Pattern Recognition Letters
journal homepage: www.elsevier.com

Using an autoencoder in the design of an anomaly detector for smart manufacturing.

Antonio L. Alfeo[a], Mario G.C.A. Cimino[a], Giuseppe Manco[b], Ettore Ritacco[b,**], Gigliola Vaglini[a]

[a]*University of Pisa, Largo Lucio Lazzarino 1 , Pisa 56122, Italy*
[b]*CNR ICAR, Via Pietro Bucci 8/9C, Rende 87036, Italy*

ABSTRACT

According to the *smart manufacturing* paradigm, the analysis of assets' time series with a machine learning approach can effectively prevent unplanned production downtimes by detecting assets' anomalous operational conditions. To support smart manufacturing operators with no data science background, we propose an anomaly detection approach based on deep learning and aimed at providing a manageable machine learning pipeline and easy to interpret outcome. To do so we combine (i) an autoencoder, a deep neural network able to produce an anomaly score for each provided time series, and (ii) a discriminator based on a general heuristics, to automatically discern anomalies from regular instances. We prove the convenience of the proposed approach by comparing its performances against isolation forest with different case studies addressing industrial laundry assets' power consumption and bearing vibrations.

1. Introduction and Motivation

Smart manufacturing paradigm aims at enhancing production adaptability and efficiency by integrating manufacturing assets with technologies such as cloud computing, internet of things, and machine learning (Kusiak, 2017). These solutions enable unprecedented data-driven decision support, with applications ranging from predictive maintenance to anomaly detection (Ghosh et al., 2016). In contrast with predictive maintenance, anomaly detection does not require large sets of examples of assets' anomalous behaviors, that are rare events by definition. Moreover, anomaly detection approaches can effectively support assets' health diagnosis. Indeed, a higher frequency of anomalies' occurrence may be linked to assets' deterioration, thus resulting in unplanned downtimes (Zhao et al., 2017).

However, the development of anomaly detection solutions in real-world manufacturing demands a trade-off between system management cost and detection accuracy. Specifically, by incorporating more domain knowledge, the system provides better accuracy (Lopez et al., 2017), but it requires a complex adhoc design, and costly post-processing, resulting in poor scalability, generalization and manageability (Fan et al., 2018).

**Corresponding author: Tel.: +39 09848317; fax: +0-000-000-0000; *e-mail:* `ritacco@icar.cnr.it` (Ettore Ritacco)

The approaches for anomaly detection in smart manufacturing can be classified according to the domain knowledge they incorporate as statistical, model-based, and phenomenological approaches (Lopez et al., 2017). With *statistical approaches*, anomalies and regular instances are distinguished according to their different statistical properties; the domain knowledge is not incorporated in the detection mechanism in any form, providing a general and manageable solution. With *model-based approaches*, the domain knowledge results in a precise model (e.g. an equations) used to characterize each instance as a regular or anomalous operational condition; e.g. the anomalies of a logic controller can be accurately detected by employing its control logic. *Phenomenological approaches* provide a trade-off between the first two; these approaches employ some degree of domain knowledge to characterize data point distributions (e.g. by defining behavioral classes), resulting in a model able to distinguish regular instances and anomalies. According to (Lopez et al., 2017) phenomenological approaches may employ different strategies. Among those, *knowledge-based approaches* compare process measurements against known fault patterns (e.g. via expert systems), but require a comprehensive knowledge of the system to link inputs combinations to system operational conditions. The same applies to approaches based on *state estimation*, i.e. modeling the anomalies as states of the system, enabling their detection through states history and current inputs and outputs. A solution based on *classification* needs a large set of regular and anomalous instances to properly model the corresponding behavioral classes. With *limit checking approaches*, anomalies are detected via signals features exceeding "regular behavior ranges", however, the definition of such ranges is often manual and needs to be repeated for each asset. Solutions based on *clustering* imply a time-consuming subsidiary task to identify the clusters with regular occurrences (Li et al., 2017). Finally, with approaches based on *regression*, the relationship between instances' features (predictors) and the regular operational conditions is modeled; anomalies' are detected since they do not conform with the model to some extent, e.g. measured with an anomaly score. These solutions offer a simpler implementation at the cost of a non-trivial score intepretability. As an example, authors in (Liu et al., 2012) use an isolation forest to partition the data according to their density, recursively obtaining subsets with fewer and fewer instances. Since anomalies are usually located in sparse regions, they result in shorter tree branches. In (Amer et al., 2013) authors use an approach based on one-class support vector machine to separate regular instances from anomalous ones through a boundary in the features space. Due to their rarity, anomalies contribute less to the decision boundary and result in a greater distance from it. Within regression-based approaches, deep neural networks provide greater efficiency while modeling complex structure underlying in the data. Among them, deep learning approaches based on *autoencoder* (Liou et al., 2008) are often used for anomaly detection. As an example, in (Lindemann et al., 2019) authors provide an anomaly detection architecture by combining an autoencoder with LSTM (long short term memory cells) to capture complex temporal dependencies in the data. A properly trained autoencoder is able to correctly reconstruct non-anomalous inputs. As such, the likelihood that a given input is an anomaly can be assessed by the autoencoder reconstruction error, i.e. the anomaly score.

