Predictive maintenance based on anomaly detection using deep learning for air production unit in the railway industry

Narjes Davari
LIAAD-INESC TEC
Porto, Portugal
narjes.davari@inesctec.pt

Bruno Veloso
LIAAD-INESC TEC
University Portucalense, Portugal
bruno.m.veloso@inesctec.pt

Rita P. Ribeiro
LIAAD-INESC TEC
University of Porto, Portugal
rpribeiro@fc.up.pt

Pedro Mota Pereira
LIAAD-INESC TEC
Porto, Portugal
pm.pereira.mail@gmail.com

João Gama
LIAAD-INESC TEC
University of Porto, Portugal
jgama@fep.up.pt

2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA) | 978-1-6654-2099-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/DSAA53316.2021.9564181

***Abstract*—Predictive maintenance methods assist early detection of failures and errors in machinery before they reach critical stages. This study proposes a data-driven predictive maintenance framework for the air production unit (APU) system of a train of *Metro do Porto* by deep learning based on a sparse autoencoder (SAE) network that efficiently detects abnormal data and considerably reduces the false alarm rate. Several analog and digital sensors installed on the APU system allow the detection of behavioral changes and deviations from the normal pattern by analyzing the collected data. We implemented two versions of the SAE network in which we inputted analog sensors data and digital sensors data, and the experimental results show that the failures due to air leakage problems are predicted by analog sensors data while other types of failures are identified by digital sensors data. A low pass filter is applied to the output of the SAE network, and a sequence of abnormal data is used as an alarm for the APU system failure. Performance indicators of the SAE network with digital sensors data, in terms of F1 Score, Recall, and Precision, are respectively, about 33.6%, 42%, and 28% better than those of the SAE network with analog sensors data. For comparison purposes, we also implemented a variational autoencoder (VAE). The results show that SAE performance is better than that of VAE by 14%, 77%, and 37% respectively, for Recall, Precision and F1 Score.**

***Index Terms*—Failure detection, Predictive maintenance, Air production unit, Sparse autoencoder, Time series**

I. Introduction

There are several maintenance techniques for transportation vehicles, e.g., preventive maintenance, corrective, and condition-based maintenance. Preventive maintenance [1] is regularly performed through scheduled inspections where equipment is replaced or repaired based on a pre-specified schedule. This type of maintenance leads to a waste of resources in repairs or replacements of equipment while it is still working so that it does not break down unexpectedly, as well as time lost responding to emergencies and diagnosing faults. In corrective maintenance [2], one waits for a breakdown to occur and then the equipment is repaired. The condition-based maintenance is a maintenance strategy that monitors the actual condition of a system to determine what maintenance requirements to be done.

Predictive maintenance (PdM) [3] is a condition-based method that uses data analysis tools for assessment historical and real-time data from various parts of the system to detect anomalies and possible defects in equipment and then fix them before they lead to a system failure. Over the years, machine learning (more recently, deep learning) methods have been suggested for PdM. Failure prediction by deep learning methods (e.g., Deep Neural Network, Recurrent Neural Network, Convolution Neural Network, and Long Short Term Memory) estimates the probability of an equipment failing through automated learning of system past data.

Sparse autoencoders are one of the most powerful deep neural networks that have been successfully applied for failure detection. Autoencoders, as an unsupervised learning model, can automatically learn features from unlabeled data.

Recent literature review shows that the PdM methods can be classified into three main categories: model-based, knowledge-based, and data-driven approaches. Data-driven PdM approaches [4] detect failure and anomalies by analyzing the data collected from multiple sensors in real-time. The data-driven algorithms effectively fuse a large amount of running data from sensors for the prediction and detection of failure and have attracted wide attention in modern industrial systems, see, for example, [5]–[8].

In this paper, we implement a data-driven PdM framework based on sparse autoencoder to early detect/predict failures on an air production unit (APU) system of a train of *Metro do Porto*. This is a highly requested and critical system throughout

This research was carried out in the context of the project FailStopper (DSAIPA/DS /0086/2018) and by the grant CHIST-ERA-19-XAI-012, funded by Fundação para a Ciência e Tecnologia by the project XPM CHIST-ERA/0004/2019.

the operation of the vehicle and, in the absence of redundancy, its failure results in the immediate removal of the vehicle for repair. This impact not only the operating company, but, above all, the customers who see their expectations of confidence in transport undermined. Our goal is to identify normal/abnormal behaviors in the data stream obtained from a set of sensors installed in the APU system while the train is in operation. The objective is to predict if a failure is evolving using unsupervised methods based on deep learning.

The paper is organized as follows. An overview of the related work in the context of anomaly detection and predictive maintenance is provided in Section II. Section III describes our proposal for fault detection by an unsupervised learning approach. We detail our PdM framework, which involves the use of a sparse autoencoder trained in an online setting The case study, a problem definition, the feature generation, and anomaly detection are presented in Section IV. Section V presents the experimental results obtained by the SAEs with analog or digital sensors data, and finally the conclusion remarks are pointed out in Section VI.

II. Related work

Failures can be detected by finding patterns in data that do not correspond to normal system behavior and which represent anomalies. In the past decades, several anomaly detection approaches have been proposed for of failure prediction or early failure detection, e.g., [9], [10].

More specifically, and regarding railway industry, two recent literature surveys on the work related to different PdM methods can be found in [11] and [12]. Among all the proposed PdM methods for the railway industry, we are interested in data-driven based on learning methods. A recent work in [13] explores data-driven PdM based on anomaly and novelty detection implemented to predict failure in the automatic door system of the train and prevents the spread of breakdown in the system. The authors developed and implemented four common learning algorithms for anomaly detection. Moreover, the results show that a low-pass filter can significantly reduce the number of false alarms. Lee [14] used a logistic regression classifier to model the compressor behavior used for air leakage detection by anomaly detection in a train's braking pipes. Also, a density-based clustering method with a dynamic density threshold was used to distinguish anomalies from outliers and detect anomalies based on the severity degree.

More recently, Chen et al. [15] focus specifically on predicting compressor failures using a recurrent neural network using Long Short-Term Memory (LSTM) architecture. The authors compared their method and a random forest method where the experimental results show that predictions by LSTM stay significantly more stable over time, while in terms of AUC score random forest slightly outperforms the LSTM. Most recently, Barros et al. [16] developed a real-time data analysis of the sensors installed on APUs that detects anomalies. They also provided rules based on peak frequency analysis. They considered definition of normal and abnormal behavior of sensors data which can be used for APU failure detection.

III. Failure detection by Unsupervised Learning

In this section, we present our proposed deep learning method for failure detection in a timely manner through the analysis of sensor data without needing any external feedback.

A. Time Series Data

Since in most of the industry areas (e.g., transportation, power energy) the amount of data collected and analyzed are challenging to be manipulated and visualized, the use of big data analysis methods has considerably increased in recent years. In this application a massive amount of multivariate time-series data is available every day, thus it is convenient to extract suitable features from raw data. Time-series data are known as observations sequentially recorded over time. Multivariate time-series data are composed of two or more variables observed over a successive period of time.

