Investigating thresholding techniques in a real predictive maintenance scenario

Apostolos Giannoulidis and Anastasios Gounaris
Aristotle University of Thessaloniki, Greece
agiannous@csd.auth.gr
gounaria@csd.auth.gr

Nikodimos Nikolaidis and Athanasios Naskos
Atlantis Engineering, Thessaloniki, Greece
nikolaidis@atlantis-engineering.com
naskos@atlantis-engineering.com

Daniel Caljouw
Philips Consumer Lifestyle - The Netherlands
Daniel.Caljouw@philips.com

ABSTRACT

We deal with a real predictive maintenance (PdM) scenario in an Industry 4.0 setting. With a help of the Sibyl platform, we can monitor live data from key components of a Philips factory equipment; in this work, we focus on a cold-forming press. Due to the dynamic environment of the operation of this press, unsupervised anomaly detection techniques are used to timely detect the wear, where early anomalies are interpreted as warning signs of a forthcoming failure. Typically such techniques assign an anomaly score, and the problem we face is how to appropriately set a threshold for this score. We introduce and compare four generally applicable thresholding techniques, two of which are dynamic, i.e., they continuously refine the threshold during the episode lifetime. We discuss the properties of these techniques and quantitatively evaluate their behavior in our case study.

1. INTRODUCTION

Predictive maintenance (PdM) is a key component in the smartization process of manufacturing industries as Industry 4.0 advocates. The objective of PdM in an industrial setting is to detect and predict upcoming failures in crucial parts of the production line. The results of successful PdM include a safer environment for employees to operate, economical benefits, and valuable knowledge gained about the production process itself. To achieve these results, several data analytics, machine learning, and data-mining techniques are proposed. According to the data that are available, unsupervised, semi-supervised, and supervised techniques are employed.

Our case calls for unsupervised techniques since there is no labelled data available and we aim to mitigate the need for expert knowledge as much as possible. Moreover, the operation is taking place in a dynamic environment in the sense that, in each production episode, the equipment is configured differently. To date, there is no known way to leverage historical data from previous episodes to tune the running production episode. The unsupervised learning techniques that we employ perform continuous anomaly detection and as such, they rely on computing anomaly (i.e. dissimilarity) scores, which correspond to the state of the monitoring component. In PdM, an anomaly should be detected not only when there is a fault in the equipment but also when there are warning signs that a fault is going to occur shortly. However, to transform the anomaly score into a decision, an appropriate threshold is set. The threshold parameter is crucial for most of the techniques used in anomaly (or deviation) detection [23]. Setting such a threshold is not trivial. In real industrial environments, since there is no opportunity to perform exhausting tests to decide threshold values or a constant threshold may be deprecated after some period of time due to external changes, the need for self-tuning and dynamic thresholding techniques arises.

The contribution of this work can be summarized in three items:

1. We present a baseline and three additional thresholding techniques, one of which is a novel proposal of ours (called *self-tuning*), while the other two are based on existing techniques that are extended to become applicable in our setting.

2. We evaluate the presented techniques in a real-world PdM case.

3. Finally, we discuss the inherent properties of the investigated techniques.

An earlier version of this work has appeared in [7]. In this manuscript, apart from clarifications and small additions, we present new experimental results considering more episodes that have become available.

Finally, we note that PdM is considered an end-to-end procedure, which may involve IoT technologies, data collection, data cleaning and general preprocessing, model building and inference, and decision support systems (DSS) apart from anomaly detection. Each of these modules can be implemented by different services as done by Sibyl platform [17]. Here, we focus on the anomaly detection module exclusively. A key characteristic is that this module in our setting is not restricted to fault diagnosis but, essentially, it performs prognosis. This is due to the fact that we aim to detect anomalies in the operational data that precede serious faults. However, the proposed solution also relies on a DSS post-processing system to trigger maintenance actions in a judicious manner based on the reported anomalies. Such a system is out of our scope.

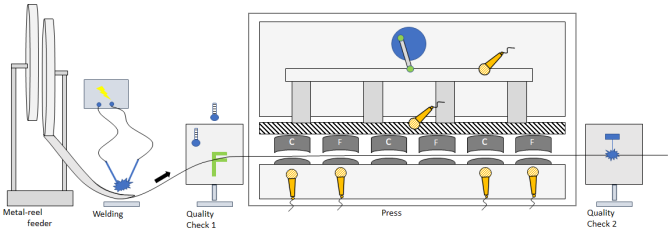

Figure 1: Cold-forming press from [17]

The remainder of this work is structured as follows. Section 2 provides a higher level description of the case study and formulates the problem. The existing unsupervised anomaly detection technique is presented in Section 3. In Section 4, we present the additional thresholding techniques. Also, we provide a qualitative discussion about the integration of such techniques in a real PdM scenario. Finally, we evaluate the presented thresholding techniques in real-world PdM case scenario and discuss the related work in Sections 5 and 6, respectively. We conclude in Section 7.

2. CASE STUDY DESCRIPTION

The current use case aims at the production line of a Philips factory and more specifically, the cold-forming press that is a part of the production line. A high level representation of the specific cold-forming press is shown in Figure 1. A metal strip is fed as input to the press. The final form of the strip is the result of various parts of the press (aka modules or dies) that cut, bend, and flatten the input strip and are placed in the correct order to produce the desired output shape. These modules are considered complex and expensive pieces of equipment, and are subject to failures induced by age and other reasons. Essentially, the modules are used in such a pipelined manner that the output of one module forms the input of the next module. Firstly, the metal strip goes through an input reel and the first quality check is performed. Then, the metal strip arrives in the press and is processed by each of the six modules in sequential order where every one of the modules changes the shape of the strip. More specifically, the press performs a circular motion in order for the modules to strike with force to the metal strip giving the desired shape. When the strip passes through all of the modules, it exits the press and the second quality check is performed.

We cannot monitor the condition or the status of each module without setting the machine off operation. During condition checks, the whole press is removed from the production line in order to plan preventive maintenance inspections. The cost of such removals and especially the cost of faulty parts of the machinery can be large for the company. This calls for a non-intrusive predictive maintenance solution in the cold-forming press.