However, the score itself does not provide a clear distinction between anomalies and regular instances and may be hard to interpret, especially for operators with no data science background. Given how critical is to obtain reliable insights from such analysis (Sun et al., 2019), *data scientists* are often employed to determine an anomaly score's threshold corresponding to a sufficiently significant deviation for a data point to be considered an anomaly. This results in additional loops in the anomaly detection management pipeline, as shown via the BPMN diagram in Fig. 1 by employing the taxonomy provided by (Saltz and Grady, 2017). The pipeline begins with the *data engineer* setting up the data ingestion, to collect and store the monitored assets' data. They care also about the data selection and preprocessing, to ensure that the data used to train the anomaly detector is representative and consistent. It follows the extraction of representative features from the data and their split in training and testing set. Those are employed by the *data scientists* to train and test the model of the anomaly detector they designed. The pipeline ends with the positive evaluation of the detection performance, which is provided by the *business expert*. Receiving a positive evaluation may require many iterations aimed at adjusting the anomaly detector hyper-parameters, and refining the anomaly score threshold. While the cost of the first iteration can be mitigated by using automated hyper-parameter tuning (Zhang et al., 2019a), the last one is a critical process often performed manually since too restrictive thresholds may increase the false negatives, and loose thresholds results in many false positives. This process is even more difficult with real world data, which may be characterized by a significant amount of noise, i.e. minor deviations. Those represent the semantic boundary between regular instances and true anomalies. Moreover, the separation between noise and anomalies is not discrete, i.e. many data points created by a noisy generative process may be deviant enough to be interpreted as anomalies according to their score (Aggarwal, 2015).

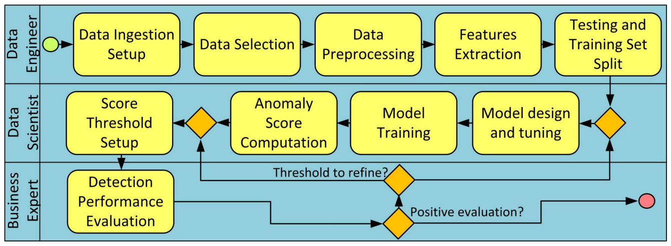

Fig. 1. BPMN diagram of the anomaly detection pipeline.

By summarizing, real-world smart manufacturing requires general and accurate anomaly detection, but may suffer from computational inefficiencies and costly human-driven post-processing to improve the effectiveness and the interpretability of the detection (Wuest et al., 2016). In this work, we address those issues by combining autoencoder with a heuristic-based discriminator. The autoencoder is aimed at providing the anomaly detector with (i) adaptation capability, i.e. being effectively employable in different contexts, and (ii) computational efficiency, to reduce the cost of refining the anomaly detector, especially with automatic tools iteratively improving the model hyper-parameters. On the other hand, the discriminator aims at (i) improving the interpretability of the anomaly score, and (ii) providing an accurate and precise detection outcome, without the intervention of a data scientist (Fig. 1). The convenience of the proposed approach is shown by comparing its performances against one of the best-in-class competitors, i.e. isolation forest, with 8 different case studies. Paper is structured as follows. In section 2, we detail our anomaly detection approach. Section 3 presents our case studies. The experimental setup and obtained results are discussed in section 4 and 5, respectively. Finally, section 6 summarizes conclusions and future works.

2. Design

The data are prepared for the analysis by extracting 16 features from each time series, then rescaling and splitting those in training and testing sets. In order to discover both local and global anomalies, the features extraction can employ a spatio-temporal clustering procedure with adaptive time granularity (Cimino et al., 2015; Galatolo et al., 2018). However, this requires an additional hyper-parameter tuning process and results in non-interpretable representational space. A more manageable solution is capturing the trends of the time series over time with statistical features computed both on the entire duration of each observation (features *a-e*) and on semi-overlapping time windows, e.g. with features *f* each time windows has a duration of one-quarter of the overall length of the series. For each time series we extract the following features:

(a) 90th, 75th, 50th and 25th percentile of the time series
(b) mean absolute deviation of the time series
(c) skewness of the time series
(d) number of continuous time intervals with values greater than 90th, 75th, 50th and 25th percentile of the time series
(e) number of samples greater than 50% and 25% of the daily maximum of the time series
(f) the difference between the mean absolute deviation of the current time window and mean absolute deviation of the whole time series

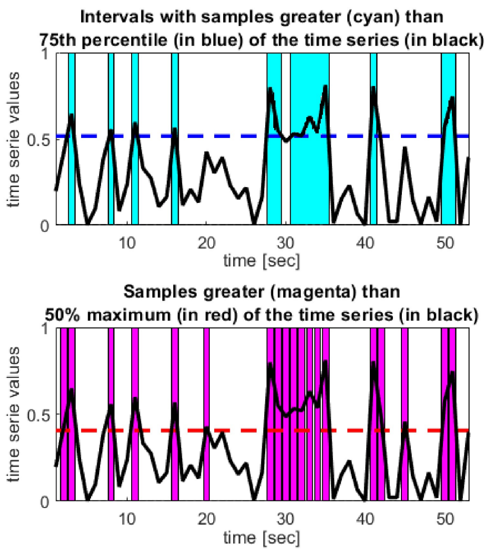

Fig. 2. Example of features extracted from time series. The time series is characterized by 8 intervals over the 75th percentile and 19 samples over 50% of the maximum.

Specifically, the number of continuous intervals in which the time series is greater than a given threshold (features *d*) represents the fluctuations across such threshold. On the other hand, the number of samples greater than a given threshold (features *e*) represents the total duration of the time series above that threshold (Fig. 2). Chosen feature thresholds aim at addressing both high and low ranges of the time series values. Finally, the thresholds for features *e* are obtained as percentages of the maximum of the time series (rather than a percentile), since the number of samples greater than a given percentile would always be the same. The features are rescaled by using a Min-Max procedure (Goldstein and Uchida, 2016). The training set is obtained by randomly picking 70% of the regular instances, whereas the testing set contains the remaining 30% of regular instances and an equal percentage of all the anomalous ones.

