, "Real-time motor fault detection by 1d convolutional neural networks," vol. 63, 11 2016.

[13] J. Xu, H. Li, and S. Zhou, "An overview of deep generative models," *IETE Technical Review*, vol. 32, no. 2, pp. 131–139, 2015. [Online]. Available: http://dx.doi.org/10.1080/02564602.2014.987328

[14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680. [Online]. Available: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

[15] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *CoRR*, vol. abs/1511.06434, 2015. [Online]. Available: http://arxiv.org/abs/1511.06434

[16] I. J. Goodfellow, "NIPS 2016 tutorial: Generative adversarial networks," *CoRR*, vol. abs/1701.00160, 2017. [Online]. Available: http://arxiv.org/abs/1701.00160

[17] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *CoRR*, vol. abs/1606.03498, 2016. [Online]. Available: http://arxiv.org/abs/1606.03498

[18] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," *CoRR*, vol. abs/1703.05921, 2017. [Online]. Available: http://arxiv.org/abs/1703.05921

[19] T. White, "Sampling generative networks: Notes on a few effective techniques," *CoRR*, vol. abs/1609.04468, 2016. [Online]. Available: http://arxiv.org/abs/1609.04468

[20] C. K. Sønderby, J. Caballero, L. Theis, W. Shi, and F. Huszár, "Amortised MAP inference for image super-resolution," *CoRR*, vol. abs/1610.04490, 2016. [Online]. Available: http://arxiv.org/abs/1610.04490

[21] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CoRR*, vol. abs/1611.07004, 2016. [Online]. Available: http://arxiv.org/abs/1611.07004

[22] E. L. Denton, S. Chintala, A. Szlam, and R. Fergus, "Deep generative image models using a laplacian pyramid of adversarial networks," *CoRR*, vol. abs/1506.05751, 2015. [Online]. Available: http://arxiv.org/abs/1506.05751

[23] D. J. Im, C. D. Kim, H. Jiang, and R. Memisevic, "Generative adversarial metric," 2016.

[24] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504 – 507, 2006.

[25] J. Downs and E. Vogel, "A plant-wide industrial process control problem," *Computers & Chemical Engineering*, vol. 17, no. 3, pp. 245 – 255, 1993, industrial challenge problems in process control.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[27] Y. Zhang, "Enhanced statistical analysis of nonlinear processes using kpca, kica and svm," *Chemical Engineering Science*, vol. 64, no. 5, pp. 801 – 811, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0009250908005472

[28] J. Rigelsford, "Fault detection and diagnosis in industrial systems advanced textbooks in control and signal processing," *Industrial Robot: An International Journal*, vol. 28, no. 5, p. null, 2001. [Online]. Available: https://doi.org/10.1108/ir.2001.04928eae.003