There are multiple monitoring data regarding this cold-forming press and the metal strip, which are continuously collected. With regards to the metal strip, the part number, the thickness, and the temperature entering the press are monitored. In addition, at the end product of the press, the thickness, the shape, and other proprietary quality measurements are collected during the quality check. To track the condition of the cold forming process, acoustic emission sensors are

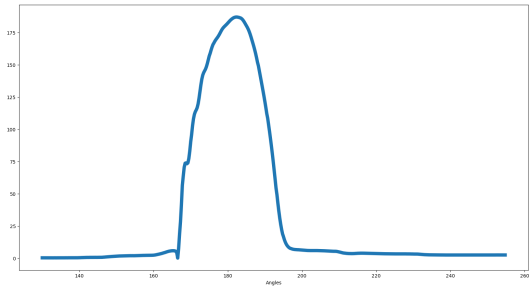

Figure 2: Typical shape of a punch

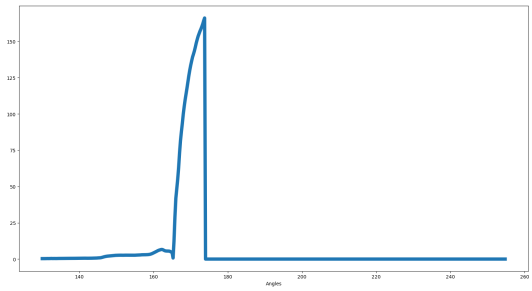

Figure 3: Signal shape when operation of press was manually stopped halfway.

brought into service; altogether, the different acoustic sensors include 6 channels, and at each press operation step, the contact of the module with the metal strip is captured using the sensor. The sensor transforms the piezoelectric signal emanated by the contact of each module with the material into acoustic waves. We are interested in the part of the circular motion of the press, where modules interact with the metal strip. Several hundreds of acoustic emission measurements are collected in that part of the motion, which are fed to our PdM solution.

2.1 Data Description and Notation

There is a distinct data stream for each of the six channels of acoustic emission signals described above. Each channel corresponds to a different part (die) of the press and produces 500 values per punch, where each value corresponds to the acoustic emission at a specific angle when the press performs a single punch. An example is shown in Figure 2. For the purpose of this work, to apply our techniques, we collect these signals in coherent sequences referring to the same die setting, termed as episodes. Each episode concerns one of the channels and starts after a maintenance action is performed in the corresponding die or after a human interaction with the die takes place. The episode ends when a failure in a die occurs (which is detected from the downstream quality checks of produced product most of the times) or the die is reconfigured for other purposes and the specific production process stops.

More formally, an episode $Ep_i = \{x_0, x_1, ..., x_f\}$ is a multivariate time series, where each $x_t = \{x_t^0, x_t^1, ..., x_t^{499}\}$ is a 500-dimensional vector, which corresponds to one hit at

timestamp t, $0 \leq t \leq f$. Timestamp f refer to the final hit of the die in the specific episode.

For each episode Ep_i, we perform domain expertise-driven data cleaning, removing all x_t entries that correspond to a sensor failure, problematic events or manual operation. For example, an operation manually stopped in halfway results in zero x_t signals (see Figure 3). Such errors can be easily detected and excluded due to their obvious difference from the signals like the one in Figure 2. Also, there are some additional cases of single anomaly punches (problematic events), which are monitored and are detected via an automated process (e.g. when double material is passed to the press); we also exclude such values.

The technical solutions described hereby are enforced through an end-to-end modular and extensible PdM system, called *Sibyl*, the software engineering aspects of which are described in [18].

3. A PROFILE-BASED SOLUTION TO ANALYSE PUNCH SIGNALS

Given a stream of data-points, Sibyl's *Profile-Based algorithm (PB)* aims first to construct a normal representation of such data in an unsupervised manner and, then, to detect deviations as new data arrive based on the normal representation constructed. The algorithm is separated into two main steps: 1) Construction of profile and 2) Anomaly detection.

With regards to the first step, a profile $pf = \{x_t, ..., x_{t+n-1}\}$ is constructed from n continuous time points ensuring that the maximum distance between two of the profile members, denoted as $m = max(dist(x_i, x_j))$, $x_i, x_j \in pf$, does not exceed a threshold; *dist* is a distance function chosen.

The rationale behind the profile construction is to select data from a normal operation period of the press while not relying on domain experts' feedback. Since the normal operation changes from episode to episode, the algorithm should be capable of reliably calculating the profile at runtime. Therefore, to construct the profile online, we use a threshold lm, which limits the m value defined above. The full process can be seen in Alg. 1. After the first n points ($\{x_0, ..., x_{n-1}\}$) of an episode Ep have arrived, *PB* considers them as a potential profile. We calculate the maximum inner distance m and check if it is less than lm. If this is the case, a profile pf has been constructed; otherwise, we wait until the next point arrives and make the same test for the points $\{x_1, ..., x_n\}$. This iterative procedure is followed until a profile is found.

After a profile pf is constructed we move to the second step for anomaly detection. Whenever a new point x_t arrives, we calculate its distance from *pf*. The distance d_t of a data-point x_t from the profile pf is determined by the closest point of the profile, $d_t = min(dist(x_t, x)), \ \forall x \in pf)$. For the distance *dist*, any distance metric, such as city-block, mahalanobis, jaccard and so on, could have been used. In this work, we employ either Euclidean or Euclidean with Dynamic Time Warping (DTW). Euclidean is a robust distance metric that can be computed fast. On the other hand, DTW is a computationally intensive calculation but can handle shifts in signals.

The d_t distance value is used as the anomaly score of the point x_t. To produce alarms, we use the threshold value th. This threshold is calculated based on the m value of the profile weighted by the *factor* parameter (see Alg. 1). We refer to this thresholding technique as *baseline*.

