








Data-Driven Predictive Maintenance in Evolving Environments: A Comparison Between Machine Learning and Deep Learning for Novelty Detection

Francesco Del Buono¹ , Francesca Calabrese² , Andrea Baraldi¹ ,
Matteo Paganelli¹ , and Alberto Regattieri² 

¹ “Enzo Ferrari” Department of Engineering, UNIMORE, 41125 Modena, Italy
{francesco.delbuono, andrea.baraldi96,
matteo.paganelli}@unimore.it

² Department of Industrial Engineering (DIN), University of Bologna, 40136 Bologna, Italy
{francesca.calabrese9, alberto.regattieri}@unibo.it

Abstract. Predictive Maintenance (PdM) is the newest strategy for maintenance management in industrial contexts. It aims to predict the occurrence of a failure to minimize unexpected downtimes of equipment and maximize the useful life of the monitored components. In a data-driven approach, PdM makes use of Machine Learning (ML) algorithms to extract relevant features from historical signals, identify and classify possible faults (diagnostics), and predict the components’ remaining useful life (RUL) (prognostics). The major challenge lies in the high complexity of industrial plants, where both operational and environmental conditions change over time and a large number of unknown a priori modes may occur. A solution to this problem is offered by novelty detection, where a representation of the normal operating state of the machinery is learned and compared with online measurements in order to identify new operating conditions. In this paper, a comparison between ML and Deep Learning (DL) methods for novelty detection is conducted, to evaluate their effectiveness and efficiency in different scenarios. To this purpose, a case study considering vibration data collected from an experimental platform is carried out. Results show the superiority of DL on traditional ML methods in all the evaluated scenarios.

Keywords: Predictive maintenance · Novelty detection · Deep learning

1 Introduction

As one of the pillars of the Industrial 4.0 paradigm, predictive maintenance (PdM) is attracting researchers and practitioners of several industrial sectors. PdM allows performing maintenance interventions before a failure takes place and maximizing the useful lives of production [1]. Thanks to enabling technologies like IIoT and edge computing, it is possible to collect a large amount of data from online condition monitoring systems in order to assess the health condition of machinery at any point in time [2].

The transformation of raw data in useful knowledge supporting the decision making process in the context of PdM is usually referred to as Prognostics and Health Management (PHM). This process mainly consists of feature extraction, fault detection and diagnosis, and prognostics [3]. A data-driven PHM approach makes use of condition monitoring techniques and machine learning (ML) algorithms to perform fault diagnostics and prognostics [4]. However, their industrial applicability is limited by the fact that the training data available to build diagnostic models typically do not include all the work conditions that components and systems may experience during their life [5, 6]. In addition, the data collected through an online condition monitoring system refer to equipment under varying operating and environmental conditions. In the literature, semi-supervised and self-adaptable approaches are proposed for fault diagnosis in evolving environments [5]. In these contexts, the occurring of a different operating condition is seen as a concept drift detection problem. When a concept drift is detected, existing diagnostic models are re-trained including new available data. Thus, the main goal is to detect abrupt concept drifts which correspond to the occurrence of a novel operating condition. The concept of drift detection can be considered as a novelty detection problem [7]. According to [8], novelties here are seen as agglomerations of abnormal observations, i.e., anomalies, representing a fundamental change in the underlying processes generating the observations. Hence, novelty detection can also be seen as a one-class classification problem [9]. Giving their promising results in several domains, e.g., feature learning, pattern recognition, time series forecasting, Deep Learning algorithms are receiving great attention in the context of novelty detection. However, a comparison in terms of classification performance between ML and DL algorithms is still missing in the field of novelty detection. In addition, existing studies only consider the case of one-class classification, including one only normal condition during the learning process. This aspect limits their application to industrial machinery, which operates under several normal conditions.

The main goal of the present study is to compare ML and DL performance in the recognition of novel operating conditions of a system. In particular, two different scenarios and two different levels of analysis will be considered, in order to determine the best models, in terms of prediction accuracy, in offline and online scenarios and when a single point or a batch is considered during models training and testing. In addition, the performance of each method is also compared in terms of required computational times for both training and prediction, and with a varying training size.

The remaining of the paper is organized as follows. In Sect. 2, common models used in the context of novelty detection are briefly reviewed. In particular, details on models adopted for the comparative analysis are provided. Section 3 shows the results obtained by the application of those models on raw vibration signals collected from a test rig.

