

A Comparison of Anomaly Detection Methods for Industrial Screw Tightening

Diogo Ribeiro[1], Luís Miguel Matos[1], Paulo Cortez[1(✉)], Guilherme Moreira[2], and André Pilastri[3]

[1] ALGORITMI R&D Centre, Department of Information Systems, University of Minho, 4804-533 Guimarães, Portugal
id6336@alunos.uminho.pt, {luis.matos,pcortez}@dsi.uminho.pt
[2] Bosch Car Multimedia, Braga, Portugal
Guilherme.Moreira2@pt.bosch.com
[3] EPMQ - IT CCG ZGDV Institute, 4804-533 Guimarães, Portugal
andre.pilastri@ccg.pt

Abstract. Within the context of Industry 4.0, quality assessment procedures using data-driven techniques are becoming more critical due to the generation of massive amounts of production data. In this paper, we address the detection of abnormal screw tightening processes, which is a relevant industrial task. Since labeling is costly, requiring a manual effort, we focus on unsupervised approaches. In particular, we assume a low-dimensional input screw fastening approach that is based only on angle-torque pairs. Using such pairs, we explore three main unsupervised Machine Learning (ML) algorithms: Local Outlier Factor (LOF), Isolation Forest (iForest) and a deep learning Autoencoder (AE). For benchmarking purposes, we also explore a supervised Random Forest (RF) algorithm. Several computational experiments were held by using recent industrial data with 2.8 million angle-torque pair records and a realistic and robust rolling window evaluation. Overall, high quality anomaly discrimination results were achieved by the iForest (99%) and AE (95% and 96%) unsupervised methods, which compared well against the supervised RF (99% and 91%). When compared with iForest, the AE requires less computation effort and provides faster anomaly detection response times.

Keywords: Autoencoder · Deep learning · Industry 4.0 · Isolation Forest · One-class classification · Random Forest · Unsupervised learning

1 Introduction

The current competition market increases the pressure for industrial companies to improve their productive processes (e.g., increasing efficiency and reducing costs). Within this context, a key aspect is the reduction of assembly errors during the production of products. Following the Industry 4.0 revolution, most

© Springer Nature Switzerland AG 2021
O. Gervasi et al. (Eds.): ICCSA 2021, LNCS 12950, pp. 485–500, 2021.
https://doi.org/10.1007/978-3-030-86960-1_34

modern factories make use of automation and robots that are interconnected with data sensors. While fully autonomous robots are used in some production plants, there are still industrial tasks that require a human operator. In effect, several companies use assembly machines that combine the flexibility of robotic arms with the guidance of human operators. In particular, handheld screwdrivers are often used in assembly factories, presenting the advantage of an automatic adaption to different torque profiles. During the screw tightening process, data is collected in real-time, generating several instances with multiple features, including angle-torque pairs. Once the operation is finished, the full screw response curve is presented to the operator, who needs to accept or reject it (due to lack of quality issues) in a small amount of time. While defects are rare in this domain, they are related with a diverse range of situations, such as floating screws, stripped screws and other situations. Often, the defect is detected too late down the production chain, which results in extra production times and costs.

A common approach to detect industrial screw tightening anomalies is to use a defect catalog that contains a curated set of normal and failure examples. Each time a new screw tightening is executed, the obtained angle-torque curve is compared with a predefined set of rules. However, this expert system detection method is rather static, requiring a manual collection and labeling of data. Thus, there is a potential to automate and improve the detection task by using Machine Learning (ML) algorithms. Within our knowledge, this approach has been scarcely studied. The research work more closely related with our approach was based on a Long Short-Term Memory (LSTM) neural network that uses annotated angle-torque screw responses [5]. Yet, such LSTM assumes a supervised learning that is rather impractical in several industrial settings. In effect, labeled data is costly (involving manual effort) and assembly companies typically produce big data. Thus, using a unsupervised learning for industrial screw tightening anomaly detection would result in a more valuable and automated approach.

One of the challenges that anomaly detection has to address is that boundaries between normal data and abnormal data are often not clearly defined [4]. Moreover, the scarcity and diversity of anomalous data makes it a non trivial task that is typically addressed by using an unsupervised or one-class learning [19]. Under the one-class learning approach, the training datasets only contain "normal" data points and the anomaly (often termed outlier) is detected when new data is considered distant from the training learning space.

In this paper, we explore a low-dimensional input approach for the anomaly detection of industrial screw tightening processes that only assumes angle-torque pairs. To model the data, we adapt three main unsupervised algorithms that are only trained with normal screw tightening process instances (thus one-class): Local Outlier Factor (LOF), Isolation Forest (iForest) and a deep learning Autoencoder (AE). For comparison purposes, we also test a supervised (two-class) Random Forest (RF). To evaluate the learning methods, we use a dataset with 2.8 million angle-torque pairs that were recently collected from an auto-

motive multimedia assembly company. Several computational experiments were conducted by using a realistic rolling window evaluation that simulates several training and testing iterations through time. For each iteration, both the anomaly detection performance and the computational effort were measured.

This paper is organized as follows. Section 2 presents the related work. Next, Sect. 3 describes the collected industrial assembly data, anomaly detection learning models and evaluation procedure. Then, Sect. 4 presents the experimental results. Finally, Sect. 5 discusses the main conclusions and future work directions.