Algorithm 1 The Profile-Based PdM solution

Input: $Ep, lm, n, factor$
- $pf \leftarrow \{x_0, ..., x_{n-1}\}$
- $m \leftarrow max(dist(x_i, x_j), \ \forall (x_i, x_j) \in pf \times pf)$
- $cnt \leftarrow 0$
- **while** $m > lm$ **do** ▷ first step
  - $cnt \leftarrow cnt + 1$
  - $pf = \{x_{cnt}, ..., x_{cnt+n-1}\}$
  - $m \leftarrow max(dist(x_i, x_j), \ \forall (x_i, x_j) \in pf \times pf)$
- $th \leftarrow factor \times m$ ▷ threshold calculation
- $t_{curr} \leftarrow cnt + n$
- **while** $t_{curr} \leq f$ **do** ▷ second step
  - $x_t \leftarrow Ep[t_{curr}]$ ▷ get the next arrived data-point
  - $d_t \leftarrow min(dist(x_t, x), \ \forall x \in pf)$
  - **if** $d_t > th$ **then**
    - $ProduceAnAlarm(t_{curr})$
  - $t_{curr} \leftarrow t_{curr} + 1$

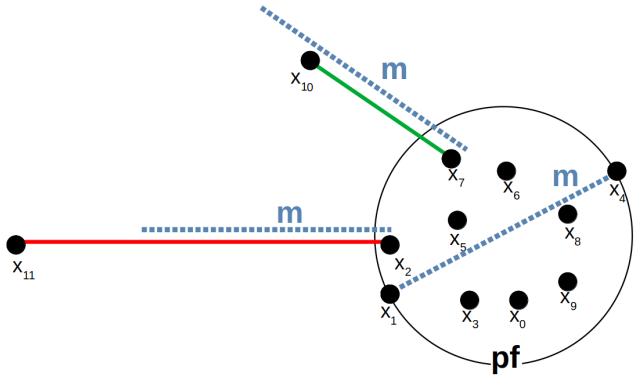

Figure 4: An example of the *PB* technique.

In Figure 4, an example of pf is shown. Here, pf is formed by the points inside the circle. We use a *factor* equal to 1 and the m distance is shown with dotted lines. Suppose that the next point after the construction of the profile is x_{10}. The closest point of x_{10} is x_7, and so, the distance of x_{10} from x_7 is the minimum distance of x_{10} from the whole pf. This distance remains smaller than $factor \times m$, hence x_{10} is tagged as a normal observation. The next point is x_{11}. The closest point to x_{11} from pf is x_2. This point is tagged as anomaly because the minimum distance from x_{11} to pf is larger than $factor \times m$.

4. THRESHOLD SETTING SOLUTIONS

In the current work, we focus on tuning the anomaly score threshold parameter. In order to make a final decision, e.g., replace a component of the press, schedule a maintenance action, and so on, a threshold parameter is used. If the produced anomaly score is higher than the threshold, then we assume that there is a deviation in the monitoring component, which triggers further actions; therefore, the threshold should be set in an as informed manner as possible.

More specifically, we evaluate three techniques in addition to the baseline one. These techniques are applicable to any anomaly detection algorithm, where an anomaly score is pro-

Algorithm 2 Windowed 2T with Standard Deviation (M2T) [23]

Input: Multiplier factor: $factor$, Anomaly Scores: $As = \{d_{ts}, ..., d_{tl}\}$, Window: w
$Aw \leftarrow \{d_{tl-w}, ..., d_{tl}\}$
$th1 \leftarrow \mu(Aw) + factor \times \sigma(Aw)$
$Aw \leftarrow \{d_t \in Aw | d_t < th1\}$
$th \leftarrow \mu(Aw) + factor \times \sigma(Aw)$
return th

duced. Furthermore, we explore a simple extension to two of them. Each of these techniques comes with its peculiarities, which we discuss in turn, and their suitability is largely application-dependent. In summary, the techniques described in this section aim to either replace the threshold calculation in Alg. 1 or even to recalculate it upon the arrival of each new point. We define the time series of anomaly scores $As = \{d_{ts}, ..., d_{tl}\}$, where ts is the timestamp of the data-point that arrived right after the construction of profile, and tl is the timestamp of the most recent data-point arrived until now.

4.1 Moving 2T

The *2T* technique is introduced in [23]. A common technique would use the mean and the standard deviation of all anomalies scores in As, where a data-point is tagged as an anomaly if its anomaly score is *factor* times the standard deviations away from the mean. The rationale of 2T is that the set (or the time series in our context) of anomaly scores will eventually contain some outliers. So, these outliers will increasingly contribute to the calculation of the threshold. Therefore, by applying twice (or more) the threshold computation technique as shown in Alg. 2, we can eliminate the effect of the outliers that are removed after the first pass.

In our context, we transform 2T into a streaming technique, called Moving 2T ($M2T$). To do so, we use a time window over As. The window contains values from the last w points of As. Then, we apply the technique described above on the w last values of As. Furthermore, we test a simple extension called *M2T-X*, where, after the final calculation of the threshold, if a data point is characterized as an anomaly, we exclude it from the As in future threshold calculations.

4.2 Dynamic threshold

In [9], a novel Long-Short-Term-Memory deep learning model to detect anomalies in telemetry data is presented. The deep learning model outputs an error (anomaly score) forming a time series. In order to set threshold on such an error time series, the authors present a dynamic threshold setting technique, which is referred to as *Dyn*. Using a time window selecting the last w anomaly scores, they calculate a threshold based on an objective function. That objective function aims to drop the mean and the standard deviation and, at the same time, penalizes a large amount of reported anomalies. To further mitigate the presence of false positives, they use a limiting *dp* parameter. After the calculation of th, *Dyn* groups anomalies into sequences, i.e., continuous values that exceed the threshold th and extracts the max value of each sequence. Sequences are then sorted by this value. Then, the technique calculates the percentage difference between adjacent maximums in the sorted array. If the percentage difference at some point exceeds the *dp* limit, then all sequences until that point are considered anomalies. A more detailed description can be found in [9].

In our context, we perform this calculation for each new d_t, and we check if it belongs to an anomaly sequence. Similar to the *M2T-X*, we test the same extension *Dyn-X* here as well, where, after the final calculation of the threshold, if a data point is characterized as an anomaly, we exclude it from the As in future threshold calculations.