Our approach employs an *autoencoder* to score the anomaly degree of each input instance. An autoencoder is a deep neural network architecture made of 2 main components: encoder and decoder (Fig. 3). The decoder is able to reconstruct the input data from a compact hidden representation. The encoder is in charge of producing a compact hidden representation of the inputs, by building a few representative features and ignoring meaningless components. The autoencoder is trained to minimize the reconstruction error, i.e. *mean square error* between training set inputs and their reconstruction. By being trained with regular occurrences, the autoencoder can reproduce these with a little reconstruction error, whereas anomalous occurrences result in greater errors.

The reconstruction errors are further processed by the *discriminator*, which rescales them between 0 and 1 by using a sigmoidal function tuned via two parameters, i.e. α and β (Fig. 3). Values lower than α are transformed into 0. Values higher than beta are transformed into 1. Values between these two values are progressively shifted toward 1 or 0 according to their proximity to α and β. The value of α is obtained as the 99th percentile of the reconstruction errors in the training set. The midpoint between α and β corresponds to the average reconstruction error obtained by processing two minor synthetic anomalies, created by collecting the maximum and minimum of all the features in the training set. The value of β corresponds to α plus twice the distance between α and the midpoint (Fig. 3). The discriminator is aimed at automating the distinction between anomalies and regularities. With the classic anomaly detection pipeline (Fig. 1), this was obtained with the threshold iteratively chosen by the data scientist, but, by using the discriminator, the anomaly score has a precise meaning: regular occurrence (0), anomalous occurrence (1), and noise, e.g. warning operational condition (between 0 and 1).

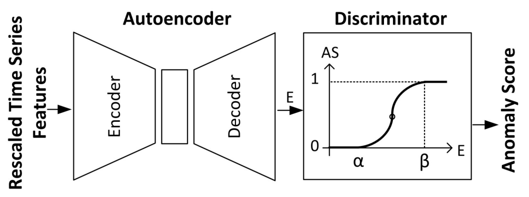

Fig. 3. Anomaly detector architecture.

3. Case studies

In smart manufacturing context, assets' operational condition can be inferred by analyzing power consumption, vibrations, or acoustic emission (Zhang et al., 2019b). Among those, vibration and acoustic monitoring usually require additional expensive sensors, whereas power consumption is measured within each asset's machine controller. Moreover, power consumption has shown great potential in supporting health monitoring (Simon et al., 2017), and anomaly detection (Ouyang et al., 2018) with smart manufacturing assets. As such, in this work we address the anomalies in assets' energy consumption, while as manufacturing scenario we employ an industrial laundry.

In an industrial laundry, each asset cares a specific phase of garments processing. This process begin with the arrangement of the garments into batches to be moved to the washing process. The washing process is carried out by the continuous batch washing (CBW) or the washer-extractor. A CBW is a large cylinder separated into compartments arranged as a screw; by rotating, batches are moved through each compartment, and each one of them performs a predefined task (pre-wash, wash, etc.). The washer-extractor is the industrial version of a household washing machine, and its washing programs includes a spin-dryer function. Alternatively, the moisture in washed garments is reduced by heating or squeezing them, i.e. by using a dryer or a hydro-extracting press. Then, the garments undergo the finishing operations, carried out by mangles, finishing tunnel (i.e. tunnel with jets of hot air or steam) or ironers, which uses padded rollers that press the garments against a heated surface to remove the wrinkles and completely dry them. Eventually, the clean and folded garments are arranged neatly in a warehouse, ready to be returned.

Among those assets, we choose the ones to analyze by exploiting the information our domain expert has provided. Specifically, we know that (i) the dryer, CBW, washer-extractor, and hydro-extracting press exhibit an high energy consumption, which increases in poor maintenance conditions (e.g. asset aging); (ii) an ironer can exhibits peaks of power consumption in case of stacked garments or during the start-up of the ironing process. Thus, our analysis focus on the power consumption of *CBW*, *washer extractor* and *ironer*. These time series are realistically reconstructed by exploiting the official information of assets' manufacturers (e.g. nominal energy consumption), a baseline example for each asset from previous studies (IEEA, 2010), and the details provided by our project industrial partner. This allows representing a variety of regular and anomalous behaviors (Zenisek et al., 2018), which is commonly missing in real-world scenarios given that (i) the extensive adoption of remote monitoring technologies in manufacturing is quite new, and (ii) anomalies are rare event by definition.