2 Related Work

The topic of anomaly detection has been researched in several fields of study [6]. It is often regarded as a one-class task, since anomalous patterns are typically scarce in most datasets. Several learning methods have been proposed for an unsupervised anomaly detection, such as [1,3]: Local Outlier Factor (LOF), One-Class Support Vector Machine (OC-SVM) and Isolation Forest (iForest). Following the success of Deep Learning, Autoencoders (AE) have been increasingly adopted for anomaly detection tasks [21]. When compared with conventional methods (e.g., LOF, OS-SVM and IF), AEs tend to provide faster training times, thus are capable of handling a larger amount of training data. In particular, the AE model can be updated to include new instances without retraining the full model [14], which is highly relevant when high velocity data is generated.

Regarding the specific screwing tightening use case, the research literature involving ML approaches is scarce. For the automatic identification of behaviour patterns on blind fasteners installation, [7] presents a kernel density-based pattern methodology to classify good or bad examples. In [18], Support Vector Machine (SVMs) with different kernel functions (e.g., linear and polynomial) were used to monitor screw fastening processes on the cover of hard disks based on the driver motor data. The obtained data-driven models were then used to detect incomplete screwing processes. This work differs from ours as it correlates motor power on a time series instead of the actual handled screw driver output (e.g., angle and torque). In [15], another data-driven analysis was conducted on thread fastening tasks that used a robotic manipulator. The problem was addressed by a supervised SVM classifier, aiming to estimate how much of the screw has been inserted by using the thread vibration from a torque sensor. As explained in Sect. 1, the most similar related work was performed by Cao et al. [5], where a supervised learning LSTM was used to infer the quality of the assembly by analyzing the screwing tightening angle-torque curve. The main novelty of our approach is that we use a pure unsupervised one-class learning, which is more suited for high volume industrial screwing data.

3 Materials and Methods

3.1 Industrial Data

Our case study is related with a major automotive multimedia assembly company, where a new fastening process starts with the insertion of several sub-parts (the total number varies from product to product) into the assembly jig. Then a scanner reads the imprinted barcode and the machine automatically loads the screw tightening program. Next, the human operator proceeds with an alignment of the matching pair and initiates the task. Using the help of the handheld screwdriver, the operator is guided to fasten each screw in a specific order, leaving the remaining technical details (e.g., torque and angle) to be handled automatically by the machine. Screw after screw, the operator is presented with a "Good Or Fail" (GOF) result, which is calculated automatically by the assembly machine based on its internal configuration. Each time the tightening is finished, the assembly system generates a local `.csv` file with all process details. All files are then imported into a centralized big data server cluster. The new data is then made available via the export of a `.json` file. Until this point, we have no control over the generated data stream.

In this work, we have designed a Python virtual environment that is capable of retrieving the industrial data by using Apache Spark queries to the central cluster. The raw data variables of the retrieved data are summarized in Table 1. Let i denote a screw fastening procedure for some product unit, where $i \in \{1, 2, ..., N\}$ and N represents the total number of screw tightening processes. For each i process, there is a real-time generation of several $k \in \{1, 2, 3, ..., K_i\}$ values, where K_i denotes the total number of observations and each product unit can produce a different K_i value. For a particular i and k value, the machine automatically registers the angle ($\alpha_{i,k}$) and torque ($\tau_{i,k}$) measurements. Typically, each screw fastening produces hundreds of angle-torque pairs, such as exemplified in Fig. 1. It should be highlighted that while there is no direct temporal variable, the angle ($\alpha_{i,k}$) attribute can be used as sequential or temporal measure of the fastening, since in almost cases its value increases as the procedure continues.

A tightening process is composed of four main steps $s \in \{1, 2, 3, 4\}$ (Table 2). In order to succeed at combining sub-parts, every screw must sequentially meet the transition conditions of each step within the parameterized time. The steps are exemplified in Fig. 1, where the short blue curve denotes the initial rotation (step s =0) used to catch the screw and the remainder curve colors ($s = 1$ – red, $s = 2$ – green, $s = 3$ –purple and $s = 4$ – orange) denote the four main fastening steps. For each step, the screw fastening tool stores the step number, total angle and torque values. At the beginning of step s =4, the fastening tool computes an estimate of the *DTM Clamping Angle* (*dtmclampt*). Once the process is nearly finishing, the real *DTM Clamping Angle* is measured (*dtmclampa*). Then, the final GOF result is computed by the screw assembly machine (attribute *screwgof*).

Table 1. Description of the industrial screw tightening data variables.

Variable	Description
	Hundreds of values per screw fastening (k):
angle ($\alpha_{i,k}$)	Value for the angle (e.g., $-208.1°$)
torque ($\tau_{i,k}$)	Value for the torque (e.g., 35.6 Ncm)
gradient	Torque gradient for two consecutive angle values (e.g., 20)
	Four values per fastening (one for each step):
stepnr	Screwing step number ($s \in \{1,2,3,4\}$)
tstepangle	Total step angle (e.g., 990°)
tsteptorque	Total step torque (e.g., 39.3)
	One value per fastening (i):
dtmclampa	Value for actual DTM Clamp (e.g., 79.1)
dtmclampt	Value for predicted DTM Clamp (e.g., 30.2)
timestamp	Timestamp (e.g., "2020-10-08 06:30:51")
product	Product family identifier (e.g., "2222111")
serial	Product serial number (e.g., "11111")
screwnr	Screw number ($\in \{1, 2, ..., 8\}$)
screwenergy	Total energy required (e.g., 19959)
totalangle	Total angle (e.g., 2993°)
totaltorque	Total torque (e.g., 30000 Ncm)
screwgof (y_i)	Screwing process "Good" (1) or "Fail" (0)