Algorithm 3 Profile-based with self-tuning

Input: Ep, lm, n, $factor$
$pf \leftarrow \{x_0, ..., x_{2n-1}\}$ ▷ *changed
$m \leftarrow max(dist(x_i, x_j),\ \forall (x_i, x_j) \in pf \times pf)$
$cnt \leftarrow 0$
while $m > lm$ **do**
  $cnt \leftarrow cnt + 1$
  $pf = \{x_{cnt}, ..., x_{cnt+2n-1}\}$ ▷ *changed
  $m \leftarrow max(dist(x_i, x_j),\ \forall (x_i, x_j) \in pf \times pf)$
$pf = \{x_{cnt}, ..., x_{cnt+n-1}\}$ ▷ *changed
$pf_{th} \leftarrow \{x_{cnt+n}, ..., x_{cnt+2n-1}\}$ ▷ *added
$distances_{th} \leftarrow []$ ▷ *added
for $x_t \in pf_{th}$ **do** ▷ *added
  $d_t \leftarrow min(dist(x_t, x),\ \forall x \in pf)$ ▷ *added
  $distances_{th}.append(d_t)$ ▷ *added
$th \leftarrow \mu(distances_{th}) + factor \times \sigma(distances_{th})$ ▷ *changed
$t_{curr} \leftarrow cnt + n$
while $t_{curr} \leq f$ **do** ▷ second step
  $x_t \leftarrow Ep[t_{curr}]$ ▷ get the next arrived data-point
  $d_t \leftarrow min(dist(x_t, x),\ \forall x \in pf)$
  if $d_t > th$ **then**
    $ProduceAnAlarm(t_{curr})$
  $t_{curr} \leftarrow t_{curr} + 1$

4.3 Self-tuning

Self-tuning is a novel anomaly score thresholding solution that we propose. In *PB*, we heuristically try to enforce that the profile comprises measurements depicting the healthy equipment's operation. Building upon this, instead of searching for profile of length n, we search for a profile of length of double length, i.e., $2n$. We use the first part (i.e., the initial n points) for profile construction and the second half to calculate a threshold. More specifically, we pass the second half to *PB* and get as output a sequence of anomaly scores. Then, we compute the final threshold as the mean of this sequence plus the standard deviation multiplied by the *factor* parameter. The rationale is that the threshold is better configured based on the expected value of anomaly scores in normal data not used for profile construction rather than based on the measurements directly used for the profile construction as in the baseline. The modified *PB* algorithm is shown in Alg. 3. The modifications in the original algorithm are explicitly annotated in the comments.

4.4 Discussion

Rather than regarding the techniques discussed above as direct competitors, it is important to understand the anomalies each technique targets instead of treating them as black boxes. In a PdM scenario, the decision to characterize a point as an anomaly or not could eventually change the production schedule of a factory. So, it is important to under-

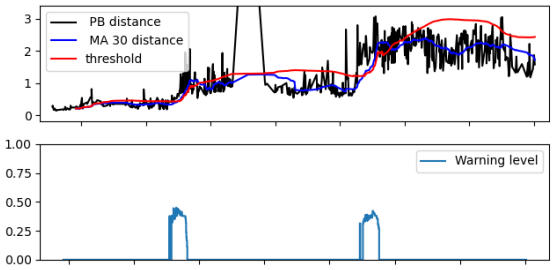

Figure 5: The upper plot shows the produced error from *PB* along its moving average. The bottom plot shows the warning level when using *Dyn*.

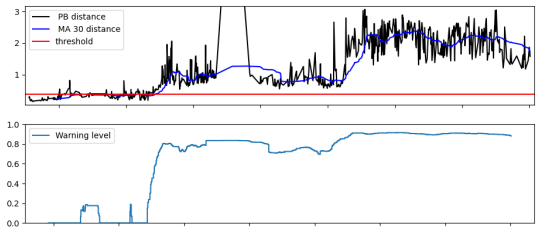

Figure 6: The upper plot shows the produced error from *PB* along with its moving average. The bottom plot shows the warning level produced by *self-tuning*.

stand *why* a point is tagged as an anomaly and what is the impact of such an anomaly.

All the threshold techniques that we test avoid using a predefined value in the error domain to set a threshold, which would be a too naive solution that would be ineffective in a setting, where operational characteristics change frequently. We can separate them into two groups: a) dynamic ones (*Dyn, M2T*), where the threshold value is calculated continuously during a single episode, and b) constant ones (*baseline, self-tuning*), where a threshold value is calculated at the beginning of the episode.

Each of the two categories has its own characteristics. For example, as shown in Figures 5 and 6, the dynamic ones can detect the rising of the error value produced from *PB*, but after that, if the error value remains at the same level, eventually (depending on the w number of historical values), dynamic techniques adapt to the new potentially erroneous behavior. On the contrary, in the constant techniques, if the error value rises and exceeds the threshold, we keep raising alarms as long as the error remains larger than the threshold. In addition, dynamic techniques can detect the rise in the error independently of the absolute value of the error. This may be valuable information in some PdM applications.

Furthermore, it may be suitable to use different threshold techniques depending on the output characteristics and the task. For example, if we are interested in spikes in anomaly scores (i.e., few consecutive data-points with an obvious difference), we should use dynamic techniques without the smoothing step. In our case, we are interested in the general wear of die components in the press, not only spikes.

An additional note is that we could have used the median instead of the mean value in the techniques. However, in our episodes, such a change had no impact and the behavior remained the same.

Finally, the efficiency of a threshold technique strongly depends on the output of *PB*. If *PB* cannot model the deviation in its output, for any possible reason, whatever threshold technique we may use, it cannot perform well.

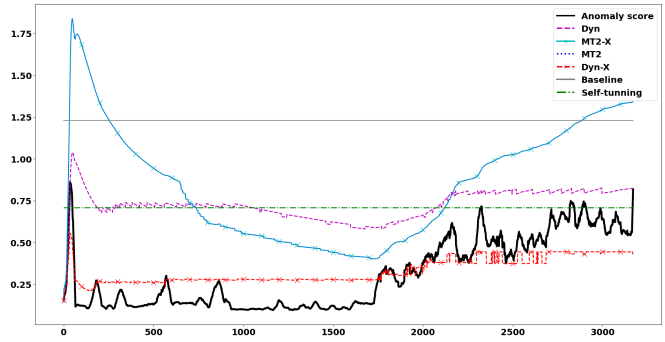

Figure 7: Threshold techniques.