The power consumption time series are characterized by a sampling rate of 1 Hz (Humphrey et al., 2014) and a standard deviation per minute of 0.015 (Skoogh et al., 2011). Each time series address the power consumption during the day (24 hours) corresponding to a total amount of 86400 samples per time series. Moreover, each asset's power consumption has a peculiar temporal behavior. Specifically, the CBW may require 2-4 kWh, but it can decrease due to production stop or minor workload between 0.25 and 0.75 kWh. Those production downtimes are randomly arranged in the working day with a duration from 1 to 4 hours (IEEA, 2010). Similarly, the washer-extractor energy consumption can range from 3.5 to 4.6 kWh with a few brief drops (between 0.5 and 3 kWh) randomly arranged during the working time. After those, the power consumption rises to 4.5 kWh and then progressively decreases with a couple of spikes of +0.2/+0.3 kWh (Kathrin et al., 2011). The power consumption of an ironer exhibits a peak (22 kWh) during the first working hour, followed by a sharp drop. Then it ranges between 15 kWh and 20 kWh. It can have short (a couple of hours at most) intervals characterized by a lower consumption, between 12 and 8 kWh. Depending on the model of the ironer, its power consumption can reach 22 kWh when fully operational (David et al., 2013). Those temporal behaviors are summarized by the following features: (i) kWh consumed by the machine when fully operational; (ii) kWh consumed by the machine when stopped; (iii) occurrence of short stops (15 to 30 minutes long); (iv) occurrence of stops in the working day; (v) duration of stops in minutes. By randomly extracting values from the known ranges of each feature, we build each power consumption time series. Table 1 shows the features' ranges for the 3 above-mentioned assets.

We obtain the time series of anomalous power consumption by manipulating the regular ones as it follows. The ironer features a few peaks (3 to 4) in the power consumption, each one lasting from 10 to 20 minutes, the energy consumption increases between 110% and 150% of the expected maximum value; the machine start-up or the stacking of garments are simulated. The CBW and the washer extractor feature an increased power consumption, between 120% and 150% of the expected maximum value; the aging of the machine is simulated.

According to classic statistical models (Chandola et al., 2009), anomalous instances should correspond to less than 1% of the whole dataset. However, this assumption seems to be optimistic with real-world smart manufacturing data. In this context, the amount of anomalies is usually higher, often resulting in percentage of anomalous instances up to 5% (Paulheim and Meusel, 2015). We employ such a percentage of anomalies in our datasets to have a realistic setup. Please note that, this is not an assumption or a requirement for the proposed approach. Indeed, once trained, the system can handle any amount of anomalies.

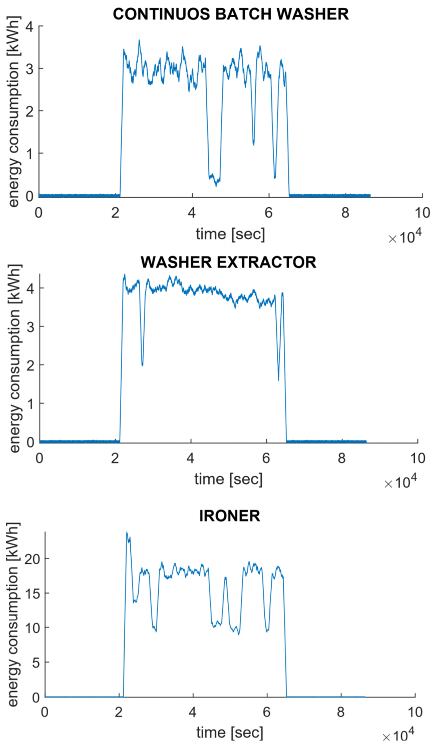

Fig. 4. Daily energy consumption of CBW, washer extractor and the ironer.

Finally, to prove the generalization capability of the proposed approach, we test it also with real-world industrial bearing vibrations' data (Bechhoefer, 2013). The dataset consists of 20 real-world bearing vibrations time series sampled at 25 Hz, representing 3 regular operational conditions and 17 anomalous

Table 1. Power consumption time series features. Short downtimes are always between 15 and 30 minutes.

Asset	kWh Operation	kWh Downtime	Short Downtimes per day	Downtimes per day	Downtimes Duration [min]
CBW	[2, 4]	[0.25, 0.75]	[2, 4]	[1, 1]	[60, 240]
Washer-extractor	[3.5, 4.6]	[0.5, 3]	[3, 5]	[0, 1]	[15, 15]
Ironer	[15, 20]	[8, 12]	[1, 3]	[2, 3]	[45, 180]

operational conditions: 10 outer race fault conditions, and 7 inner race fault conditions respectively. The anomalous operating condition is not reconstructed, nor are their characteristic behavior known. This entirely leaves the recognition of the anomaly to the capability of the anomaly detection system. We split each time series in time windows of 1000 samples length, corresponding to a duration of 40 seconds. Then, 5 new datasets are created by randomly sampling regular (baseline conditions) and anomalous (faulty conditions) time windows, corresponding to 95% and 5% of each new dataset, respectively. Basically, we build an imbalanced classification problem that, given the diversity of the original fault conditions and the random subsampling used to inject them into these new datasets, can approximate an anomaly detection one (Aggarwal, 2015).

By summarizing, the data used to test our approach are composed by (i) 3 case studies addressing the power consumption of industrial laundry assets, i.e. CBW, washer extractor and ironer (PC1, PC2, PC3 respectively) and, (ii) 5 case studies about bearing vibrations (BV1, BV2, BV3, BV4, BV5). Table 2 reports the main characteristics of the data employed in our experiments, such as actual duration of the time series, their sampling rate and the number of samples, the total amount of time series per dataset, how many of them are used for training and testing the system, the percentage of anomalies in the dataset, and if the dataset is real or not.