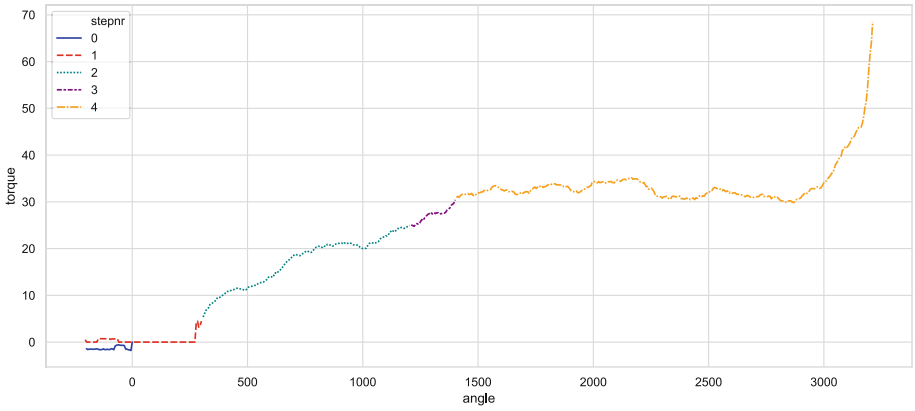

Fig. 1. Example of a normal ("Good") screw tightening.

In this work, we adopt the last variable (*screwgof* attribute) as the supervised label of the screwing process. Thus, this variable, denoted as y_i for the i-th screw tightening process, is used to select the instances for learning methods (e.g., only normal examples are fed into the one-class methods) and also to define

Table 2. Description of the screw fastening process steps.

Step	Description	Transition condition
Simple torque ($s = 1$)	Serves as a transition to the next step when the thread starts to be formed	Achieve either the transition torque or angle within the parameterized time range
Angle ($s = 2$)	Fixed number of turns conducted	Min Angle < Angle Target < Max Angle within stipulated time
Torque ($s = 3$)	Apply a fixed torque value	Min Torque < Torque Target < Max Torque within stipulated time
Seating control ($s = 4$)	Apply a torque value from the screw seating value to the part	Clamping Torque < Max Seating Torque; Min Total Angle < Total Angle < Max Total Angle; Min Total Torque < Total Torque < Max Total Torque

the normal and abnormal labels used to evaluate the models (when using test "unseen" data). The values were directly computed by the assembly machine software tool, which includes a rather rigid internal set of rules that define a normal fastening cycle (based on a catalog of normal and abnormal examples). It should be noted that in a few cases, this assembly tool provides a wrong labeling, such as classifying a fastening cycle as normal when it actually was a defect. However, since we have big data and it is costly to manually verify all fastening cycles, we assume here that the provided y_i values are a reasonable oracle for testing the proposed anomaly detection methods (which do not use any labels).

In terms of preprocessing, we grouped all records by product family (*product*), serial number (*serial*) and screw number (*screwnr*, maximum of 8 screws per individual unit). This grouping allows a fast identification of all k values related with a tightening process (i), which is useful for anomaly detection evaluation purposes. The resulting dataset corresponds to around two days of screw tightening processes, collected in November of 2020, and it includes a total of 2,853,967 entries related with $N = 6,162$ fastening cycles. On average, there is around $\overline{K_i} = 463$ records (angle-torque pairs) per cycle. Moreover, the dataset is highly unbalanced. In effect, there are 74 defects that correspond to only 2% of the screw tightening cycles.

3.2 Anomaly Detection Angle-Torque Approach

There are two screw tightening anomaly detection goals. The first and most important goal is to automatically detect if there has been a failure in the screw fastening cycle (y_i = "Fail"). Once a failure has been detected, the secondary goal is to automatically identify where (within the full cycle) the defect took place. The secondary goal is needed to fine tune the detection details, providing

additional information that helps the human operator to accept the machine learning result.

In this work, we assume that each anomaly detection model outputs a decision score (d_i), such that the higher the score, the more probable is the chance that an anomaly has occurred for the i-th tightening screw process. In order to achieve both goals simultaneously, the anomaly detection methods are fed with the all individual and more instant k values (one at the time). This means that each i-th screw fastening produces a total of $k \in \{1, 2, ...K_i\}$ examples that are used for training or testing purposes, resulting in $d_{i,k}$ anomaly decision scores.

To compute the overall decision score d_i (thus perform the main goal), the individual scores are averaged, thus $d_i = \sum_k d_{i,k}/K_i$. For distinct i values, the resulting scores (d_i) can be compared with the target variable (y_i), allowing the computation of the Receiver Operating Characteristic (ROC) curve [8]. If class labels are needed, then a fixed Th threshold value needs to be set, such that the i-th output is interpreted as an anomaly if $d_i > Th$. The specific Th can be selected by using historical unlabeled data (e.g., by adopting a given percentage above the maximum AE reconstruction error) or by selecting the best specificity-sensitivity trade-off from a ROC curve that is generated when using a labeled validation set. As for the secondary goal, if fine-grained target values ($y_{i,k}$) were available, then the individual decision scores ($d_{i,k}$) could be used to generate ROC curves or other classification performance measures. Since our dataset does not contain any fine-grained target labels, we will adopt a fixed threshold to demonstrate the applicability of the secondary goal but without robustly studying it. Moreover, given that we use pure unsupervised ML models, the threshold will be set using historical unlabeled data (see Sect. 4).