2 Methods for Novelty Detection

In general, novelty detection methods learn, during the training, a representation of the normal operation of the machinery and are used at serving time to identify deviations from this representation. During the test phase, they are therefore required to assign novelty scores to each test data and then categorize them as belonging to a new operating condition depending on whether the score exceeds a certain threshold or not [10].

The most common ML methods adopted for novelty detection are summarized in this section. The Local Outlier Factor (LOF) algorithm is an unsupervised anomaly detection method that computes the local density deviation of a given data point with respect to its neighbors. It considers as outliers the samples that have a substantially lower density than their neighbors [11]. The Isolation Forest ‘isolates’ observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature. Since recursive partitioning can be represented by a tree structure, the number of splits required to isolate a sample is equivalent to the path length from the root node to the terminating node. This path length, averaged over a forest of such random trees, is a measure of normality and our decision function. Random partitioning produces noticeably shorter paths for anomalies. Hence, when a forest of random trees collectively produces shorter path lengths for particular samples, they are highly likely to be anomalies [12]. The One-Class Support Vector Machine is an unsupervised learning algorithm that is trained only on the ‘normal’ data, in our case the negative examples. It learns the boundaries of these points and is therefore able to classify any points that lie outside the boundary as outliers [13]. The Principal Component Analysis (PCA) is frequently used in exploratory data analysis because it reveals the inner structure of the data and explains the variance in the data. PCA looks for correlations among the variables and determines the combination of values that best captures differences in outcomes. For anomaly detection, each new input is analyzed, and the anomaly detection algorithm computes its projection on the eigenvectors, together with a normalized reconstruction error. The normalized error is used as the anomaly score. The higher the error, the more anomalous the instance is [14]. Online clustering can be considered a distance-based novelty detection approach, in which the “normal” class is characterized by a small number of prototype points in the data space [10]. During the prediction step, the distance between the “normal” points and new points is computed. A threshold is set to determine whether the current pattern belongs to the same cluster as the normal one, or creates a new cluster. Among DL approaches, the most widely adopted methods for novelty and anomaly detection problems are autoencoders. They are neural architectures that compress the input data into a compact vector representation (encoding phase) and try to reconstruct the original data starting from this intermediate representation (decoding phase) [9, 10, 15]. In the context of novelty detection, these architectures identify new operating conditions when the reconstruction error obtained exceeds a certain threshold, which confirms that the processed input cannot refer to any normal condition encountered in the training phase. This generic architecture can be implemented using different types of neural networks: simple Feed-Forward neural networks, Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs). Feed-Forward AutoEncoder [9] relies on Multilayer Perceptrons, or MLPs for short, to encode and decode the input data and intermediate representations respectively. CNN AutoEncoder [15] applies convolutive filters to an input organized in a grid to derive an intermediate representation that encodes the spatial proximity information of the original data (encoding phase) and adopts an inverse strategy to re-expand this intermediate knowledge. RNN AutoEncoder [10] (LSTM in our case) is based on a recurrent connection of hidden representations generated from multiple MLPs, which

is exploited to compress and reconstruct data while preserving their sequentiality and order of occurrence.

3 Experimental Evaluation

For the purpose of the present study, an experimental platform was built in the Department of Industrial Engineering of the University of Bologna. Several tests have been conducted to get vibration signals and apply the methods described in the previous section. The goal of this analysis is to provide a comparative evaluation of them in terms of effectiveness and efficiency in order to understand the main trade-offs deriving from their use. In particular, two scenarios are considered. In the first scenario, named offline, the models are first trained on a single operating condition; then, their ability to discriminate between the known condition and the other, i.e., novel conditions, was analyzed; this scenario corresponds to the common approach, which requires the re-train of models each time a new condition occurs; in the second scenario, named online, the models are evaluated in terms of their ability to incorporate new knowledge. This scenario evaluates an incremental learning approach, in which the ability to learn machinery conditions that were unknown at the time of the initial offline training is assessed. In addition, for each scenario, two levels of analysis are conducted. In the first case, each sample is considered separately; the prediction accuracy is computed by Eq. 1, where N is the number of samples, $I(x)$ converts the outcome of a boolean condition (true or false) into 1 or 0, y_i and \tilde{y}_i represent the true and the predicted labels for the i -th sample, respectively

$$Accuracy (Acc) = \frac{1}{N} \sum_{i=1}^N I(y_i = \tilde{y}_i) \quad (1)$$

A second level of analysis considers a batch of samples instead of single samples. In this case, the batch accuracy is given by Eq. 2, where $|B_k|$ indicates the cardinality of the k -th batch, with $k = 1, \dots, M$ and M the number of batches, $y_{k,j}$ and $\tilde{y}_{k,j}$ represent respectively the true and the predicted labels for the j -th sample in the k -th batch.

$$Batch Accuracy (B. Acc) = \frac{1}{M} \sum_{k=1}^M I\left(\sum_{j=1}^{|B_k|} I(y_{k,j} = \tilde{y}_{k,j}) = |B_k|\right) \quad (2)$$

Hence, a prediction is considered correct when all the samples of a batch are correctly predicted. Finally, the performance of each model is also evaluated in terms of computational time of both training and testing.