B. Autoencoder

An autoencoder is an artificial neural network (NN) in which the number of neurons in input and output layers are the same, and the hidden layers, typically, have less neurons than the input layer. This type of neural network is trained to reconstruct the input values at the output layer. In fact, it maps the input data to a reconstruction of the input through a learning function. The encoder learns to compress a high-dimensional input to a low-dimensional latent space and the decoder then attempts to faithfully reconstruct the output with minimal error. The autoencoder (AE) works as an unsupervised learning method and can be used to extract the typical features of the raw data. In the basic structure of AE, the input is as

$$\mathbf{X}_i = x_{i1}, x_{i2}, ..., x_{im} \tag{1}$$

where $i = 1, \ldots, n$ and $j = 1, \ldots, m$ denote the sample numbers and the dimension of the sample, respectively. In particular, a deep autoencoder is a feed-forward multi-layer neural network that learns to reconstruct its inputs through a pair of encoding and decoding phases

$$\hat{\mathbf{X}} = \mathbf{D}(\mathbf{E}(\mathbf{X})) \tag{2}$$

where $\mathbf{X}$ and $\hat{\mathbf{X}}$ are the input data and the reconstructed input data, and the nonlinear functions $\mathbf{E}$ and $\mathbf{D}$ are encoding and decoding functions that map the input data to the hidden layer and the hidden layer to the output layer, respectively.

The basic AE minimizes the reconstruction error which is the divergence between the input data and its reconstructed one through training $\mathbf{E}$ and $\mathbf{D}$ functions. In particular, an autoencoder can be viewed as a solution to the following optimization problems:

$$\arg\min_{D,E} \|\mathbf{X} - \mathbf{D}(\mathbf{E}(\mathbf{X}))\| \tag{3}$$

where $\|\cdot\|$ is usually chosen to be the 2-norm. Since the AE extracts the meaningful features from raw data and reconstructs them at the output layer, it may simply copy the input to the output. The improved versions of the traditional AE

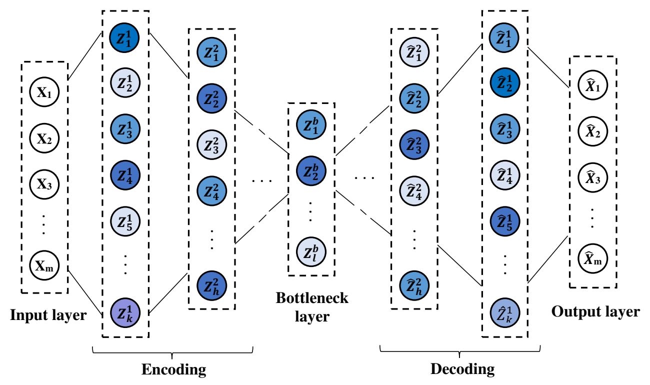

Fig. 1: The structure of a stacked sparse autoencoder. Darker nodes show larger activation values, while lighter nodes show lower activation values.

have been proposed to prevent this type of problem, such as de-noising AE [17], [18], sparse AE [19]–[21]. They produce more robust and more interesting features. Wen et al. [22] proposed a three-layer sparse AE to extract the features of raw data, and applied the maximum mean divergence value to minimizing the difference penalty between the features of training and testing data. Also, a lot of research (such as [23]–[25]) has been carried on the Sparse Autoencoder (SAE) for fault diagnosis. This paper presents the implementation of a SAE to find the anomalies on a time series data.

C. Sparse Autoencoder

Sparse autoencoder, as an extension of the auto-encoder, offers an alternative method and learns relatively sparse features by introducing a sparse penalty term into the autoencoder. For this purpose, a regularization item is added to the cost function for penalizing the weights. Unlike the traditional AE, in which the number of nodes at hidden layer(s) is less than that of input layer, SAE initially has a large number of neurons in its hidden layer(s), among which only a small number of neurons are allowed to be activated and trained to encode and decode, at the same time.

Note that network weights are normally regularized, and the level of activation of each neuron corresponds to its regularized weight. In the trained network, the individual neurons are activated; however, by changing the input to the network, their activation values also change; i.e., they are data-dependent.

Figure 1 shows the structure of a sparse autoencoder neural network in which the encoding, and decoding layers are symmetrical with respect to the bottleneck layer. Note that the number of active nodes in each hidden layer is changing over the training process. Furthermore, a sparse penalty term is applied on the hidden layer(s) to control the number of "active" neurons.

The general form of the measured data of sensors as a multivariate time-series data set can be written as

$$\mathbf{X} = \mathbf{X}_1, \mathbf{X}_2, .., \mathbf{X}_i, .., \mathbf{X}_n \tag{4}$$

where n is the number of samples (features) and $X_i \in \mathcal{R}^m$ is as shown in Equation (1). The data set $\mathbf{X}$ is inputted to the SAE network.

The input layer is mapped to the hidden layer by encoder part of the network through $E_\theta = f(\mathbf{W}^1 X_i + b^1)$, which in turn is mapped to output layer by the decoding part through $D_\theta = g(\mathbf{W}^2 E_\theta + b^2)$, where f and g are the activation functions of the encoder and decoder, respectively. For any sample from a data set, $\mathbf{X}_i$, the encoded input vector is $\mathbf{Y}_i = f(\mathbf{W}^1, b^1)\mathbf{X}_i$ and then the reconstructed vector of the input data is computed by $\hat{X}_i = g(\mathbf{W}^2, b^2)\mathbf{Y}_i$, where $\mathbf{W}$ denotes the weights between input-hidden layer and hidden-output, and b^1 and b^2 are biases. Then, the appropriate values of parameters $\theta_D = (\mathbf{W}^1, b^1)$ and $\theta_E = (\mathbf{W}^2, b^2)$ are obtained by minimizing the cost function of SAE:

$$J_{SAE} = \frac{1}{n}\sum_{j=1}^{n} \left\|\mathbf{X}_j - \hat{\mathbf{X}}_j\right\|^2 + \lambda\|\mathbf{W}\|_2^2 + \beta \sum_{j=1}^{m} KL(\rho\|\hat{\rho}_j) \tag{5}$$

where λ, β and ρ are the weight decay parameter to prevent overfitting, the sparsity penalty parameter that controls the weight of the sparsity penalty term, and the sparsity parameter, respectively. Note that $KL(\cdot)$, Kullback–Leibler divergence, is an extra penalty term added to the cost function that penalizes the desired distribution $\hat{\rho}$ deviating significantly from actual distribution ρ, and can control the number of "active" neurons. Moreover, to create a sparse hidden layer, most of the nodes in this layer need to be inactive. Therefore, a sparsity constraint imposed on the hidden nodes as given by

$$\hat{\rho}_j = \frac{1}{m}\sum_{i=1}^{m} \mathbf{a}_j^{(l)}(x_i) \tag{6}$$

where $\hat{\rho}_j$, $\mathbf{a}_j^l$ and x_i are the average activation of hidden node j, the activation (output) of unit j in layer l of the network and i^{th} input to unit j in layer l, respectively.