Regarding the input features, the industrial screw tightening data includes several attributes with different granularities (Table 1): fine-grained – angle ($\alpha_{i,k}$), torque ($\tau_{i,k}$) and gradient; step-grained – step number, total step angle and torque; and one per each i-th tightening – such as *dtmclampa*, *screwenergy* and y_i. To select the input variables (and also to tune the AE models, see Sect. 3.3), we conducted preliminary experiments by considering only the training data of the first rolling window iteration, thus related with the oldest data records (Sect. 3.4). This oldest training data was further split into training (to fit the models, with $W - T$ tightening screw processes) and validation sets (with T tightening cycles). In particular, we initially tested anomaly detection models that were fed with all possible input variables (e.g., angle, torque, gradient, total step angle, energy). However, the obtained models provided substantially worst results when compared with the low-dimensional angle-torque input approach. Moreover, the full input anomaly detection models required much more memory and computational effort. In some cases, such as when using LOF, the computational training process even halted due to an out-of-memory issue. Based on these results, all screw anomaly methods described in this work assume just two input values: the simpler angle-torque pairs ($\alpha_{i,k}, \tau_{i,k}$) for each (i, k) example.

3.3 Learning Models

Three unsupervised learning algorithms were selected for the empirical comparison: LOF, iForest and a AE. These methods are only trained with normal cases (one-class approach). For benchmarking purposes, we selected the popular Random Forest (RF) method. However, we note that RF is a supervised learning method that, contrary to the one-class methods, requires labeled data for the training procedure. Thus, RF was trained with both normal and abnormal angle-torque instances.

To implement the methods we used the `scikit-learn` (for RF, LOF and iForest) [16] and `TensorFlow` (for AE) [9] Python modules. At an initial stage, we also explored the OC-SVM implementation of `scikit-learn`. However, our training sets are too large for the algorithm, which did not compute results in an affordable time (e.g., the first rolling window iteration execution was halted after 40 h of execution time). In order to provide a fair comparison, and also reduce the computational effort, in the experiments we assumed in general the default `scikit-learn` and `keras` hyperparameter values. Before feeding the models, all angle-torque data values were first normalized, where the numeric values were transformed to fit into a [0,1] scale by using a max-min normalization, via the `MinMaxScaler` procedure of the `sklearn` module.

Unlike the other anomaly detection methods, LOF is a density-based algorithm that heavily depends on K-nearest neighbors [3]. It is designed as local because it depends on how well isolated an object is from the surrounding neighborhood. Instead of classifying an object as being an outlier or an inlier, an outlier factor is assigned describing up to which degree the object differs from the rest of the dataset. It should be noted that LOF can be used to detect both local and global outliers. Moreover, it can be trained with multi-class instances. In this paper, in order to provide a fair comparison, we only use normal examples to fit the model, where the outlier factor score is directly used as the anomaly decision score ($d_{i,k}$).

Isolation Forests (iForest) take advantage of two major characteristics of anomalous points: they are present in fewer quantities and are also numerically different to normal instances. This means that anomalous instances are prone to be more easily isolated (separated from the rest of the instances) than normal points. Using this principle, an anomaly detection method can be based on the construction of a tree structure to isolate every single instance and then evaluate their normality. Due to their nature and tendency to be quickly isolated, abnormal instances are closer to the root of the tree. This characteristic, also known as Isolation Tree or iTree, forms the basis of anomaly detection for this model. An iForest [13] is based on an ensemble of iTrees for a given dataset. Anomalies are the instances that are closer to the root of the tree and will have shorter average path lengths. The `scikit-learn` provides an anomaly score that ranges from $\hat{y}_{i,k} = -1$ (highest abnormal score) to $\hat{y}_{i,k} = 1$ (highest normal score). Thus, we rescaled these scores into the [0, 1] anomaly probability range by computing $d_{i,k} = (1 - \hat{y}_{i,k})/2$.

Autoencoders (AE) are unsupervised learning techniques that efficiently compresses and encode data to a lower-dimensional representation [10]. The includes an encoding stage, where compression occurs by reducing the input data (the number of features describing the dataset) into a latent space (defined by a bottleneck layer), and a decoding stage, where the model tries to reconstruct the original data from the latent space. When adapted for our anomaly detection case, the training algorithm is only fed with normal instances and it attempts to generate two output values $(\hat{\alpha}_{i,k}, \hat{\tau}_{i,k})$ that are identical to its inputs $(\alpha_{i,k}, \tau_{i,k})$. When a new (i,k) instance is evaluated, we compute the Mean Absolute Error (MAE) [1]:

$$MAE_{i,k} = (|\alpha_{i,k} - \hat{\alpha}_{i,k}| + |\tau_{i,k} - \hat{\tau}_{i,k}|)/2 \quad (1)$$