4.5 An example

To provide a more detailed view of the behavior of the thresholding techniques and their characteristics, we show how they perform over the anomaly score produced in one of the test episodes in Figure 7. As will be explained later, the parameters for each thresholding technique is selected by utilizing the two normal episodes so that no false alarms are produced. The produced anomaly score of that episode includes a high spike at the beginning, and after that, it remains relatively low for some period before increasing again and remain at the increased level until the end of episode. In reality, a fault occurred only at the end of the episode.

The *baseline* technique yields a high threshold value. This is one of the main problems encountered in this technique and one of the main motivating reasons for this work. The high threshold value is due to the high m distance (i.e., the maximum inner distance of the profile pf). It is difficult to select an optimal lm parameter for the profile-based algorithm, thus the m distance of the profile is unnecessarily high in some episodes. High m values denote a sparse profile, which is more likely to yield lower anomaly scores in general. Overall, high m values are prone to situations where no alarms are produced due to misconfigurations. In the example shown, the *baseline* technique did not manage to raise an alarm. On the other hand, *self-tunning* computes a constant threshold for the entire episode but in a different manner and manages to produces alarms at the end of episode, where anomaly score takes its maximum values. However, it did not avoid raising false alarms at the beginning.

M2T and *M2T-X* exhibit identical behavior. This holds in every episode where no alarms are produced by *M2T*, since *MT2-X* differs only in that it removes previously annotated outliers' information to calculate the threshold. *M2T-X* is heavily affected by the spike in the beginning of the episode, which results in a high variance in the anomaly scores time series and an early increase in the threshold. In the example, it failed to detect the failure.

Dyn is more aggressive than the other methods. Initially,

the threshold is calculated in such a manner that an anomaly is detected in the first spike, but after that, the threshold increases because its calculation is affected by that spike. On the other hand, *Dyn-X* does not suffer from this limitation because it ignores the previously annotated anomalies. Although *Dyn-X* produces a false alarm in the beginning of the episode, it derives a threshold, which better catches the rising in the anomalies scores at the end of the episode.

5. EVALUATION

We assess the efficiency of the techniques using the recall, precision, and F1 metrics. To this end, we need to define what true positive (TP), false positive (FP) and false negative (FN) mean in our setting. We evaluate the techniques using two different definitions for TP, FP, and FN. In both cases, we use three different predictive horizons (PH). True negatives (TN) correspond always to the case that no alarms are raised in an episode that ends with no failure.

Setting 1: In the first setting, for each episode, we count one TP if an alarm is raised within the PH period. We count one FP if at least one alarm is raised before the PH period. Finally, we count one FN, if no alarm was raised within the PH period (FN=1-TP). The rationale here is to have an equal number of TP and FP if alarms are raised both before and during the PH. Essentially, no matter how many alarms are raised during the PH, the behavior counts as a TP if there is at least one alarm raised during that period. In addition, no matter how many alarms are raised prematurely, i.e., before the PH, the behavior counts as a single FP at most.

Setting 2: In the second setting, a stricter definition is used. Here, each episode is characterized by only one FP or TP or FN or TN overall. We characterize a whole episode as FP if an alarm is raised before the PH, because, in the real world, this would lead to an investigation or unnecessary maintenance of the component and the production would stop. An episode counts as a TP if alarms are raised exclusively within PH, which results in the full usage of the component. Finally, if no alarms are made although the episode ends with a failure, we characterize the episode as FN. This setting penalizes techniques that are prone to produce FPs regardless of whether or not they detect anomalies in the PH.

Note that in both settings, it is adequate to raise an alarm a single time to count as a FP; therefore, the characteristic of *self-tuning* to persistently raise alarms is not translated to a strong advantage. Regarding the setting of PH itself, we use meaningful time periods for the maintenance operators, although this results in relatively short PH periods compared to the duration of episodes, which entails that the likelihood to consider an alarm as a FP is increased. We also use a buffer period, i.e., the end of the PH does not coincide with the end of the episode since maintenance decisions cannot be enacted in an instantaneous manner in reality. Overall, we employ an evaluation setting that does not facilitate high performance but captures realistic aspects encountered in an industrial environment.

Our data comprise 12 preselected episodes, where two of them have been characterized as normal ones, i.e. after some period of operation, a preventive maintenance took place and reported that the corresponding module is in healthy state. The other 10 episodes end with a reported failure.

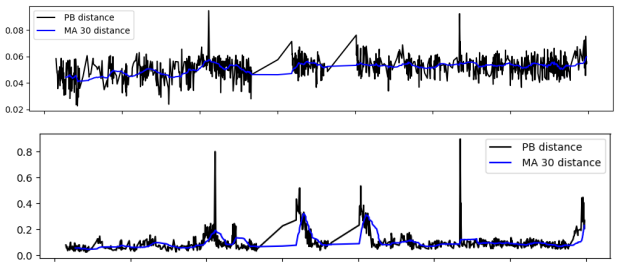

Figure 8: *PB* distances in a normal episode using DTW (top) and the Euclidean distance (bottom).

We use the 2 normal episodes to tune the parameters of the presented techniques and the remaining 10 episodes for testing, where no TN are applicable. The episodes used are more than the ones mentioned in our work in [7]; so, the results presented are slightly different but the main message remains the same.

Configuration. The profile length n in *PB* is set to 30, which, based on domain experts, corresponds to an acceptable time period and is order(s) of magnitude shorter than an episode length. For lm, we employ a fixed empirical value, since it is out of the scope of this work to optimize all parameters. Moreover, a smoothing technique is used over the produced distances, which is implemented as a moving average with a window of length 30. This is known to be more representative in our case study.

Furthermore, as described above, we use the two healthy episodes to configure the parameters of threshold techniques to not produce any alarm during these episodes. For example, when we test *self-tuning* and the smallest factor that does not produce any alarm in the normal episodes is f_{min}, in the evaluation process, we will test factors that are equal to or greater than f_{min}.