D. Online Learning

In this particular online learning procedure, the SAE is trained in a predetermined time window and predicts the anomaly scores of the examples in a subsequent test time window. The learning procedure is repeated once a new test data set is available. In the training phase, only normal data set is inputted to the network. Then, in the testing phase, a well-trained network distinguish abnormal data by a high anomaly score. Beforehand, a set of parameters need to be determined, namely, the number of neurons (according to the number of features), number of hidden layers and the neurons in which, number of epochs, batch size, and sparse parameter.

At the beginning of the learning procedure, weights and biases are initialized randomly, while for the next training - test cycles the weights and biases are borrowed from the previous cycle. More specifically, at t^{th} cycle of the learning procedure, a time window of normal data, with length $T_{training}$, from T_s^t to T_e^t (see, Figure 2) is considered as the network input. Then, the SAE is trained by minimizing the reconstruction error (Equation (5)) through back-propagation [26], which

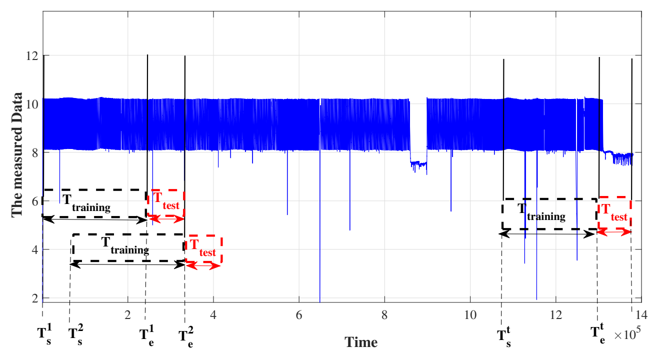

Fig. 2: Data over time with training and testing windows

is a conventional training algorithm in deep learning, setting the target values to be equal to the inputs. It consists of two steps: forward and backward propagation. By inputting training data, the forward propagation is executed where all the activations through out the network, including the output value of the hypothesis, are computed. Then, for an output node, the difference between the network's output and the true value is calculated as the reconstruction error. In the backward propagation step, the gradient of the cost function is used to update the weights and biases. In the testing phase, a time window with length T_{test} (from T_e^t to $T_e^t + T_{test}$) of data is inputted to the network where it can discriminate the abnormal data by a high anomaly score, which then can be used to identify possible failures (see Section III-E).

Whenever a new test data set is available, the online learning procedure is repeated by moving forward the training time window for T_{test} time stamps, i.e., from $T_s^{t+1} = T_s^t + T_{test}$ to $T_e^{t+1} = T_e^t + T_{test}$, and excluding the anomalies detected in the previous test data set (i.e., from T_e^t to $T_e^t + T_{test}$). The new learned SAE is then used to detect anomalies in the new test data set (from T_e^{t+1} to $T_e^{t+1} + T_{test}$), and so on. The algorithm for one training - test cycle is summarized in Algorithm 1.

Algorithm 1
SAE for online anomaly detection: training - test cycle.

```
 1: Input:
      Parameters of input data:
        training data set $\mathbf{X}$ and test data set $\mathbf{Y}$,
        window sizes $T_{training}$ and $T_{test}$
      Parameters of network:
        nr_features, nr_epochs, batch_size,
        activation functions $f$ and $g$,
        parameters $\lambda$, $\beta$, $\rho$
      Parameters of LPF:
        smoothing factor $\alpha$.
      // Pre-processing data:
 2: Extract appropriate features from $\mathbf{X}$ with length $T_{training}$
      // Training of SAE:
 3: Initialize SAE parameters: weights and biases.
 4: for $k = 1$ to nr_epochs do
      // Forward propagation:
 5:   $\hat{\mathbf{X}} = \mathbf{D}(\mathbf{E}(\mathbf{X}))$ (cf. Eq. 2)
 6:   Compute $\|\mathbf{X} - \hat{\mathbf{X}}\|$
 7:   Compute $J_{SAE}$ (cf. Eq. 5)
      // Backward propagation:
 8:   Update SAE parameters by backpropagating the error
 9:   Minimize $J_{SAE}$ using Adam [27] algorithm
10: end for
      // Predict:
11: Predict outputs for training data with the SAE ($\hat{\mathbf{X}}$)
12: Compute reconstruction error $\mathbf{e}_r = |\mathbf{X} - \hat{\mathbf{X}}|$
13: Determine $thr$ using the boxplot on $\mathbf{e}_r$
      // Testing of SAE:
14: Extract appropriate features from $\mathbf{Y}$ with length $T_{test}$
15: Predict outputs for test data with the SAE ($\hat{\mathbf{Y}}$)
16: Compute reconstruction error $\mathbf{e}_r = |\mathbf{Y} - \hat{\mathbf{Y}}|$
17: Apply LPF to $\mathbf{e}_r$
18: Compute $\bar{\mathbf{e}}_r$ thresholding the filtered $\mathbf{e}_r$ (cf. Eq. 7)
19: Output: Detected anomalies on test data
```

E. Predictive Maintenance Framework

The general procedure of the failure detection framework is shown in Figure 3, in which the input vector (X) comprises extracted features of data set, which are used to train the network and to test it. Note that before applying the extracted features in SAE they are standardized by StandardScaler() function that is available in Sklearn library python.

The learned SAE predicts the value of the input matrix ($\hat{X}$), and the mean square of the reconstruction errors ($e_r = X - \hat{X}$) are computed. Next, a Boxplot analysis is needed to obtain a threshold value (thr) with which normal data are labeled 1 and abnormal data are labeled 0, denoted as $\bar{e}_r$ (see Equation (7)).

$$\bar{e}_r = \begin{cases} 1 & if \quad e_r < thr \\ 0 & if \quad e_r \geq thr \end{cases} \tag{7}$$

In practice, the large number of false alarms is a major challenge for many anomaly detection algorithms which can be mitigated by using filtering the outputs of a network. For instance, [13] observed that using a low-pass filter (LPF) significantly reduces the amount of the false alarms. In this work, the labeled data set is inputted to a LPF whose output is analyzed to detect failures; i.e., if a set of consecutive data are below a predetermined threshold it is a sign of failure.

IV. Case Study

The goal of this paper is to develop a data-driven predictive maintenance system that issues an alert whenever an APU system of a specific train of *Metro do Porto* is predicted to suffer a failure. In this study, we focus on the behavior of several digital and analog sensors installed on the train. The experimental data set used in this paper was collected from March to July 2020, where 16 signals (e.g., pressures, motor current, electrical signal of air intake valve) are logged at 1Hz frequency by an on-board embedded device. The data collected from the sensors are transferred to the server using the TCP/IP protocol application every five minutes.