4. Experimental Setup

We develop our anomaly detection architecture by using Tensorflow 2.0 (Abadi et al., 2016). The autoencoder consists of with 3 dense neural network layers for the encoder (16+8+4 neurons) and the same for the decoder (4+8+16 neurons). As loss function we use the mean square error (MSE), as neurons' activation function we use *relu* (rectified linear unit), and *adam* as optimization strategy due to its computational efficiency and little memory requirements (Zhu et al., 2017). We compare our anomaly detection approach against isolation forest, given its competitive performance (Riazi et al., 2019). The implementation of isolation forest is provided by (Ye and Guang-Tong, 2017), it does not imply features sub-sampling and sets as maximum tree depth the logarithm of two of the number of instances (Liu et al., 2012). Moreover, we test our approach effectiveness and computational efficiency by varying its main hyperparameters, i.e. by increasing the model's complexity. Specifically, with autoencoder the complexity of the model is due to the number of links between neurons in the network, which depends on the number of layers and neurons in each one of them. The basic model consists of 6 layers, three for the encoder with 16, 8, and 4 neurons, and three for the decoder with 4, 8, 16 neurons, resulting in 336 links between neurons. By doubling them we obtain 6 layers for the encoders with 16,14,12,10,8,6,4 neurons and the same for the decoder (in reverse order), resulting in 1344 links. By doubling their number again we get 24 layers, 12 in the encoder with 16 to 4 neurons, and 12 in the decoder from 4 to 16 neurons, resulting in 2696 links. On the other hand, with isolation forest, the complexity of the model lies in the number of nodes for each decision tree, and on the number of trees. We counted the average number of nodes in the trees for our application, and this is between 42 and 62, on average 55. Therefore, we can roughly define the complexity of the model based on isolation forest as the number of trees multiplied by 55. In order to test the effectiveness of the recognition using a comparable complexity between the two approaches, isolation forest is used in the following configurations: 10, 25 and 50 trees corresponding to about 550, 1375 and 2750 nodes.

Furthermore, by increasing the number of training epochs we expect a greater training time and a smaller detection error. The hardware/software platform used for our tests employs a CPU Intel Core i7-6700 at 2.60-3.50 GHz, 6M Cache, 16 GB DDR3L 1600MHz RAM, Windows 10 OS.

To have an easy to interpret detection result, the outcome of the discriminator can be rounded and used as a label to discern anomalies from non anomalies. As such, it is possible to compute the detection accuracy and precision. To verify the discriminator effectiveness we use this approach also in conjunction with isolation forest, for testing whether it increases the detection performances. Finally, we compare the effectiveness of the discriminator with another well-known method for discriminating anomalies based on their score, i.e. making Gaussian assumptions about the distributions of anomaly scores and using a threshold obtained by considering the scores' mean (μ) and standard deviation (σ) as $\mu + 3\ \sigma$ (Jin et al., 2016).

Achieved results are presented as confidence intervals obtained with 10 repetitions. At each repetition training and testing sets are respectively generated by randomly sampling the 70% of the regular instances, and 30% of the regular and anomalous instances from the data available for each case study, resulting in a monte carlo cross-validation.

5. Results and Discussion

First, we measure the effectiveness of the detection as the MSE between the anomaly scores and the anomaly labels (1 anomalous, 0 otherwise) using 100 training epochs both with our approach and isolation forest. This is repeated with different hyper parameter settings, arranged in Table 3 to make adjacent settings correspond to comparable model complexities within the two approaches. The results show that the proposed approach always performs better, even with the minimum model complexity. As this increases, the confidence intervals of isolation forest shift downwards. On the other hand, with autoencoder there is no significant improvement except with the

Table 2. Datasets and time series (TS) main characteristics.

Name	TS duration	Sampling Rate	Samples per TS	# Instances	# Train	# Test	% Anomalies	Real
PC1-PC3	24 hours	1 Htz	86400	1000	665	300	5%	no
BV1-BV5	40 seconds	25 Htz	1000	600	400	180	5%	yes

case study BV1. One possible explanation for those behaviors lies in the effects obtained by increasing model complexity with isolation forest and autoencoder: in the first case, adding decision trees means building a more general model, whereas in the second case adding layers may result in overfitting, preventing the deep neural network to improve its performances.

Considering the training time needed with both approaches (Table 4), the autoencoder offers greater computational efficiency as the complexity of the model increases. This directly affects the management cost linked to the refinement of the anomaly detection model (Fig. 1) or its re-training. As such, this property is critical, especially with new or more complex detection problems (e.g. with noisier data).

In order to test the detection effectiveness according to the number of training epochs, we take the most complex versions of both algorithms, i.e. isolation forest with 50 trees and the autoencoder with 24 layers, and we test them as the training epochs increase from 100 to 200 and 500. The resulting MSE 95% confidence intervals are shown in Fig. 5. Clearly, the proposed approach outperforms isolation forest in every experimental setting. In addition, with most of the case studies, a higher number of epochs does not result in a lower error nor a tighter confidence interval, and this may suggest again the presence of overfitting (Caruana et al., 2001). Such results confirm 100 epochs training as good setting for the experimental results.

Finally, we assess the ability of the system to provide an interpretable and reliable output, by measuring its precision and accuracy as described in section 4. The discriminator used in our approach is also used in conjunction with isolation forest to study the impact of this component. In addition, the autoencoder reconstruction error is also treated with the 3sigma approach for comparison purposes. Compared to the 3sigma (Table 5), the discriminator greatly improves the precision of the detection with our approach and with isolation forest, both in terms of mean and confidence interval width. The same applies to the accuracy, which is largely improved by combining isolation forest with the discriminator. Moreover, as a confirmation of the effectiveness of the discriminator, please note that the accuracy of isolation forest and autoencoder are almost comparable when both of them employ the discriminator proposed in this work. Finally, given the performance of the discriminator, its adoption can replace the threshold refinement loop carried by the data scientist (Fig. 1).