We investigate two distance functions, Euclidean and DTW when computing the distances between signals. As shown in Figure 8, the Euclidean metric results in more variance in the anomaly score. This is due to shifted signals in the angle domain. Such a shift is due to speed changes in the press, or due to the cold start after an idle period and does not reflect a fault. Larger variances in a normal episode result in higher factor parameters in the threshold techniques in order not to characterize these data as anomalies. The different behavior of the distance functions heavily impacts on the techniques' performance, as discussed below.

Results - Setting 1: In the first setting, we show the results using euclidean and DTW distance metrics in Figure 9. The current baseline implementation, where a threshold is defined according to the maximum inner distance of the profile, yields a 0.27 F1 score using both DTW and Euclidean distance, respectively. *Self-tuning* exhibits the most robust behavior with similar evaluation metrics for both distance metrics. Under the DTW distance, it succeeds equal performance to the best one. In the case of the Euclidean distance, the *self-tuning* technique manages to detect one failure less than the best performing technique (i.e. 0.1 drop in recall value), which results in slightly decreased F1. However, in general, it outperforms the baseline in all combinations of distance functions and PH values.

The best results for the DTW distance are achieved by *Dyn-*

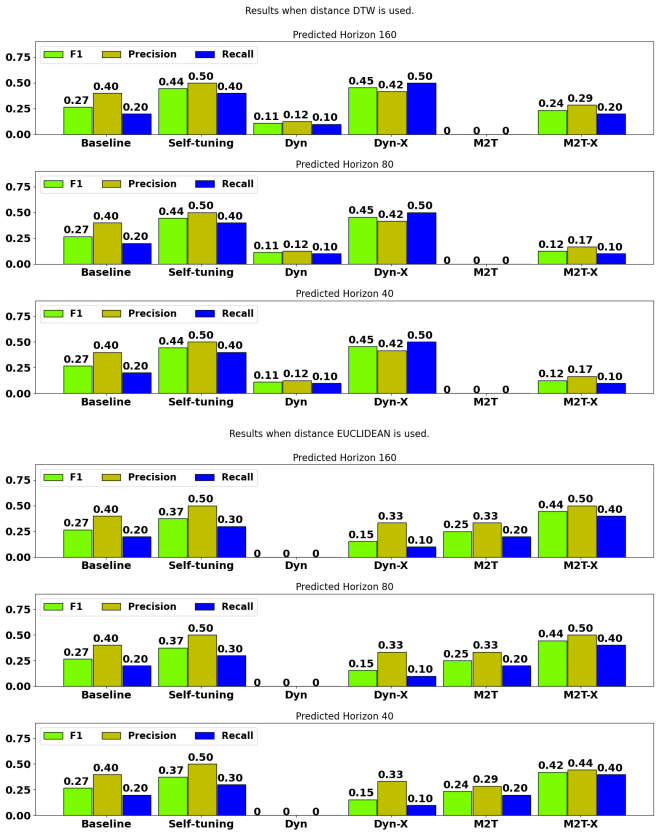

Figure 9: Setting 1 results for DTW (top) and Euclidean distance (bottom) and different PH values.

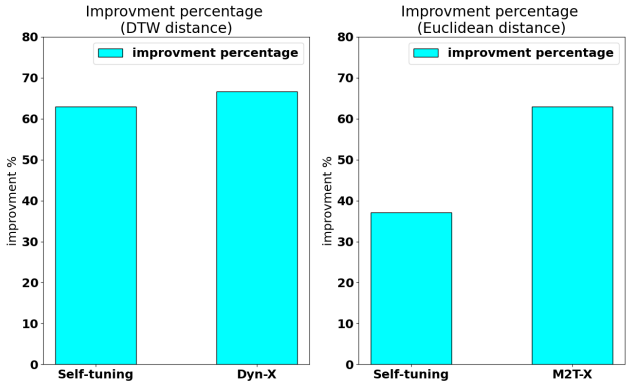

Figure 10: Percentage F1 score improvement over the baseline technique.

X (using $dp = 0.11$). The difference between *Dyn* and *Dyn-X* is notable; after *Dyn* detects some initial deviations manifested as an increase in the anomaly score, it adapts to this increased anomaly score, which yields suboptimal results in our setting. Additionally, both *M2T* and *M2T-X* could not operate efficiently when DTW is used. Although both *Dyn* and *M2T* are dynamic approaches, they largely differ in the calculation of the threshold. *Dyn* is more greedy and calculates the threshold relatively close to the anomaly score

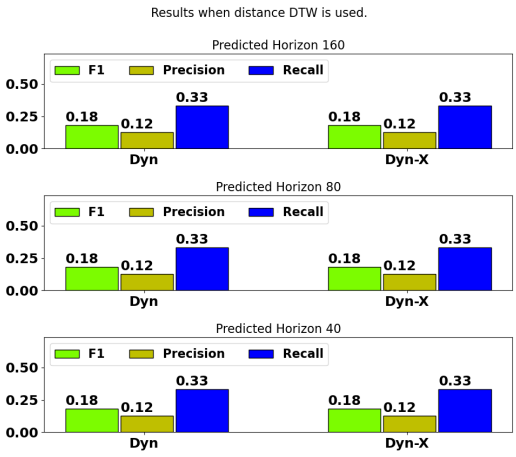

Figure 11: Setting 2 results for DTW distance and different PH values.

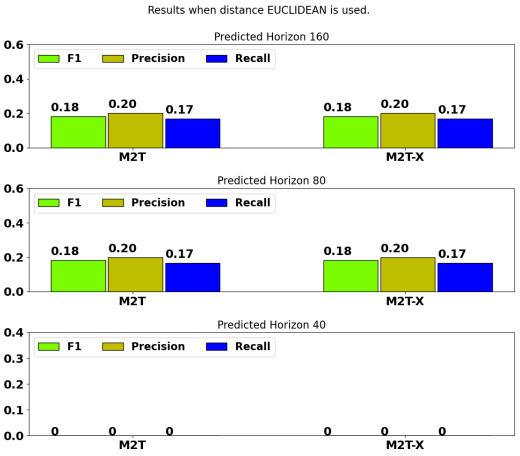

Figure 12: Setting 2 results for Euclidean distance and different PH values.

using the *dp* value in a pruning post-step. On the other hand, *M2T* uses a factor, which remains constant, to calculate the threshold each time. Because the DTW results in a smoother anomaly score, it is more difficult to produce values that exceed the threshold of *M2T* or *M2T-X*. Using the Euclidean distance, we observe the opposite behavior. *Dyn* could not perform well because of the fact that the variance is higher and it needs a much higher *dp* (0.25) to suppress alarms in normal episodes. Such a high *dp* value results in no alarms even in the faulty episodes. Using the extended version of *M2T*, we get the best results. Finally, Figure 10 shows the percentage improvements in F1 over the baseline.