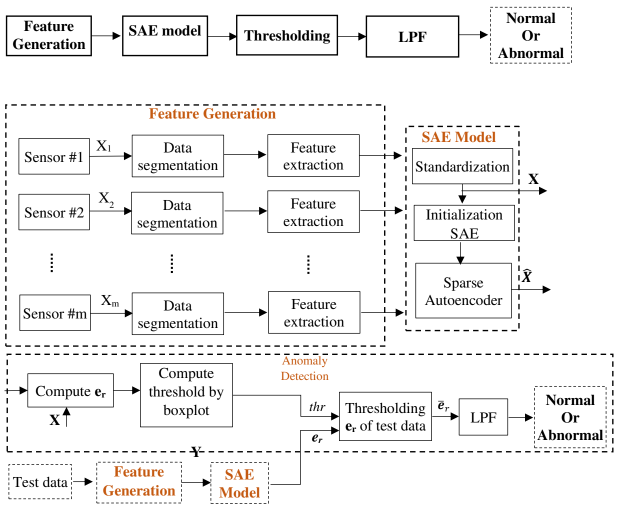

Fig. 3: A schematic view of data-driven anomaly detection.

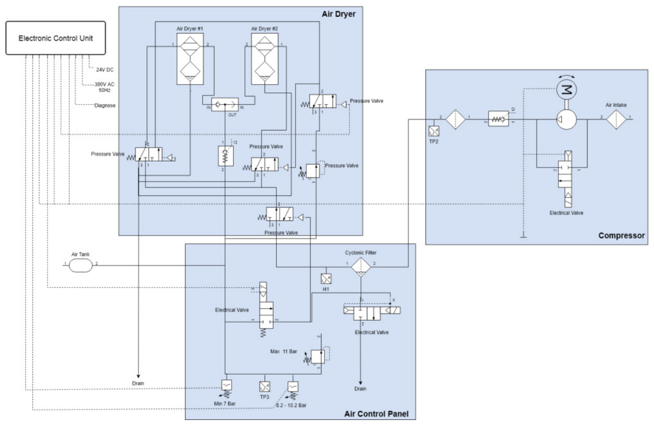

Fig. 4: The Air Production Unit system with the position of the main sensors.

A. Problem Definition

Air Production Unit, as one of most important parts of compressor, converts the power from an electric motor into kinetic energy through compressing and pressurizing air. The compressor pumps air into the braking pipe which is consumed by actions such as braking, door opening and closing and bioreactor usage. Therefore, the performance of APU is very critical and if it fails (due to breaks down) the equipment receive the compressed air fail, subsequently.

Here, we consider data from a set of analog and digital sensors installed on an APU system that can be used to detect real-time changes in performance of APU. While the data collected by analog sensors are real values, those collected by digital sensors are binary values. The APU schema with the position of the sensors is described in Figure 4, and the sensors are listed as

Analog sensors:

- measure the pressure generated in the pneumatic panel (TP3),
- measure the current of one phase of the three-phase motor (Motor-Current)

Digital sensors:

- measure the electrical signal of air intake valve on compressor (COMP)
- measure the electrical signal of the solenoid valve (DV Electric)
- indicate which tower is drying the air and which tower is draining the humidity removed from the air (TOWERS)
- activate the COMP intake valve to start the compressor under load (MPG)
- indicate when the pressure is lower than 7 bars (LPS)
- detect the discharge in the air drying towers (Pressure-Switch)
- count the absolute amount of air passing from the APU to the reservoirs (Caudal)

B. Feature Generation

In this study, we are dealing with several sensors with a sampling frequency of 1Hz; i.e., in a time duration of about 24 hours a large volume of data is available as input to a classification algorithm. Therefore, we need to create a method for representing and extracting useful information from this big data set.

For that purpose, the data set is segmented into intervals of the “Compressor Run Time”, T_{run}, and “Compressor Idle Time”, T_{idle}. The former is the time required for a compressor to pump air inside the main reservoir while during the latter interval the compressor is switched off and the braking pipe consumes air in the main reservoir. Since the Run Time and the Idle Time change over time, so there are the segmented data of different lengths. For example, for the i^{th} segment there is an input vector, as given in Equation (1). Following the segmentation step, seven features are extracted for two consecutive T_{run} and T_{idle} intervals. Then, the set of 14 features extracted for the two analog sensors in addition to T_{run} and T_{idle} form an input vector for the SAE network. The procedure is described in the following three steps:

1) Search for finding T_{run} and T_{idle} of compressor for each train. (Note that the range and distribution of T_{run} and T_{idle} change over the time due to differences in configuration, age, operational usage, and failures.)
2) Time intervals T_{run} and T_{idle} are segmented into two and five equal-length time intervals, respectively, each following on [13] is called a “bin”.
3) The mean value of time series data in each bin is computed and its value is multiplied by the cycle duration which is sum of T_{run} and T_{idle}.

For instance, the pressure generated in the pneumatic panel, measured by TP3 sensor, is showed in Figure 5 which is segmented into seven bins ($B_i, i = 1, \ldots, 7$). Note that the slope of the curve in T_{run} and T_{idle} indicate the speed of air generation and the speed of air consumption. On the other hand, for the data collected by digital sensors, the features are created by enumerating the ones in each cycle duration (i.e., $T_{run} + T_{idle}$), and the seven features (of seven sensors) ($N_j, j = 1, \ldots, 7$) in addition to T_{run} and T_{idle} form an input vector for the SAE network.

The features extracted from analog and sensors data are summarised in Table I, in which B^{TP3}, B^{MC} and $\mathbf{N}_j$ are the bins for TP3 data, the bins for Motor-current data and the number of ones for j^{th} digital sensors data, respectively. The number of input features to the SAE network which extracted from analog and digital sensors data are 16 and 9, respectively.

TABLE I: The input features to the SAE network in one cycle duration

Sensor	Input Features	Description
Analog	$\mathbf{B}_i^{TP3},\ i = 1, \ldots, 7,$	7 bins for TP3 sensor data
	$\mathbf{B}_i^{MC},\ i = 1, \ldots, 7,$	7 bins for Motor-current data
	T_{run} and T_{idle}	Run Time and the Idle Time of compressor
Digital	$\mathbf{N}_j,\ j = 1, \ldots, 7,$	Number of ones for 7 digital sensors
	T_{run} and T_{idle}	Run Time and the Idle Time of compressor

C. Anomaly Detection

Anomalies are those data significantly deviate from the normal pattern. In the proposed method, the reconstruction errors (e_r) of 16 features for analog sensors and nine features for digital sensors are computed, which in turn are used to compute mean square error (MSE) for each cycle duration.