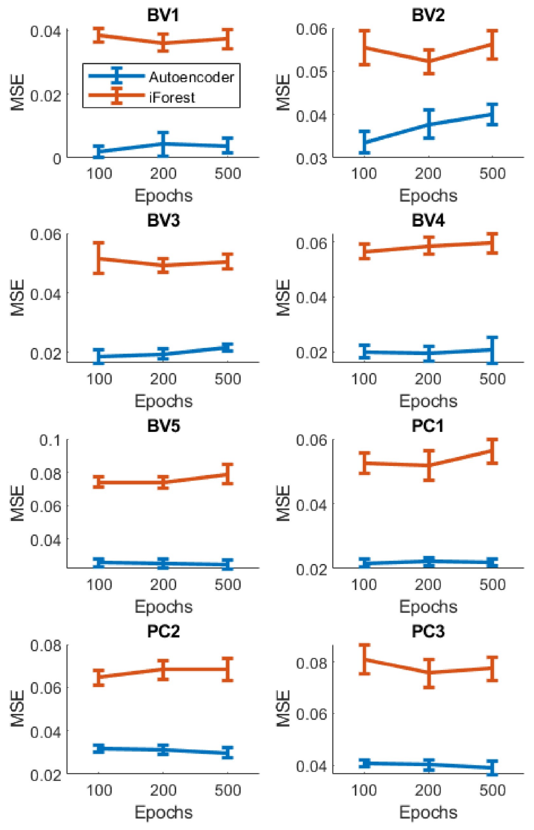

Fig. 5. 95% confidence interval of the MSE for each case study by varying the training epochs.

6. Conclusion and Future Work

In this work, we present a general, efficient, and accurate anomaly detection approach, aimed at reducing the human-driven post-processing needed to improve the reliability and interpretability of the detection. Our approach employs an autoencoder to extract a score of the anomaly degree of a time series. This is further processed with a heuristics-based discriminator aimed at rescaling such a score between 0 (regular behavior) and 1 (anomaly).

This approach can easily support operators with no data science background, since the interpretation of the resulting anomaly score is straightforward, reliable, and does not require the intervention of a data scientist for post-processing. Moreover, the short time required for training the system results in low management costs associated with system re-training due to the varying production cycles of a real-world manufacturing plant, e.g. to model the machine's behavior while making a new product. The proposed anomaly discriminator is able to improve the precision and accuracy of approaches other than ours. Finally, with respect to isolation forest, a state-of-the-art algorithm for anomaly detection, our approach results more accurate and computationally convenient despite (i) the dataset on which it is tested, e.g. power consumption of industrial laundry assets or bearing vibrations, and (ii) the complexity of the model, e.g. number of layers and neurons for the autoencoder, number of trees for isolation forest. Given such promising results, as future works, we aim at extending our approach to (i)

Table 3. Confidence Interval of the detection MSE with the proposed approach (AE+DS) and isolation forest, by using 100 training epochs.

Approach	iForest	AE+DS	iForest	AE+DS	iForest	AE+DS
Complexity	~ 550 nodes	336 links	~ 1375 nodes	1344 links	~ 2750 nodes	2696 links
BV1	0.049 ± 0.008	0.002 ± 0.002	0.044 ± 0.006	0.002 ± 0.004	0.039 ± 0.002	0.0004 ± 0.00
BV2	0.071 ± 0.011	0.034 ± 0.003	0.063 ± 0.004	0.035 ± 0.002	0.055 ± 0.004	0.035 ± 0.001
BV3	0.068 ± 0.009	0.019 ± 0.002	0.061 ± 0.006	0.021 ± 0.002	0.052 ± 0.005	0.023 ± 0.003
BV4	0.077 ± 0.010	0.020 ± 0.002	0.066 ± 0.008	0.024 ± 0.003	0.057 ± 0.003	0.028 ± 0.003
BV5	0.100 ± 0.010	0.026 ± 0.003	0.086 ± 0.006	0.029 ± 0.002	0.074 ± 0.003	0.029 ± 0.002
PC1	0.075 ± 0.007	0.022 ± 0.001	0.062 ± 0.008	0.021 ± 0.001	0.053 ± 0.003	0.020 ± 0.001
PC2	0.080 ± 0.009	0.032 ± 0.002	0.069 ± 0.007	0.031 ± 0.002	0.065 ± 0.004	0.029 ± 0.001
PC3	0.093 ± 0.013	0.041 ± 0.001	0.081 ± 0.008	0.042 ± 0.002	0.081 ± 0.006	0.044 ± 0.002

Table 4. Confidence Interval of the training time (in seconds) obtained with the proposed approach (AE+DS) and isolation forest, with 100 training epochs.