Results - Setting 2: Here, we are stricter regarding the assessment of the performance of the techniques and this is reflected upon the results; see Figures 11 and 12, where only non-zero values are depicted. Only the dynamic techniques yield non-zero metrics. The previous discussion on *Dyn* and *M2T* holds here as well. As shown in the results, for this setting, both extensions succeed in equal F1 scores. This is an expected behavior since, for Setting 2, we characterize the whole episode as a false positive in case of an alarm outside of the predictive horizon. So even if anomalies are

removed after their appearance, and then alarms are made inside PH, that does not modify the characterization of the episode. While in the previous Setting 1, even when the dynamic techniques raised a false positive alarm, by ignoring previously annotated anomalies, their extensions could produce alarms inside PH too.

6. RELATED WORK

The benefits of PdM techniques when applied in industrial settings have resulted in an increase of attention from both the research community and industries themselves. In [4], a taxonomy of data-driven prognostic techniques, trends, directions and a comparative overview is provided with respect to industrial settings. Moreover, open technical challenges have been identified regarding the consequences of each application in an industrial setup. In addition to the aforementioned remarks, an important conclusion from [10] is that each application comes with its own unique characteristics; this also applies to our case, where we reason about the behavior of generically applicable thresholding techniques in our specific case study.

PdM techniques can be distinguished based on the form of input data they manipulate in two main categories, namely event-based techniques (processing discrete data) and sensor-based ones (processing continuous data commonly stemming from sensors). Regarding the first category, there are both supervised and unsupervised approaches. The techniques tend to use predefined event codes or artificial ones, extracted from raw data, in order to detect a deviation in the data and predict anomalies. Manco et. al predict failures in train doors using events from pre-installed software in [15]. In [5], the authors try to predict the number of stoppages of a packing machine using ARIMA and Prophet algorithms. Extreme learning is used in [6] for the prediction of failure in trains in the near future. The ARMA algorithm is used in [1] as a feature along with other thirteen statistical features and PCA is performed before the event prediction process using KNN, RF, and SVM. In [20], a novel framework is proposed for vehicle failure prediction. Another supervised methodology is presented in [22] using a graph with relations among known anomaly sequences and sub-sequences of the data. Finally, the authors in [17] use both supervised and unsupervised methods for event-based predictive maintenance in the same industry case study, where a novel algorithm with Matrix Profile [24] was used to transform data into time series of events. Here, we emphasize on the impact of thresholding rather than the presentation of a new technique.

In settings where sensor values are available, a common approach is to predict the Remaining Useful Lifetime (RUL) of a component [12; 8]. Aiming to mitigate the need for fine-tuning in a supervised setting, transfer learning [13] and AutoML [21] are used. In our work, an unsupervised technique is chosen to model the deviation in data. Unsupervised models are preferable when we are dealing with dynamic environments and no labels are available. E.g., the authors in [3] employ clustering algorithms.

Our work mostly relates to PdM proposals that focus on setting the anomaly score judiciously. The authors in [9] use a Long Short-Term Memory (LSTM) deep neural network to detect anomalies in telemetry data. To set threshold on the error produced from the model, they propose a novel dynamic thresholding technique *Dyn*, which chooses a threshold that optimises an objective function. We explicitly consider this approach as discussed in the main technical sections of this work. For the same problem, this technique was used in [16], where the difference is that the model used to produce the error was a transformer instead of LSTM. Furthermore, in [11], the authors extend the dynamic threshold to fit to their case. More specifically, they perform an additional pre-processing step in the error signal before they apply the *Dyn* algorithm. We also extend *Dyn*, as described previously yielding the *Dyn-X* variant. In addition, the authors in [19] employ a dynamic thresholding technique to produce outliers using the concept of sliding window. The *2T* solution is presented in [23]. David et al. in [2] use a dynamic thresholding technique over each feature to perform DDoS flood attack detection; in our work, there is a single feature considered due to the specificity of our case study. Finally, in [14], the threshold configuration is optimized based on historic data. In our case, data from previous episodes provide little information on how to tune the current episode.

7. CONCLUSIONS

In a real world dynamic PdM environment without any labelled training data, it is more appropriate to employ unsupervised anomaly detection using an advanced thresholding technique rather than just a constant threshold value over the produced anomaly scores. In this work, we investigate four such thresholding techniques in an industrial setting that involves a cold-forming press. Our results show that the appropriate choice of the thresholding technique could lead to significantly higher F1 scores compared to the currently running choices. Dynamic techniques along with our proposed extensions seem to be the dominant alternatives. However, the choice of a thresholding technique strongly depends on the characteristics of the problem. For example, *self-tuning*, which is not dynamic, is the most appropriate in an environment where we are interested in persisting alarms. Also, in our work, we discuss the impact of the distance function employed. In the future, we aim to extend our work investigating (i) more variants and techniques, such as using the median instead of the mean values and exploring the potential of deep learning solutions, and (ii) post-processing maintenance scheduling decision support.

8. ACKNOWLEDGEMENTS

The research of A. Giannoulidis and A.Gounaris was carried out as part of the project "Smart Maintenance Platform for production equipment in the Factory of the Future leveraging on advanced prediction and fault detection models" (Project code: KMP6-0077768) under the framework of the Action "Investment Plans of Innovation" of the Operational Program "Central Macedonia 2014-2020", that is co-funded by the European Regional Development Fund and Greece.