This reconstruction error is used as the decision score $d_{i,k} = MAE_{i,k}$, where higher reconstruction errors should denote a higher anomaly probability. In terms of the deep AE architecture, we assume a dense AE with two inputs $(\alpha_{i,k}, \tau_{i,k})$, two output nodes and a fully-connected structure that includes a stack of layers in both the encoder and decoder components. All hidden layers use the popular ReLU activation function, while the output nodes assume the logistic function (all inputs are normalized $\in$[0,1]). Every hidden layer is also attached with a *Batch Normalization* layer, which reduces the internal covariate shift, discards the need of dropout layers and normalizes the layer inputs for each batch of data that passes through it [12]. In order to define the number of layers and units per layer, we performed several trial-and-error preliminary experiments. These experiments assume the same oldest training data used to set the input variables (Sect. 3.2). The tested architectures use a bottleneck layer with 1 hidden node (to define the latent space) and a varying range of hidden layers. The best results, measured in terms of the lowest reconstruction error for the validation set, were achieved for an AE with 20 layers (including the input, Batch Normalization and output layers), as shown in Fig. 2. In this work, the selected AE is trained with the Adam optimization algorithm using the MAE loss function, a total of 100 epochs and early stopping (with 5% of the training data being used as the validation set).

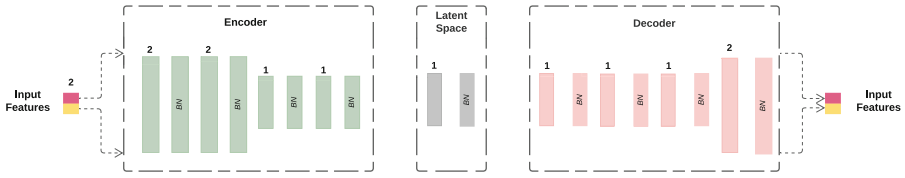

Fig. 2. The adopted dense AutoEncoder (AE) structure (the numbers denote the number of hidden units per layer and the term BN represents a batch normalization layer).

As explained in Sect. 2, the AE model can be adapted to dynamic environments when there are frequent data updates. In [14], two neural network learning modes were compared: reset and reuse. When new data arrives and the neural

network is retrained, the former assumes a random weight initialization, while the latter uses the previously trained weights as the initial set of weights. In this work, we compare both learning modes when executing the rolling window evaluation.

Finally, RF is a supervised learning algorithm that works as an ensemble of decision trees that form a "forest". Each tree depends on the values of a randomly sampled input vector and a bagging selection of training samples [2]. RF can be used for both regression and classification tasks. In this work, we used RF to output anomaly class probabilities ($\in$[0.0,1.0]), which are used as the decision score ($d_{i,k}$).

3.4 Evaluation

To compare the anomaly detection ML models, we execute the realistic rolling window procedure [14,20]. This procedure is more robust than the popular random holdout train-test split, since it realistically simulates an anomaly detection through time, producing several training and test updates, as exemplified in Fig. 3. A fixed training window with W examples is first set. Then, in the first iteration ($u = 1$), the model is trained with the oldest W instances. Once the model is fit, it performs T ahead predictions. Then, it simulates the passage of time by "rolling" the training and test sets by assuming a temporal jump step (S). The oldest S instances are discarded from the training set, being replaced by S more recent ones. A new model is then fit, allowing to perform T new subsequent predictions, and so on. In total, there are $U = (D - (W + T))/S$ model updates (training and test iterations), where D is the total dataset size. To split the data, in this work we adopt the screw tightening granularity, where all k instances for a particular i tightening process are always kept together, thus $D = N = 6,162$. After consulting the industrial experts, we adopted the values $W = 5,000$, $T = 500$ and $S = 33$, which produces a total of $U = 20$ model updates.

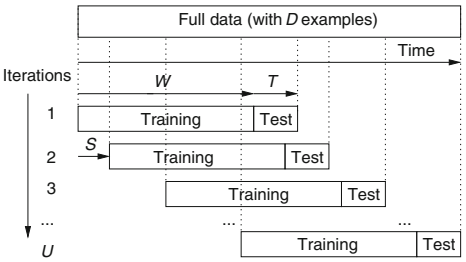

Fig. 3. Schematic of the adopted rolling window procedure.

To evaluate the anomaly detection methods, we adopt the popular ROC [8], which is computed by using the target labels (anomaly or normal) and the predicted anomaly decision scores ($d_{i,k}$) for the T tightening test examples of each u rolling window iteration. The ROC represents the full discrimination

performance of a binary classifier when considering all possible Th threshold values, plotting one minus the specificity (x-axis), also known as False Positive Rate (FPR), versus the sensitivity (y-axis) or True Positive Rate (TPR). The overall discrimination quality is measured by the Area Under the Curve $AUC = \int_0^{Th_{max}} ROC dTh$, where Th_{max} denotes the maximum threshold value (in this work, $Th_{max} = 1$ for the iForest, AE and RF methods). This AUC metric has two main advantages [17]. Firstly, when the data is highly unbalanced (which is our case), the interpretation of the goodness of the metric values does not change. Secondly, the quality of the AUC values can be interpreted as: 50% - performance of a random classifier; 60% - reasonable; 70% - good; 80% - very good; 90% - excellent; and 100% - perfect. Given that the rolling window produces several ROC curves and AUC values (one for each iteration), we aggregate the results by computing the median (AUC, training and inference time) and average (AUC) values, using the Wilcoxon non parametric statistic [11] to check if paired AUC differences are statistically significant. All experiments were executed using a 2.3 GHz Intel Core i9 processor. Besides the predictive decision scores, for each anomaly detection method we recorded the computational effort, measured in terms of the average (over all u iterations) total training and anomaly screw tightening inference times (both in s).