Approach	iForest	AE+DS	iForest	AE+DS	iForest	AE+DS
Complexity	~ 550 nodes	336 links	~ 1375 nodes	1344 links	~ 2750 nodes	2696 links
BV1	6.3 ± 0.3	6.7 ± 0.2	16.5 ± 0.8	8.0 ± 0.2	32.7 ± 0.7	10.7 ± 0.2
BV2	7.5 ± 0.3	7.0 ± 0.2	20.1 ± 1.8	8.4 ± 0.4	38.7 ± 0.6	11.8 ± 0.3
BV3	7.2 ± 0.2	7.3 ± 0.2	19.4 ± 0.7	8.9 ± 0.3	37.9 ± 0.6	12.9 ± 0.3
BV4	7.6 ± 0.3	7.5 ± 0.2	19.8 ± 0.4	9.2 ± 0.2	39.8 ± 0.9	14.1 ± 0.3
BV5	7.6 ± 0.2	7.6 ± 0.3	19.8 ± 0.4	9.3 ± 0.2	40.0 ± 1.2	15.7 ± 0.7
PC1	7.4 ± 0.2	9.6 ± 0.3	19.3 ± 0.3	13.1 ± 0.4	39.3 ± 1.1	24.3 ± 2.8
PC2	7.4 ± 0.2	9.6 ± 0.2	19.5 ± 0.5	13.3 ± 0.3	39.1 ± 1.1	21.1 ± 0.9
PC3	8.2 ± 0.2	9.8 ± 0.2	21.2 ± 0.3	14.9 ± 1.3	43.1 ± 1.1	21.7 ± 0.6

Table 5. Anomaly detection precision and accuracy with 3σ rule and the heuristic-based discriminator (DS).

Metr.	Precision				Accuracy			
Appr.	iForest		Autoencoder		iForest		Autoencoder	
+	3σ rule	DS	3σ rule	DS	3σ rule	DS	3σ rule	DS
BV1	0.464 ± 0.031	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.946 ± 0.007	0.987 ± 0.001	1.000 ± 0.000	1.000 ± 0.000
BV2	0.379 ± 0.015	0.984 ± 0.024	0.584 ± 0.017	1.000 ± 0.000	0.925 ± 0.005	0.965 ± 0.002	0.962 ± 0.002	0.960 ± 0.002
BV3	0.430 ± 0.023	1.000 ± 0.000	0.930 ± 0.035	1.000 ± 0.000	0.938 ± 0.006	0.976 ± 0.001	0.996 ± 0.002	0.968 ± 0.003
BV4	0.381 ± 0.017	0.993 ± 0.016	0.607 ± 0.015	1.000 ± 0.000	0.926 ± 0.005	0.970 ± 0.001	0.968 ± 0.001	0.960 ± 0.002
BV5	0.303 ± 0.016	0.835 ± 0.028	0.396 ± 0.015	0.917 ± 0.129	0.897 ± 0.006	0.967 ± 0.002	0.925 ± 0.005	0.960 ± 0.003
PC1	0.399 ± 0.017	1.000 ± 0.000	0.910 ± 0.058	1.000 ± 0.000	0.920 ± 0.006	0.980 ± 0.003	0.980 ± 0.002	0.977 ± 0.001
PC2	0.386 ± 0.017	0.946 ± 0.029	0.796 ± 0.062	1.000 ± 0.000	0.917 ± 0.006	0.965 ± 0.002	0.976 ± 0.005	0.966 ± 0.002
PC3	0.282 ± 0.013	0.945 ± 0.054	0.266 ± 0.041	1.000 ± 0.000	0.882 ± 0.005	0.954 ± 0.001	0.893 ± 0.015	0.953 ± 0.004

obtain a predictive maintenance solution, i.e. able to detect the progressive deterioration of industrial assets, and (ii) employ a transfer learning mechanism, in order to exploit the anomaly detector with data other than the ones it is trained on (e.g. different data of the same domain), and further reducing the costs linked to system re-training whenever possible.

Acknowledgments

This work is partially supported by (i) the Tuscany Region in the framework of the GSSMART project, approved with Law Decree 6695 31-12-2014, and (ii) the Italian Ministry of Education and Research (MIUR) in the framework of the CrossLab project (Departments of Excellence).

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al., 2016. Tensorflow: A system for large-scale machine learning, in: 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16), pp. 265–283.

Aggarwal, C.C., 2015. Outlier analysis, in: Data mining, Springer. pp. 237–263.

Amer, M., Goldstein, M., Abdennadher, S., 2013. Enhancing one-class support vector machines for unsupervised anomaly detection, in: Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description, ACM. pp. 8–15.

Bechhoefer, E., 2013. Condition based maintenance fault database for testing diagnostics and prognostic algorithms. `https://mfpt.org/fault-data-sets/`. [Online accessed Oct-2019].

Caruana, R., Lawrence, S., Giles, C.L., 2001. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping, in: Advances in neural information processing systems, pp. 402–408.

Chandola, V., Banerjee, A., Kumar, V., 2009. Anomaly detection: A survey. ACM computing surveys (CSUR) 41, 15.

Cimino, M.G., Lazzeri, A., Vaglini, G., 2015. Improving the analysis of context-aware information via marker-based stigmergy and differential evolution, in: International Conference on Artificial Intelligence and Soft Computing, Springer. pp. 341–352.

David, S., Harald, S., Jose Luis Galvez, M., 2013. Optimised large-scale or outsourced laundry operations. `http://ec.europa.eu/environment/emas/takeagreenstep/pdf/BEMP-5.5-FINAL.pdf`. [Online accessed Oct-2019].

Fan, C., Xiao, F., Zhao, Y., Wang, J., 2018. Analytical investigation of autoencoder-based methods for unsupervised anomaly detection in building energy data. Applied energy 211, 1123–1135.

Galatolo, F.A., Cimino, M.G.C., Vaglini, G., 2018. Using stigmergy to incorporate the time into artificial neural networks, in: International Conference on Mining Intelligence and Knowledge Exploration, Springer. pp. 248–258.

Ghosh, A., Qin, S., Lee, J., Wang, G.N., 2016. Fbmtp: An automated fault and behavioral anomaly detection and isolation tool for plc-controlled manufacturing systems. IEEE Transactions on Systems, Man, and Cybernetics: Systems 47, 3397–3417.