Then, the MSE values obtained for the training data set are used to calculate a threshold by the boxplot analysis. The threshold considered is the one set for extreme anomalies, i.e. $(Q3 + 3 \ast IQR)$, where $IQR = Q3 - Q1$ is the interquartile range, and $Q1$ and $Q3$ are the first and the third quartiles of the MSE values, respectively. The threshold is then used for labeling each cycle duration (either normal or abnormal) if its MSE is larger than the threshold.

As explained in Section III-E, the output of the SAE network is inputted into a LPF to be used for failure detection.

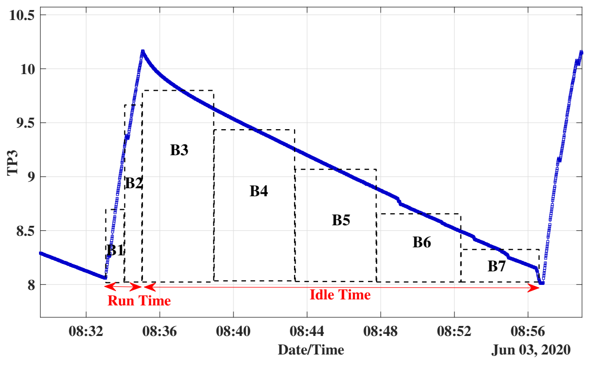

Fig. 5: The pressure generated in the pneumatic panel, measured by TP3, segmented into two time intervals and seven bins.

TABLE II: APU failures reported by the expert from April to July 2020.

Nr.	Start Time	End Time	dur.(min)	severity
#1	4/12/2020 11:50	4/12/2020 23:30	700	high
#2	4/18/2020 00:00	4/18/2020 23:59	1440	high
#3	4/19/2020 00:00	4/19/2020 01:30	90	high
#4	4/29/2020 03:20	4/29/2020 04:00	40	high
#5	4/29/2020 22:00	4/29/2020 22:20	20	high
#6	5/13/2020 14:00	5/13/2020 23:59	599	high
#7	5/18/2020 05:00	5/18/2020 05:30	30	high
#8	5/19/2020 10:10	5/19/2020 11:00	50	high
#9	5/19/2020 22:10	5/19/2020 23:59	109	high
#10	5/20/2020 00:00	5/20/2020 20:00	1200	high
#11	5/23/2020 09:50	5/23/2020 10:10	20	high
#12	5/29/2020 23:30	5/29/2020 23:59	29	high
#13	5/30/2020 00:00	5/30/2020 06:00	360	high
#14	6/01/2020 15:00	6/01/2020 15:40	40	high
#15	6/03/2020 10:00	6/03/2020 11:00	60	high
#16	6/05/2020 10:00	6/05/2020 23:59	839	high
#17	6/06/2020 00:00	6/06/2020 23:59	1439	high
#18	6/07/2020 00:00	6/07/2020 14:30	870	high
#19	7/08/2020 17:30	7/08/2020 19:00	90	high
#20	7/15/2020 14:30	7/15/2020 19:00	270	medium
#21	7/17/2020 04:30	7/17/2020 05:30	60	high

In this work, we use a simple LPF algorithm with the recursive equation $y_i = y_{i-1} + \alpha\ (x_i - y_{i-1})$, where y_i, x_i and α are the filter output, the original data in instant i, and the smoothing parameter, respectively. We consider a consecutive set of anomalies as a sign of a failure.

V. Experimental Results

The experiments were carried out using a computer that has the following specification: Intel Core i7-10510U, 1.80 GHz, and 16 GB RAM. The algorithm was implemented in Python as an object-oriented, high-level programming language.

The dataset is available from March to July 2020. We used data available for March 2020 for the initial training of the SAE network which then is tested to predict the failures in the first week of April 2020. Next, the training is repeated by moving one week forward the training time window and excluding the anomalies detected in the first week of April 2020, and so on. The rationale here is that if the model is properly trained with normally distributed data, it is expected that it can detect anomalies when data with normal and abnormal distributions are applied to it.

In addition to the predictions made by the proposed algorithm, we also have failure reports provided by an expert which makes it possible to evaluate true and false positive and false negative alarms. The failures reported from April to July 2020 by the expert are summarized in Table II. Although labelled data are available for a very short time period, due to the fact that the algorithm needs to be implemented in a real-time application, we intended to use an unsupervised learning algorithm. Note that a failure must be predicted at least two hours before it takes place.

For the evaluation purposes, we experimentally studied and tuned different SAE settings, analyzed the impacts of different LPF smoothing factors on the number of false positive alarms, compared the performance of SAE with analog sensors data versus the SAE with digital sensors data. Finally, we also compared the performance of the SAE against that of a variational autoencoder (VAE) [28] working with data measured by analog sensors and digital sensors.

A. Algorithm Settings

There is no comprehensive and complete method to obtain the appropriate number of hidden layers and neurons in a model network. However, the network topology needs to be consistent with the experimental settings. We examined several structures for the network and selected the one leads to an optimal performance in the learning and prediction stages.The parameters are summarized in Table III, where $Analog$ and $Digital$ refer to network settings for SAEs work with analog and digital sensors data, respectively. For the SAE works with analog sensors data, the encoder has three hidden layers with 128, 64 and 32 neurons, the bottleneck layer has 12 neurons, and the decoder has three hidden layers with 32, 64 and 128 neurons. Moreover, the encoder of SAE works with digital sensor data has two hidden layers with 36 and 18 neurons, the bottleneck layer has 6 neurons and the encoder has two hidden layers with 18 and 36 neurons. The batch size is relatively small for a faster model convergence.

TABLE III: Parameters of SAE network

Parameter	$Analog$	$Digital$
Nodes in input layer	16	9
Neurons in 1st hidden layer	128	36
Neurons in 2nd hidden layer	64	18
Neurons in 3rd hidden layer	32	-
Neurons in Bottleneck layer	12	6
β	5	6
λ	$1e^{-5}$	$2e^{-5}$
ρ	0.01	0.05
Batch size	30	40
Number of epochs	100	100

B. Impact of Low-Pass Filter Smoothing Factor

The output of the autoencoder is filtered by a low-pass filter. The LPF filters high-frequency variations of the input signal and transforms it into a smooth sequence of outputs. The LPF smoothing factor plays an obvious role on the overall performance of the algorithm. To evaluate the impacts of LPF smoothing factor (α), we repeated an experiment for ten different values between 0.01 and 0.1 and compare the number of false positive alarms as presented in Figure 6 for SAEs work with analog sensors data and digital sensors data. Generally, the number of false positive alarms generated by SAE with analog sensors data is higher than that generated by SAE with digital sensors for almost all the experiments. Moreover, for SAE with analog sensors data the lowest false positive alarms is obtained when $\alpha = 0.04$, while for the SAE with digital sensors data it happens when $\alpha = 0.02$.