4 Results

Table 3 summarizes the rolling window evaluation results, while Fig. 4 shows the evolution of AUC values through the rolling window iterations. The supervised method (RF) provides a high median and average AUC scores (99% and 91%), which sets a high comparison standard for the unsupervised one-class methods. Regarding LOF, it corresponds to the fastest training method, requiring an average of 18.3 s for each rolling window iteration. However, this method also provides the worst AUC values (median of 80% and average of 75%). When comparing both AE learning modes, reuse and reset, the former provides better median and average AUC results, and lower average computational training times (explained by the usage of the early stopping procedure). The best one-class median and average results are obtained by the iForest (99%). As shown in Table 3, the median AUC differences are only significant when comparing RF, iForest or the AEs with LOF. As for the average AUC values, iForest is significantly better than other methods, while AE reuse is significantly better than LOF and AE reset. Figure 4 confirms that the two best one-class performing models continuously provide an excellent (almost perfect) discrimination along the distinct evaluation iterations. Given these results, we recommend the two unsupervised methods iForest and AE resuse, since they do not require labeled data (as RF). In particular, iForest obtains the highest discrimination capability. As for AE reuse, the method produces an almost similar performance while requiring much less computational effort. In effect, when compared with iForest, the AE reuse training is around 7 times faster. Moreover, the AE reuse anomaly inference time is half of the one required by iForest and one third of the response time required by RF.

Table 3. Rolling window screw tightening anomaly detection results (best values in **boldface** font; selected models in *italic* font).

	AUC		Train	Inference
	Median	Avg.	Time (s)	Time (s)
LOF	0.80	0.75	**18.30**	0.12
RF	**0.99***	0.91*	25.33	0.12
iForest	**0.99***	**0.99**†	168.18	0.08
AE reset	0.93*	0.92*	29.64	**0.04**
AE reuse	0.95*	0.96$^\diamond$	23.00	**0.04**

$\star$ – Statistically significant under a paired comparison with LOF.
† – Statistically significant under a paired comparison with all other methods.
$\diamond$ – Statistically significant under a paired comparison with LOF and AE reset.

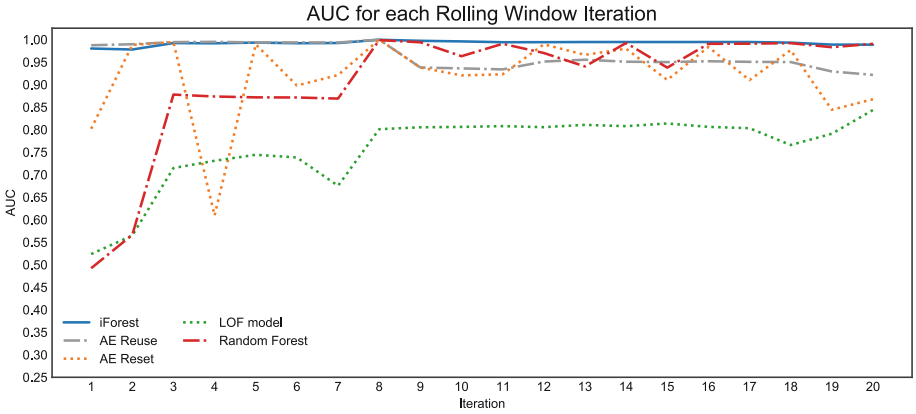

Fig. 4. AUC evolution for the screw tightening anomaly detection methods.

As an example, Fig. 5 plots the individual ROC curves for the selected models during the third rolling window iteration. The curves overlap in the sensitive (top right) region, with the AE reuse model slightly presenting a better specificity (bottom left region) in this example.

To demonstrate the secondary goal (identification of the anomaly process angle-torque regions), we selected the iForest and AE reuse models that were obtained in the same rolling window iteration and a defect example from the test set related with a torque not reached error. Then, we defined a threshold value (Th) that corresponds to the average plus one standard deviation of the training decision scores (a different Th value was used for each method). Figure 6 shows the obtained anomaly points that correspond to the highest decision scores, such that $d_{i,k} > Th$. It is interesting to note that both models signal several identical

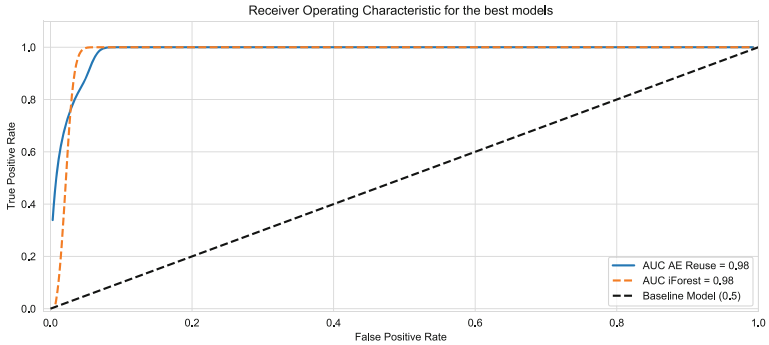

Fig. 5. ROC curves for the best two models for the rolling window iteration $u = 3$ (dashed black line denotes the performance of a random baseline classifier).