Goldstein, M., Uchida, S., 2016. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. PloS one 11, e0152173.

Humphrey, S., Papadopoulos, H., Linke, B., Maiyya, S., Vijayaraghavan, A., Schmitt, R., 2014. Power measurement for sustainable high-performance manufacturing processes. Procedia CIRP 14, 466–471.

IEEA, 2010. The carbon trust industrial energy efficiency accelerator (ieea), executive summary. `https://www.carbontrust.com/media/206508/ctg064-laundries-industrial-energy-efficiency.pdf`. [Online accessed Oct-2019].

Jin, X., Sun, Y., Que, Z., Wang, Y., Chow, T.W., 2016. Anomaly detection and fault prognosis for bearings. IEEE Transactions on Instrumentation and Measurement 65, 2046–2054.

Kathrin, G., Markus, B., Eva, B., Carl-Otto, G., Ina, R., Shailendra, M., Raul, C., Thibault, F., Lyons, L., 2011. Washing machines and dryers. part 4: Technical analysis of existing products. `https://www.eup-network.de/fileadmin/user_upload/Produktgruppen/Lots/Working_Documents/EuP_Lot24_Wash_T4_Report_ENER_clean_01.pdf`. [Online accessed Oct-2019].

Kusiak, A., 2017. Smart manufacturing must embrace big data. Nature 544, 23–25.

Li, G., Hu, Y., Chen, H., Li, H., Hu, M., Guo, Y., Liu, J., Sun, S., Sun, M., 2017. Data partitioning and association mining for identifying vrf energy consumption patterns under various part loads and refrigerant charge conditions. Applied energy 185, 846–861.

Lindemann, B., Fesenmayr, F., Jazdi, N., Weyrich, M., 2019. Anomaly detection in discrete manufacturing using self-learning approaches. Procedia CIRP 79, 313–318.

Liou, C.Y., Huang, J.C., Yang, W.C., 2008. Modeling word perception using the elman network. Neurocomputing 71, 3150–3157.

Liu, F.T., Ting, K.M., Zhou, Z.H., 2012. Isolation-based anomaly detection. ACM Transactions on Knowledge Discovery from Data (TKDD) 6, 3.

Lopez, F., Saez, M., Shao, Y., Balta, E.C., Moyne, J., Mao, Z.M., Barton, K., Tilbury, D., 2017. Categorization of anomalies in smart manufacturing systems to support the selection of detection mechanisms. IEEE Robotics and Automation Letters 2, 1885–1892.

Ouyang, Z., Sun, X., Chen, J., Yue, D., Zhang, T., 2018. Multi-view stacking ensemble for power consumption anomaly detection in the context of industrial internet of things. IEEE Access 6, 9623–9631.

Paulheim, H., Meusel, R., 2015. A decomposition of the outlier detection problem into a set of supervised learning problems. Machine Learning 100, 509–531.

Riazi, M., Zaiane, O., Takeuchi, T., Maltais, A., Günther, J., Lipsett, M., 2019. Detecting the onset of machine failure using anomaly detection methods, in: International Conference on Big Data Analytics and Knowledge Discovery, Springer. pp. 3–12.

Saltz, J.S., Grady, N.W., 2017. The ambiguity of data science team roles and the need for a data science workforce framework, in: 2017 IEEE International Conference on Big Data (Big Data), IEEE. pp. 2355–2361.

Simon, V., Johansson, C.A., Galar, D., 2017. Aggregation of electric current consumption features to extract maintenance kpis. Management Systems in Production Engineering 25, 183–190.

Skoogh, A., Johansson, B., Hansson, L., 2011. Data requirements and representation for simulation of energy consumption in production systems, in: Proceedings of the 44th CIRP Conference on Manufacturing Systems, pp. 1–3.

Sun, H., Jin, R., Luo, Y., 2019. Supervised subgraph augmented non-negative matrix factorization for interpretable manufacturing time series data analytics. IISE Transactions , 1–12.

Wuest, T., Weimer, D., Irgens, C., Thoben, K.D., 2016. Machine learning in manufacturing: advantages, challenges, and applications. Production & Manufacturing Research 4, 23–45.

Ye, Z., Guang-Tong, Z., 2017. Isolation forest. `https://github.com/zhuye88/iForest/`. [Online accessed Nov-2019].

Zenisek, J., Wolfartsberger, J., Sievi, C., Affenzeller, M., 2018. Streaming synthetic time series for simulated condition monitoring. IFAC-PapersOnLine 51, 643–648.

Zhang, M., Krintz, C., Mock, M., Wolski, R., 2019a. Seneca: Fast and low cost hyperparameter search for machine learning models, in: 2019 IEEE 12th International Conference on Cloud Computing (CLOUD), IEEE. pp. 404–408.

Zhang, W., Yang, D., Wang, H., 2019b. Data-driven methods for predictive maintenance of industrial equipment: A survey. IEEE Systems Journal .

Zhao, P., Kurihara, M., Tanaka, J., Noda, T., Chikuma, S., Suzuki, T., 2017. Advanced correlation-based anomaly detection method for predictive maintenance, in: 2017 IEEE International Conference on Prognostics and Health Management (ICPHM), IEEE. pp. 78–83.

Zhu, A., Meng, Y., Zhang, C., 2017. An improved adam algorithm using lookahead, in: Proceedings of the 2017 International Conference on Deep Learning Technologies, ACM. pp. 19–22.