The Dataset. The platform is shown in Fig. 1. It is composed of an asynchronous motor, a gearbox made of two pulleys that exchange the rotation through a belt, two shafts that share the motion thanks to a couple of gears, and an electromagnetic brake. The platform is provided with three triaxial accelerometers, which are placed on the bearing's support, next to the second pulley and the two gearboxes, respectively. They have a sampling frequency of 12.8 kHz per axis and an acceleration range of 500 G_{peak}. A complete description of the platform can be found in [17]. For the purposes of experimentation,

tests in four distinct operating conditions and a fault condition are conducted. The rotational speed is fixed at 660 rpm, while the distance between the pulley and the braking torque varies. The parameters, the duration, and the number of batches of each condition are shown in Table 1. Note that each batch has a length of 10 min. A representation of the raw signals in the 4 operating conditions is provided in Fig. 1. The considered signals represent a multivariate series where each feature is an acceleration. As can be seen, while the accelerations in the first operating condition are rather stable, significant oscillations occur in the other conditions, but only state 4 describes an anomalous operation of the machinery (i.e. states 1–3 represent normal operating conditions). Note that, since C4 is a fault condition, it will be used only as test data (i.e. no model will be trained on this anomalous state).

Table 1. Dataset description

Operating conditions	Distance between pulleys (mm)	Braking torque (Nm)	Duration (min)	Number of batches
C1	27.33	0.1	70	7
C2	27.33	0.5	150	15
C3	27.54	0.1	70	7
C4	27.54	0.1	30	3

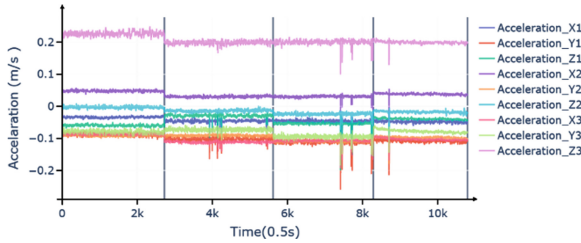


Fig. 1. Raw signals corresponding to each operating condition

Offline Evaluation Scenario. To carry out this evaluation we trained the models in turn on one batch at a time and we evaluated them on the remaining batches. This evaluation is repeated until each batch has been used for model training. The behavior we expect to obtain is that for all the test data associated with the same machine condition used for the training no novelty state is detected, while new machinery conditions are detected for the other samples. The results of this experiment are shown in Table 2, where for each model and training machinery condition, the accuracy, given by Eq. 1, and the batch accuracy, given by Eq. 2, computed over all the datasets are reported. The models that provide the worst performance are SVM, IF and LOF, while the remaining models are almost equivalent. With the exception of C3 where they produce poor results, these three models generate good performance on single samples (i.e. they obtain accuracy values in the

range 0.66–0.97), however, in batch-level evaluation, they produce many false alarms (i.e. they obtain batch accuracy values in the range 0.3–0.6). Furthermore, it is possible to see how the most difficult operating condition to identify is C3, in which SVM, IF and MLP show the most significant reductions in performance, while C1 is recognized by the models with the highest effectiveness. This differentiation in performance does not apply to LSTM and CNN which are highly effective on all scenarios without distinction.

Table 2. Breakdown of model performance by operating condition

Algorithm	All		C1		C2		C3	
	Acc.	B. Acc.	Acc.	B. Acc.	Acc.	B. Acc.	Acc.	B. Acc.
Clustering	0.988	0.758	0.998	0.941	0.978	0.627	0.995	0.830
LOF	0.817	0.572	0.990	0.840	0.915	0.544	0.411	0.326
PCA	0.965	0.808	0.999	0.945	0.939	0.752	0.981	0.772
SVM	0.658	0.351	0.982	0.715	0.703	0.290	0.189	0.067
IF	0.880	0.619	0.994	0.867	0.967	0.598	0.561	0.379
MLP	0.957	0.911	1.000	1.000	0.945	0.906	0.933	0.821
LSTM	0.989	0.944	0.998	0.977	0.984	0.927	0.990	0.942
CNN	0.989	0.939	0.998	0.980	0.984	0.919	0.989	0.933

Online Evaluation Scenario. Drawing inspiration from [7], in this experiment we simulate the adoption of the models in a dynamic scenario where a continuous monitoring of the machinery is performed, and an incremental knowledge of the operating conditions of the machinery is learned by the diagnostic system. To create this experimental scenario we have considered the three settings shown in Table 3.