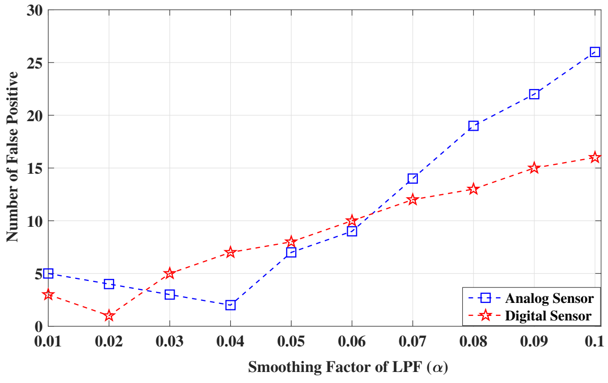

Fig. 6: The number of False Positives (SAE) for different values of the smoothing factor of LPF

C. Comparison of SAE and VAE with Analog vs. Digital Sensors Data

In this section, the normality and abnormality of a cycle duration are investigated. A data point is classified as an anomaly if the value of LPF output is below a specific threshold which is empirically set to 0.3 and 0.5 for analog and digital data set, respectively; i.e., a cycle duration is predicted as abnormal if its LPF output value is less than 0.3. The normality/abnormality overtime for the SAEs works with analog. Digital sensors data are presented in Figures 7 and 8, respectively, in which the x and y axes, respectively, represent date/time and the value of normality that changes between one (normal) and zero (abnormal). Detecting a single cycle duration is insufficient to conclude a persistent failure (it may generate a high number of false-positive alarms). Therefore, we consider a sequence of abnormal cycle duration as a sign of failure (i.e., a positive alarm).

In Figures 7 and 8, the areas marked with black dashed rectangles show the time duration when a real failure occurred, according to the data reported in Table II (true positive), the orange dashed rectangles indicate the false alarms (false positive) and the green dashed rectangles show the real failures that are incorrectly detected as normal data (false negative).

With reference to the actual failures reported in Table II, it can be seen that the SAE network with analog sensors data predicts seven failures (#7-#11, #20, #21), while the network with digital sensors data predicts ten failures (#7-#13, #19-#21). Analog data analysis fails to predict three of the actual failures of APU system that are, indeed, predicted by digital sensors data, i.e., failures #12, #13, and #19.

Figures 9 and 10 show the normality/abnormality over time for the VAEs with analog and digital sensors data, respectively; in which the x and y axes, respectively, represent date/time and the value of normality that changes between one (normal) and zero (abnormal).

Figures 11 and 12 show the analog and digital sensors data sets in the time window where the failures #12 and #13 occurred, respectively. We can see that the SAE network predicts the failures due to air leakage by analysis analog

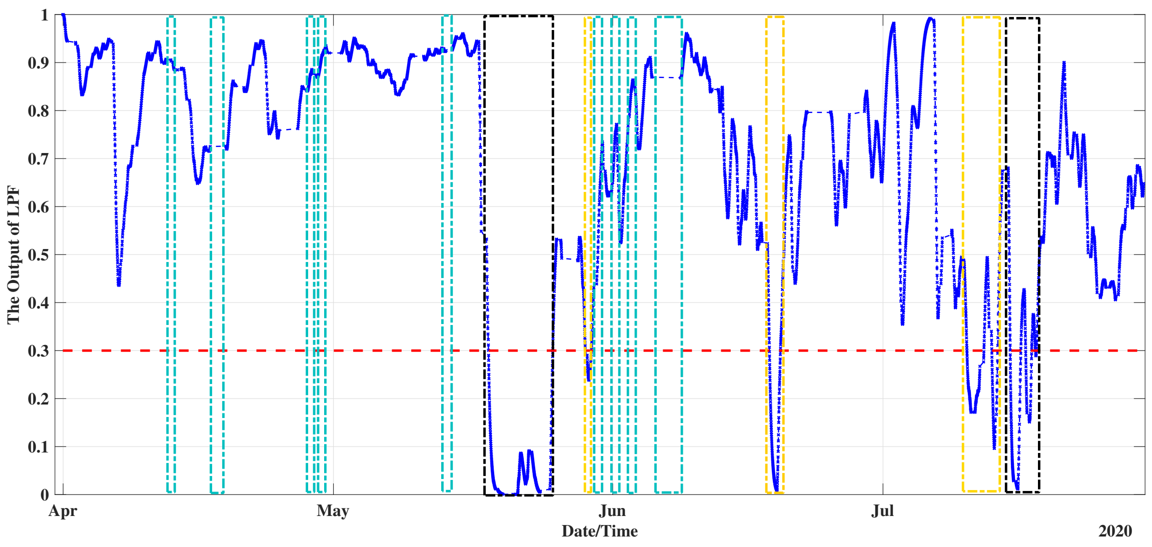

Fig. 7: The output of LPF over time for analog sensors data inputted to SAE; data below 0.3 are predicted as abnormal. The dashed rectangles indicate true positives (black), false positives (orange), false negatives (green).

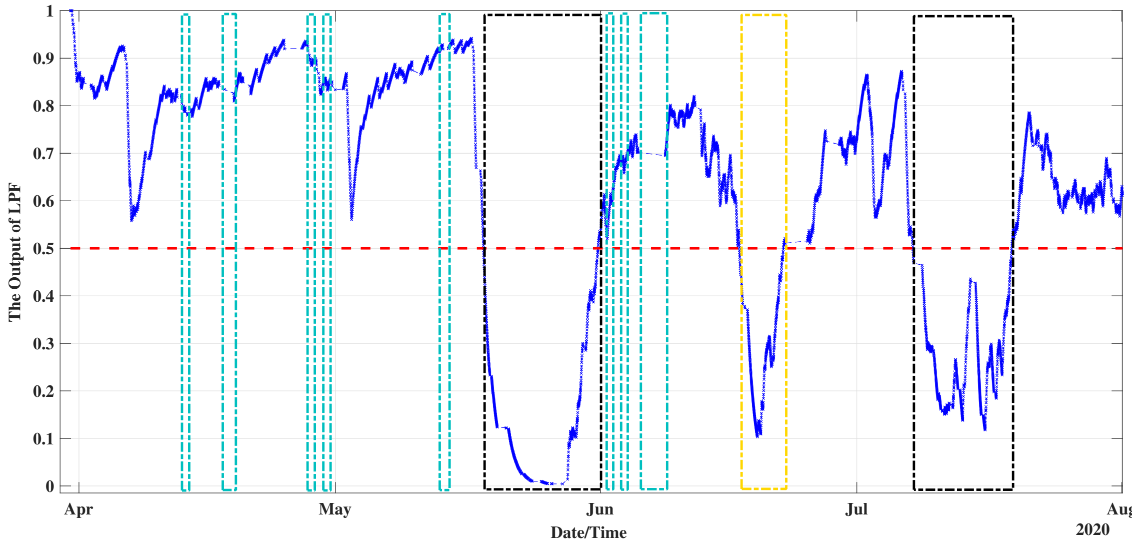

Fig. 8: The output of LPF over time for digital sensors data inputted to SAE; data below 0.5 are predicted as abnormal. The dashed rectangles indicate true positives (black), false positives (orange), false negatives (green).

sensors data, while other types of failures can be predicted by analyzing digital sensors data.