9. REFERENCES

[1] M. Baptista, S. Sankararaman, I. P. de Medeiros, C. Nascimento, H. Prendinger, and E. M. Henriques. Forecasting fault events for predictive maintenance using data-driven techniques and arma modeling. *Computers & Industrial Engineering*, 115:41–53, 2018.

[2] J. David and C. Thomas. Efficient ddos flood attack detection using dynamic thresholding on flow-based network traffic. *Computers & Security*, 82:284–295, 2019.

[3] A. Diez, N. L. D. Khoa, M. Makki Alamdari, Y. Wang, F. Chen, and P. Runcie. A clustering approach for structural health monitoring on bridges. *Journal of Civil Structural Health Monitoring*, 6(3):429–445, Jul 2016.

[4] A. Diez-Olivan, J. Del Ser, D. Galar, and B. Sierra. Data fusion and machine learning for industrial prognosis: Trends and perspectives towards industry 4.0. *Information Fusion*, 50:92–111, 2019.

[5] G. Filios, I. Katsidimas, S. Nikoletseas, S. Panagiotou, and T. P. Raptis. An agnostic data-driven approach to predict stoppages of industrial packing machine in near future. In *2020 16th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 236–243, 2020.

[6] O. Fink, E. Zio, and U. Weidmann. Extreme learning machines for predicting operation disruption events in railway systems. In *ESREL 2013*, pages 1–8, Amsterdam, Netherlands, Sept. 2013.

[7] A. Giannoulidis, N. Nikolaidis, A. Naskos, A. Gounaris, and D. Caljouw. Investigating thresholding techniques in a real predictive maintenance scenario. In *Machine Learning and Knowledge Discovery in Databases - International Workshops of ECML PKDD 2022 (to appear)*. Springer, 2022.

[8] L. Guo, N. Li, F. Jia, Y. Lei, and J. Lin. A recurrent neural network based health indicator for remaining useful life prediction of bearings. *Neurocomputing*, 240:98–109, 2017.

[9] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 387–395, New York, NY, USA, 2018. Association for Computing Machinery.

[10] P. Korvesis, S. Besseau, and M. Vazirgiannis. Predictive maintenance in aviation: Failure prediction from postflight reports. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 1414–1422, 2018.

[11] T. Li, M. Comer, E. Delp, S. R. Desai, J. L. Mathieson, R. H. Foster, and M. W. Chan. A stacked predictor and dynamic thresholding algorithm for anomaly detection in spacecraft. In *MILCOM 2019 - 2019 IEEE Military Communications Conference (MILCOM)*, pages 165–170, 2019.

[12] X. Li, Q. Ding, and J.-Q. Sun. Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering & System Safety*, 172:1–11, 2018.

[13] Y. Li, C. Liu, J. Hua, J. Gao, and P. Maropoulos. A novel method for accurately monitoring and predicting tool wear under varying cutting conditions based on meta-learning. *CIRP Annals*, 68(1):487–490, 2019.

[14] Y. Liang, S. Wang, W. Li, and X. Lu. Data-driven anomaly diagnosis for machining processes. *Engineering*, 5(4):646–652, 2019.

[15] G. Manco, E. Ritacco, P. Rullo, L. Gallucci, W. Astill, D. Kimber, and M. Antonelli. Fault detection and explanation through big data analysis on sensor streams. *Expert Systems with Applications*, 87:141–156, 2017.

[16] H. Meng, Y. Zhang, Y. Li, and H. Zhao. Spacecraft anomaly detection via transformer reconstruction error. In Z. Jing, editor, *Proceedings of the International Conference on Aerospace System Science and Engineering 2019*, pages 351–362, Singapore, 2020. Springer Singapore.

[17] A. Naskos, G. Kougka, T. Toliopoulos, A. Gounaris, C. Vamvalis, and D. Caljouw. Event-based predictive maintenance on top of sensor data in a real industry 4.0 case study. In P. Cellier and K. Driessens, editors, *Machine Learning and Knowledge Discovery in Databases - International Workshops of ECML PKDD 2019, Würzburg, Germany, September 16-20, 2019, Proceedings, Part II*, volume 1168 of *Communications in Computer and Information Science*, pages 345–356. Springer, 2019.

[18] A. Naskos, N. Nikolaidis, V. Naskos, A. Gounaris, D. Caljouw, and C. Vamvalis. A micro-service-based machinery monitoring solution towards realizing the industry 4.0 vision in a real environment. *Procedia Computer Science*, 184:565–572, 2021. The 12th International Conference on Ambient Systems, Networks and Technologies (ANT) / The 4th International Conference on Emerging Data and Industry 4.0 (EDI40) / Affiliated Workshops.

[19] I. G. A. Poornima and B. Paramasivan. Anomaly detection in wireless sensor network using machine learning algorithm. *Computer Communications*, 151:331–337, 2020.

[20] U. Shafi, A. Safi, A. R. Shahid, S. Ziauddin, and M. Q. Saleem. Vehicle remote health monitoring and prognostic maintenance system. *Journal of Advanced Transportation*, 2018:8061514, Jan 2018.

[21] T. Tornede, A. Tornede, M. Wever, F. Mohr, and E. Hüllermeier. Automl for predictive maintenance: One tool to rul them all. In J. Gama, S. Pashami, A. Bifet, M. Sayed-Mouchawe, H. Fröning, F. Pernkopf, G. Schiele, and M. Blott, editors, *IoT Streams for Data-Driven Predictive Maintenance and IoT, Edge, and Mobile for Embedded Machine Learning*, pages 106–118. Springer International Publishing, Cham, 2020.

[22] J. Wang, C. Liu, M. Zhu, P. Guo, and Y. Hu. Sensor data based system-level anomaly prediction for smart manufacturing. In *2018 IEEE International Congress on Big Data (BigData Congress)*, pages 158–165, 2018.

[23] J. Yang, S. Rahardja, and P. Fränti. Outlier detection: How to threshold outlier scores? In *Proceedings of the International Conference on Artificial Intelligence, Information Processing and Cloud Computing*, AIIPCC '19, New York, NY, USA, 2019. Association for Computing Machinery.

[24] C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh. Matrix profile i: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 1317–1322, 2016.