Table 3. Online scenario configurations

Conf	Training set	Test set	
		Known set	Novel set
S1	C1 (10 min)	C1 (70 min)	C2, C3, C4 (150 + 70 min + 10 min)
S2	C1, C2 (10 + 10 min)	C1, C2 (70 + 150 min)	C3, C4 (70 min + 10 min)
S3	C1, C2, C3	C1, C2, C3	C4

In the first configuration, each model is trained exclusively on a 10-min batch of C1 and an operating cycle is then applied to the other machinery settings. In the second configuration, it is assumed that the model has also learned of the existence of C1 and C2, and the same operating cycle is applied to other machinery settings. Finally, the

last configuration evaluates the behavior of each model when trained jointly on all three states. Note how each model stores for each state a limited amount of data compared to the totality of measurements made (i.e. only 10 min of data for each state are considered). In this way 1) the model is trained quickly and it can continue to monitor the behavior of the machinery and 2) no dedicated storage is needed to store the entire measurement history. Results of this scenario are reported in Table 4, where for each model the batch accuracy is reported both for the entire test set, i.e., all observations included in both known and novel tests as defined in Table 3, and the known and novel sets as defined in Table 3, individually. Similar results were obtained considering the record-level accuracy, which were not reported due to space constraints. From Table 4, it is possible to observe that the configuration where the models perform best is the first one (S1), where the models are trained exclusively on C1. This confirms that C1 is significantly different from the other states, thus facilitating its distinction with respect to the other states. In this configuration, the worst-performing models are SVM, LOF, and IF, which correctly recognize C2, C3, and C4 as new operating conditions, however they tend to wrongly categorize batches belonging to C1. In the second configuration, on the other hand, there is a significant reduction in the effectiveness of the models, with the exception of LSTM and CNN. In more detail, the LOF, PCA, and Clustering models hardly recognize C1 and C2 as already known operating conditions (low accuracies on the known set). This is probably due to the integration of C2 with the training data, which has made the separation between the operating conditions less marked. Finally, by analyzing the third configuration, it is possible to note how all the models produce good performance in recognizing the known set, however they are no longer able to discriminate it with respect to the state C4, which actually presents, in the phase preceding the failure, very similar characteristics compared to other conditions. In particular, all models except LSTM and CNN have an accuracy equal to 0, i.e. they cannot correctly predict even a batch. A more detailed analysis of these results revealed that the record-level accuracy of these models varies in the range 0.1–0.33, while for LSTM and CNN in the range 0.81–0.83.

Table 4. Models batch accuracy in online scenario

Algorithm	C1			C2			C3		
	Test	Known	Novel	Test	Known	Novel	Test	Known	Novel
Clustering	0.941	0.732	1.000	0.567	0.503	0.700	0.410	0.455	0.000
LOF	0.840	0.304	0.990	0.456	0.395	0.586	0.690	0.767	0.000
PCA	0.945	0.750	1.000	0.235	0.000	0.729	0.000	0.000	0.000
SVM	0.715	0.321	0.825	0.618	0.565	0.729	0.790	0.878	0.000
IF	0.867	0.411	0.995	0.618	0.626	0.600	0.757	0.841	0.000
MLP	1.000	1.000	1.000	0.871	1.000	0.600	0.900	1.000	0.000
LSTM	0.997	0.893	1.000	0.935	0.952	0.900	0.743	0.772	0.476
CNN	0.980	0.911	1.000	0.922	0.912	0.943	0.790	0.847	0.286

To further inspect the superiority shown by the LSTM-based AutoEncoder over competitive methods, we propose in Fig. 2 a visual inspection of its internal representation for the examined operating conditions. In the figure this representation is compared with the raw distribution of operating states when projected into a two-dimensional space generated by the popular t-SNE technique [15]. As you can see, the embedded space created by the LSTM emphasizes the separation between the different operating conditions more than in the original feature space.