To effectively evaluate the performance of SAE and VAE algorithms, some metrics including Recall, Precision, and $F1$ Score(%) are applied and are defined as follows:

$$Recall(\%) = \frac{TP}{TP + FN} \times 100 \tag{8}$$

$$Precision(\%) = \frac{TP}{TP + FP} \times 100 \tag{9}$$

$$F1\ Score(\%) = \frac{2 * Precision * Recall}{Precision + Recall} \tag{10}$$

where TP, FP and FN represent the true positives, false positives and false negatives, respectively. Detailed comparison between various performance indicators for the SAE and VAE networks are reported in Table IV. Both SAE and VAE with digital sensor data obtain higher Recall, Precision, and F1 Score compared to those that use analog sensors data (respectively, about 44%, and 13%, and 32% for the SAE and about 50%, and 9%, and 29% for the VAE), which allow us to conclude that networks with digital sensor data outperform those with analog sensor data.

VI. Conclusions

In this paper, a data-driven predictive maintenance algorithm based on anomaly detection using a deep learning method is implemented to predict and detect abnormal data. We applied two sparse autoencoder (SAE) networks to combine the features extracted from data collected by analog and digital

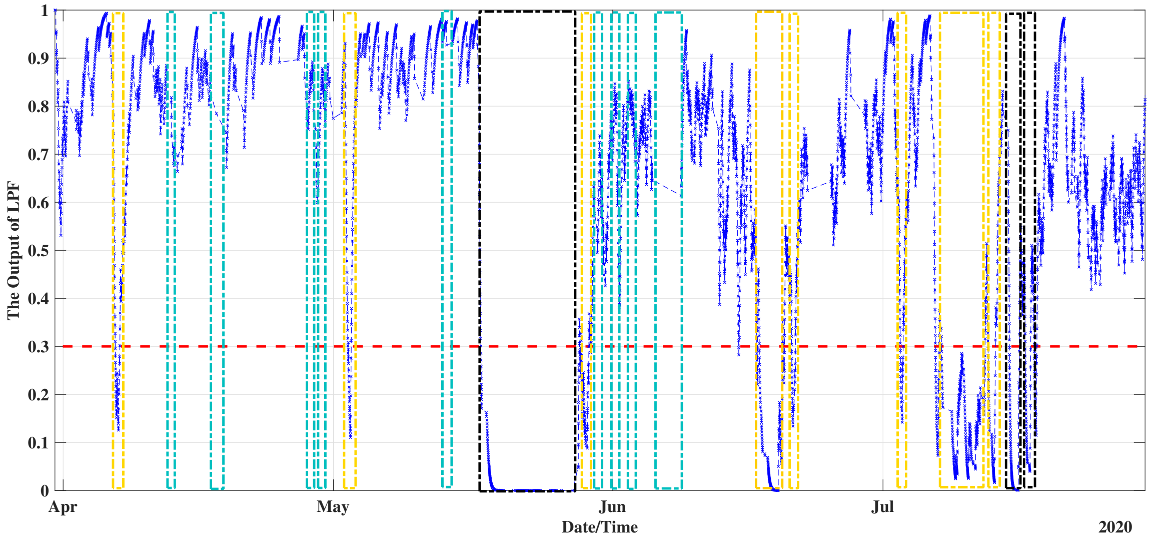

Fig. 9: The output of LPF over the time for analog sensors data inputted to VAE; data below 0.5 are predicted as abnormal. The dashed rectangles indicate true positives (black), false positives (orange), false negatives (green).

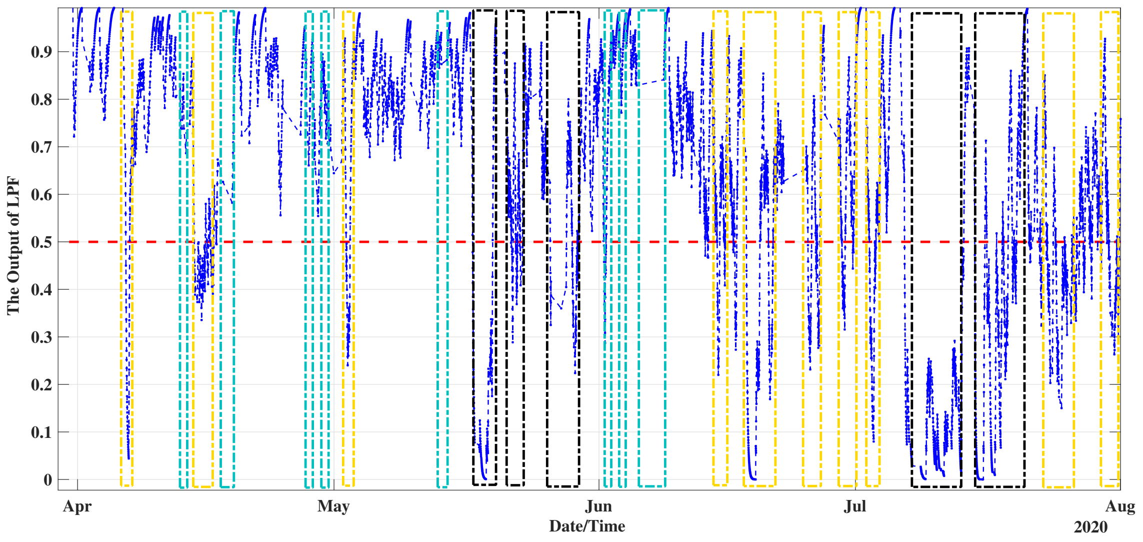

Fig. 10: The output of LPF over the time for digital sensors data inputted to VAE; data below 0.5 are predicted as abnormal. The dashed rectangles indicate true positives (black), false positives (orange), false negatives (green).

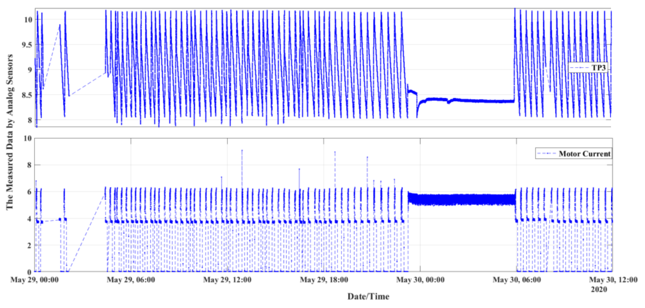

Fig. 11: Analog sensors, TP3 and Motor-current, data in the time window where the failures #12 and #13 occurred.