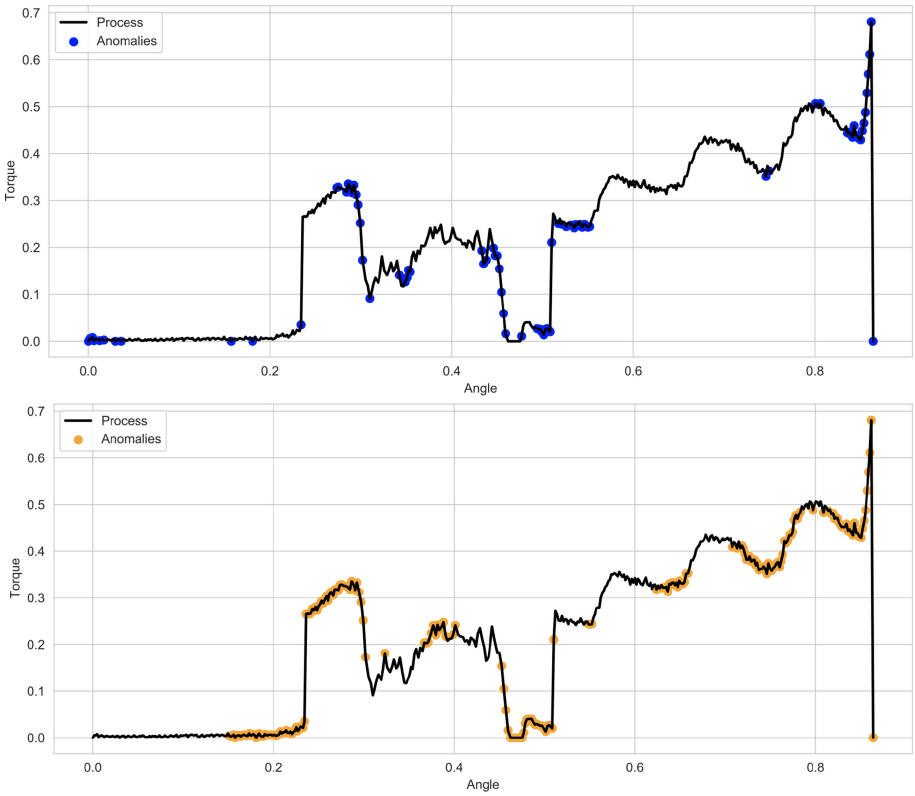

Fig. 6. Anomaly points identified by iForest (top, blue points) and AE reuse (bottom, orange points) for a screw fastening defect process. (Color figure online)

abnormal regions, such as the three abrupt torque decays (set at the normalized x-axis angle values of 0.30, 0.46 and 0.86).

The obtained results were shown to the industrial experts, who considered them very positive. Besides the high quality AUC results, the iForest and AE reuse models allowed to detect one real defect example that was labeled as "normal" by the screw assembly expert system tool. Thus, there is a potential for a better screw tightening defect detection by the proposed data-driven models. Moreover, the secondary goal demonstration example (Fig. 6) was valued as interesting and useful, particularly in the end part of the process (right $x-$axis region) when there is an abrupt torque decay. The experts highlighted that this is one of the defect behaviors. Nevertheless, they also mentioned that there are other types of screw tightening failures (at more initial stages), which will be analysed in future work.

5 Conclusions

Following the Industry 4.0 phenomenon, there is currently a vast abundance of digital data that reflect several stages of the manufacturing process. By using these data, analytic tools based on machine learning methods can potentially provide valuable production insights. In particular, within the electronic component assembly industry, the automatic detection of screw tightening processes can improve productivity and reduce costs. The assembly of screws is currently a semi-automated procedure, involving both a human operator and robotic machines. Thus, typically each tightening procedure requires a two-stage inspection, first by the machine and then by the operator. Moreover, current screw tightening failures, as identified by the machine tool, are often based on a defect catalog comparison, which uses a rigid set of rules defined by an expert system. Thus, there is room for improvement by using machine learning algorithms, creating a data-driven model that can more quickly adapt to changes in the assembly environment (e.g., assembly of new products).

In this paper, we focus on an industrial screw tightening anomaly detection task by using an unsupervised approach, which does not require any labeled data and thus it is more easy to automate. Our approach assumes that only normal (thus one-class) angle-torque input pairs are used to train the models. Using recently collected data, from an automotive electronics assembly company, we collected around 2.8 million angle-torque pairs, related with 6,162 screw fastening cycles. Four different learning algorithms were compared: unsupervised – Local Outlier Factor (LOF), Isolation Forest (iForest) and a dense Autoencoder (AE, under two learning modes: reset and reuse); and supervised - Random Forest (RF), used for benchmarking purposes. The algorithms were trained with individual angle-torque observations (two input variables), allowing to output an anomaly decision score for each angle-torque pair. The anomaly detection methods were compared by using a realistic rolling window evaluation, which simulated 20 training and test iterations through time. Overall, the best unsupervised learning results were obtained by the iForest and AE reuse. Both methods provide an excellent anomaly discrimination capability that is identical or

better than the supervised RF (median of 99% and average of 91%) benchmark. In effect, iForest produces the best unsupervised anomaly detection result (median and average of 99%), slightly outperforming the AE reuse (median of 95%, average of 96%) approach. However, it should be noted that the AE reuse method requires much less computational effort, which is relevant in this industrial domain, since it generates high velocity data.