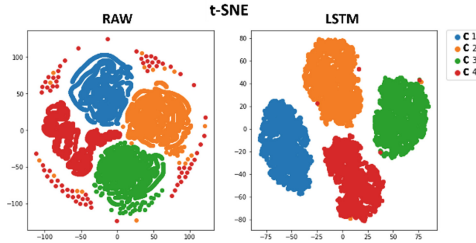


Fig. 2. LSTM-based AutoEncoder embedded space compared with the original feature space

Impact of Data Size on Performance. In this subsection, the impact of the size of the training data on model performance and computation time is assessed. For the evaluation of the performance, each model is trained on a variable number of batches associated with the same machinery state, i.e. 5, 10, 20 and 30 min of data. Each model is then required to recognize the test data as referring to a novel or already known condition of the machinery. Results of this experiment are shown in Fig. 3, where for each model the evolution of batch accuracy as the training size increases is reported for all the three states considered. Results show that the performance of deep learning models is high even with a small amount of training data (e.g., 5 min) and is not influenced by the availability of further training data. As for the other approaches, the variable availability of training data influences their performance (with the exception of the PCA which produces equivalent results for all the settings considered). In particular, Clustering and SVM are more effective as the size of the training set increases: SVM achieves accuracy improvements in the range between 9–20%, while Clustering between 6–20%. Similar trends are also confirmed for the LOF and IF models on C1 and C2. However, the latter two models perform worse as the size of the training set increases when trained on C3. A more detailed analysis of these results has revealed that in this setting they are unable to distinguish new states (i.e., LOF and IF generate a batch accuracy of 3 and 6%, respectively). The evaluation of computational time with varying training size is conducted to assess 1) the impact of a training process on the inactivity of each model, and 2) their velocity in detecting possible new operating conditions of the machinery (i.e., the prediction time). Note that models are run on a VM deployed on Google Cloud with 12 GB of RAM, GPU K80, and Intel(R) Xeon(R) CPU @ 2.30 GHz. For each model, both training and test times are considered. Each model is trained on 5, 10, 20 and 30 min of data and the relative times are recorded. In addition, the prediction time over 1 sample, 10-, 20- and 30-min batches was also recorded. The results of these two

experiments are displayed in Fig. 4 on the left and right plots respectively. From the plot on the left in Fig. 4, it is possible to observe how the models require very different training times. Models like SVM and PCA only take a few milliseconds to complete the training. Times equal to almost two orders of magnitude are instead produced by IF and MLP models. Clustering and CNN, on the other hand, require times in the order of a second or a few tens of seconds. Finally, the LSTM model is the one that produces the highest training times: from 4 min in the configuration with fewer data to a maximum of 20 min. A confirmation of these trends is obtained by analyzing the prediction times shown in the right plot, although the latter are two orders of magnitude lower. From these results it is also possible to note that the Clustering approach produces significantly higher prediction times than the other techniques (with the exception of LSTM). This is due to the quadratic nature of the approach, although it was partially alleviated through a mixed training strategy where a first clustering solution was produced from 1000 samples and then the remaining data were included in an online fashion. Finally, it is possible to observe how the time required to evaluate whether a single sample belongs to an already known or novel state is a few milliseconds, making them all suitable for operating in an online scenario.

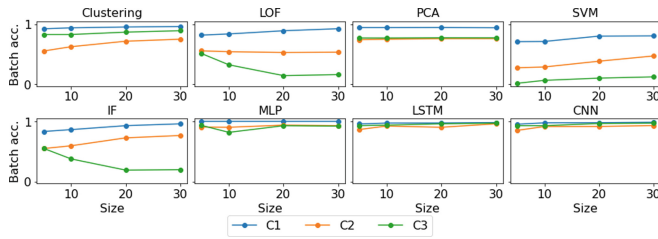


Fig. 3. Performance evaluation as the training size increase

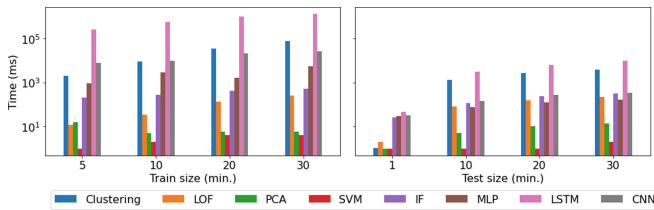


Fig. 4. Train (left) and test (right) performance by varying the data size

4 Conclusions

In this paper we have provided a comparative analysis of the performance of traditional techniques (e.g. Clustering, LOF, PCA, SVM, IF) and more advanced approaches based on deep learning models (Autoencoder, CNN) for novelty detection tasks in the context