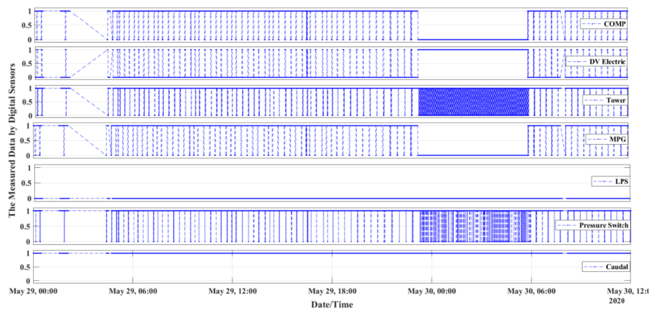

Fig. 12: Digital sensors data in the time window where the failures #12 and #13 occurred.

TABLE IV: Performance comparison of SAE and VAE

Metric	SAE		VAE	
	Analog	Digital	Analog	Digital
TP	7	10	6	9
FP	3	1	8	10
FN	14	11	15	12
$Recall(\%)$	33	47	28	42
$Precision(\%)$	70	90	43	47
$F1\ Score(\%)$	45	62	34	44

sensors installed in the APU system of a train for the period of first of March to the end of July 2020. The features inputted to the network are extracted from analog sensors data, namely: 14 bin values obtained from TP3 sensor and motor-current data (in one cycle duration), Idle Time, and Run Time of compressor. The features extracted from digital sensors data are seven values generated by enumerating the ones in each cycle duration for each digital sensor as well as the Idle Time and Run Time of compressor. The SAE uses self-training, and does not need external feedback; thus, it can work finely online, an essential factor for real-time predictive maintenance applications. Since detecting a specific abnormal data is not adequate to conclude a persistent failure, the network generates an alarm when a sequence of abnormal data is predicted. The output of SAE network is inputted to a low-pass filter through which the sudden variations are removed which in turn decrease the number of false alarms. The experimental results for the two SAEs show that the failures due to air leakage problem are predicted by analog sensors data while other type of failures are identified by digital sensors data. Finally, the comparison results of SAE and VAE show that SAE performance is better than that of VAE by 14%, 77%, and 37% respectively, for Recall, Precision and F1 Score.

Acknowledgment

This work is financed by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within project UIDB/50014/2020.

References

[1] G. Budai, D. Huisman, and R. Dekker, "Scheduling preventive railway maintenance activities," *Journal of the Operational Research Society*, vol. 57, no. 9, pp. 1035–1044, 2006.

[2] C. Stenström, P. Norrbin, A. Parida, and U. Kumar, "Preventive and corrective maintenance–cost comparison and cost–benefit analysis," *Structure and Infrastructure Engineering*, vol. 12, no. 5, pp. 603–617, 2016.

[3] J. Yan, Y. Meng, L. Lu, and L. Li, "Industrial big data in an industry 4.0 environment: Challenges, schemes, and applications for predictive maintenance," *IEEE Access*, vol. 5, pp. 23 484–23 491, 2017.

[4] W. Zhang, D. Yang, and H. Wang, "Data-driven methods for predictive maintenance of industrial equipment: a survey," *IEEE Systems Journal*, vol. 13, no. 3, pp. 2213–2227, 2019.

[5] K. Wang, "Intelligent predictive maintenance (ipdm) system–industry 4.0 scenario," *WIT Transactions on Engineering Sciences*, vol. 113, pp. 259–268, 2016.

[6] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao, "Deep learning and its applications to machine health monitoring," *Mechanical Systems and Signal Processing*, vol. 115, pp. 213–237, 2019.

[7] G. Li, C. Deng, J. Wu, X. Xu, X. Shao, and Y. Wang, "Sensor data-driven bearing fault diagnosis based on deep convolutional neural networks and s-transform," *Sensors*, vol. 19, no. 12, p. 2750, 2019.

[8] Y. Jiang and S. Yin, "Recursive total principle component regression based fault detection and its application to vehicular cyber-physical systems," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 4, pp. 1415–1423, 2017.

[9] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[10] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019.

[11] E. Fumeo, L. Oneto, and D. Anguita, "Condition based maintenance in railway transportation systems based on big data streaming analysis," *Procedia Computer Science*, vol. 53, pp. 437–446, 2015.

[12] P. C. L. Gerum, A. Altay, and M. Baykal-Gürsoy, "Data-driven predictive maintenance scheduling policies for railways," *Transportation Research Part C: Emerging Technologies*, vol. 107, pp. 137–154, 2019.

[13] R. P. Ribeiro, P. Pereira, and J. Gama, "Sequential anomalies: a study in the railway industry," *Machine Learning*, vol. 105, no. 1, pp. 127–153, 2016.

[14] W.-j. Lee, "Anomaly detection and severity prediction of air leakage in train braking pipes," *International Journal of Prognostics and Health Management*, vol. 21, 2017.

[15] K. Chen, S. Pashami, Y. Fan, and S. Nowaczyk, "Predicting air compressor failures using long short term memory networks," in *EPIA Conference on Artificial Intelligence*. Springer, 2019, pp. 596–609.

[16] M. Barros, B. Veloso, P. M. Pereira, R. P. Ribeiro, and J. Gama, "Failure detection of an air production unit in operational context," in *IoT Streams for Data-Driven Predictive Maintenance and IoT, Edge, and Mobile for Embedded Machine Learning*. Springer, 2020, pp. 61–74.

[17] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, and L. Bottou, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." *Journal of machine learning research*, vol. 11, no. 12, 2010.

[18] T. Tagawa, Y. Tadokoro, and T. Yairi, "Structured denoising autoencoder for fault detection and analysis," in *Asian Conference on Machine Learning*. PMLR, 2015, pp. 96–111.

[19] A. Ng *et al.*, "Sparse autoencoder," *CS294A Lecture notes*, vol. 72, no. 2011, pp. 1–19, 2011.

[20] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, "On optimization methods for deep learning," in *ICML*, 2011.

[21] A. Lemme, R. F. Reinhart, and J. J. Steil, "Online learning and generalization of parts-based image representations by non-negative sparse autoencoders," *Neural Networks*, vol. 33, pp. 194–203, 2012.

[22] L. Wen, L. Gao, and X. Li, "A new deep transfer learning based on sparse auto-encoder for fault diagnosis," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 1, pp. 136–144, 2017.

[23] R. Memisevic, "Gradient-based learning of higher-order image features," in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 1591–1598.

[24] W. Sun, S. Shao, R. Zhao, R. Yan, X. Zhang, and X. Chen, "A sparse auto-encoder-based deep neural network approach for induction motor faults classification," *Measurement*, vol. 89, pp. 171–178, 2016.

[25] F. Jia, Y. Lei, J. Lin, X. Zhou, and N. Lu, "Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data," *Mechanical Systems and Signal Processing*, vol. 72, pp. 303–315, 2016.

[26] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1, no. 2.

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[28] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE*, vol. 2, no. 1, pp. 1–18, 2015.