In future work, we intend to explore more deep learning architectures, such as based on an AE with Long Short Term Memory (LSTM) layers, which can capture directly sequential data [5]. Another interesting research direction is to extend the research study by comparing the proposed data-driven approach with the screw assembly expert system tool (main goal) and also evaluate the data-driven model capability to correctly identify specific defect angle-torque regions (secondary goal), which however would require the collection of a more costly fine-grained human labeled data.

Acknowledgments. This work is supported by: European Structural and Investment Funds in the FEDER component, through the Operational Competitiveness and Internationalization Programme (COMPETE 2020) [Project n 39479; Funding Reference: POCI-01-0247-FEDER-39479]. The authors also wish to thank the automotive electronics company staff involved with this project for providing the data and the valuable domain feedback.

References

1. Alla, S., Adari, S.K.: Beginning Anomaly Detection Using Python-Based Deep Learning. Apress, Berkeley (2019). https://doi.org/10.1007/978-1-4842-5177-5_8
2. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001). https://doi.org/10.1023/A:1010933404324
3. Breunig, M.M., Kriegel, H., Ng, R.T., Sander, J.: LOF: identifying density-based local outliers. In: Chen, W., Naughton, J.F., Bernstein, P.A. (eds.) Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16–18, 2000, Dallas, Texas, USA, pp. 93–104. ACM (2000). https://doi.org/10.1145/342009.335388
4. Cao, N., Lin, Y.R., Gotz, D., Du, F.: Z-glyph: visualizing outliers in multivariate data. Inf. Vis. **17**(1), 22–40 (2018). https://doi.org/10.1177/1473871616686635
5. Cao, X., Liu, J., Meng, F., Yan, B., Zheng, H., Su, H.: Anomaly detection for screw tightening timing data with LSTM recurrent neural network. In: 15th International Conference on Mobile Ad-Hoc and Sensor Networks, MSN 2019, Shenzhen, China, December 11–13, 2019, pp. 348–352. IEEE (2019). https://doi.org/10.1109/MSN48538.2019.00072
6. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. ACM Comput. Surv. **41**(3), 15:1–15:58 (2009). https://doi.org/10.1145/1541880.1541882
7. Diez-Olivan, A., Penalva, M., Veiga, F., Deitert, L., Sanz, R., Sierra, B.: Kernel density-based pattern classification in blind fasteners installation. In: Martínez de Pisón, F.J., Urraca, R., Quintián, H., Corchado, E. (eds.) HAIS 2017. LNCS (LNAI), vol. 10334, pp. 195–206. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59650-1_17

8. Fawcett, T.: An introduction to ROC analysis. Pattern Recognit. Lett. **27**(8), 861–874 (2006). https://doi.org/10.1016/j.patrec.2005.10.010
9. Gulli, A., Pal, S.: Deep learning with Keras. Packt Publishing Ltd, Birmingham (2017)
10. Hinton, G., Salakhutdinov, R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006). https://doi.org/10.1126/science.1127647
11. Hollander, M., Wolfe, D.A., Chicken, E.: Nonparametric Statistical Methods. Wiley, Hoboken (2013)
12. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: Bach, F.R., Blei, D.M. (eds.) Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6–11 July 2015. JMLR Workshop and Conference Proceedings, vol. 37, pp. 448–456. JMLR.org (2015)
13. Liu, F.T., Ting, K.M., Zhou, Z.: Isolation forest. In: Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15–19, 2008, Pisa, Italy, pp. 413–422. IEEE Computer Society (2008). https://doi.org/10.1109/ICDM.2008.17
14. Matos, L.M., Cortez, P., Mendes, R., Moreau, A.: Using deep learning for mobile marketing user conversion prediction. In: International Joint Conference on Neural Networks, IJCNN 2019 Budapest, Hungary, July 14–19, 2019. pp. 1–8. IEEE (2019). https://doi.org/10.1109/IJCNN.2019.8851888
15. Matsuno, T., Huang, J., Fukuda, T.: Fault detection algorithm for external thread fastening by robotic manipulator using linear support vector machine classifier. In: 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, May 6–10, 2013, pp. 3443–3450. IEEE (2013). https://doi.org/10.1109/ICRA.2013.6631058
16. Pedregosa, F., et al.: Scikit-learn: Machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011). http://dl.acm.org/citation.cfm?id=2078195
17. Pereira, P.J., Cortez, P., Mendes, R.: Multi-objective grammatical evolution of decision trees for mobile marketing user conversion prediction. Expert Syst. Appl. **168**, 114287 (2021). https://doi.org/10.1016/j.eswa.2020.114287
18. Ponpitakchai, S.: Monitoring screw fastening process: an application of SVM classification. Naresuan Univ. Eng. J. (NUEJ) **11**(2), 1–5 (2016). https://doi.org/10.14456/nuej.2016.11
19. Ruff, L., et al.: Deep one-class classification. In: Dy, J.G., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10–15, 2018. Proceedings of Machine Learning Research, vol. 80, pp. 4390–4399. PMLR (2018). http://proceedings.mlr.press/v80/ruff18a.html
20. Tashman, L.J.: Out-of-sample tests of forecasting accuracy: an analysis and review. Int. J. Forecast. **16**(4), 437–450 (2000). https://doi.org/10.1016/S0169-2070(00)00065-0
21. Zhou, C., Paffenroth, R.C.: Anomaly detection with robust deep autoencoders. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13–17, 2017. pp. 665–674. ACM (2017). https://doi.org/10.1145/3097983.3098052