of fault diagnosis under varying operating conditions. The evaluation has been conducted in multiple test scenarios. The effectiveness was measured both in offline and online settings, in order to compare the ability of the models to exploit already known information and to incorporate new ones for the purpose of novelty detection. Furthermore, a variable dimension of the data was considered to analyze their impact on the time and effectiveness performance of these techniques. The main outcomes of the evaluation can be summarized as follows. First, traditional methods are less effective than DL-based models, however the latter requires more time for both training and inference. Second, methods based on autoencoders have shown greater robustness to noisy signals than competitive approaches. In summary, these results support the direction towards a beneficiary use of Deep Learning techniques in the context of novelty detection. Results in terms of testing times are promising for industrial streaming applications of fault diagnostics under dynamic environments. Further assessments in more complex scenarios will be conducted to verify the generality of this consideration.

References

1. Ayvaz, S., Alpay, K.: Predictive maintenance system for production lines in manufacturing: a machine learning approach Using IoT data in real-time. *Expert Syst. Appl.* **173**, 114598 (2020)
2. Cakir, M., Guvenc, M.A., Mistikoglu, S.: The experimental application of popular machine learning algorithms on predictive maintenance and the design of IIoT based condition monitoring system. *Comput. Ind. Eng.* **151**, 106948 (2021)
3. Jardine, A.K.S., Lin, D., Banjevic, D.: A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mech. Syst. Signal Process.* **7**(20), 1483–1510 (2006)
4. Çinar, Z.M., Nuhu, A.A., Zeeshan, Q., Korhan, O., Asmael, M., Safaei, B.: Machine learning in predictive maintenance towards sustainable smart manufacturing in Industry 4.0. *Sustain* **12**(19), 8211 (2020)
5. Hu, Y., Baraldi, P., Di Maio, F., Zio, E.: A systematic semi-supervised self-adaptable fault diagnostics approach in an evolving environment. *Mech. Syst. Signal Process.* **88**, 413–427 (2017)
6. Calabrese, F., Regattieri, A., Botti, L., Mora, C., Galizia, F.: Unsupervised fault detection and prediction of remaining useful life for online prognostic health management of mechanical systems. *Appl. Sci.* **10**(12), 4120 (2020)
7. Cariño, J.A., et al.: Fault detection and identification methodology under an incremental learning framework applied to industrial machinery. *IEEE Access* **6**, 49755–49766 (2018)
8. Gruhl, C., Sick, B., Tomforde, S.: Novelty detection in continuously changing environments. *Futur. Gener. Comput. Syst.* **114**, 138–154 (2021)
9. Pimentel, M.A.F., Clifton, D.A., Lei, A., Tarassenko, L.: A review of novelty detection. *Signal Process.* **99**, 215–249 (2014)
10. Pang, G., Shen, C., Cao, L., Hengel, A.: Deep learning for anomaly detection: a review. *ACM Comput. Surv.* **54**(2), 1–38 (2021)
11. Breunig, M., et al.: LOF: identifying density-based local outliers. In: *SIGMOD/PODS00: ACM International Conference on Management of Data and Symposium on Principles of Database Systems*. Dallas Texas USA, May 2000
12. Wei, X., Ling, H., Fox, A., Patterson, D., Jordan, M.: Large-scale system problems detection by mining console logs. In: *22nd ACM Symposium on Operating Systems Principles*, Montana (2009)

13. Schölkopf, B., et al.: Estimating the support of a high-dimensional distribution. *Neural Comput.* **13**(7), 1443–1471 (2021)
14. Liu, F.T., Ting, K.M., Zhou, Z.-H.: Isolation-based anomaly detection. *ACM Trans. Knowl. Disc. Data (TKDD)* **6**(1), 1–39 (2012)
15. Li, X., Li, X., Ma, H.: Deep representation clustering-based fault diagnosis method with unsupervised data applied to rotating machinery. *Mech. Syst. Signal Process* **143**, 106825 (2020)
16. Van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. *J. Mach. Learn. Res.* **9**(11), 53 (2008)
17. Calabrese, F., Regattieri, A., Bortolini, M., Gamberi, M., Pilati, F.: Predictive maintenance: a novel framework for a data-driven, semi-supervised, and partially online prognostic health management application in industries. *Appl. Sci.* **11**(8), 3380 (2021)