A Machine Learning-based Workflow for Automatic Detection of Anomalies in Machine Tools

Marwin Züfle[a,*], Felix Moog[b], Veronika Lesch[a], Christian Krupitzer[c], Samuel Kounev[a]

[a] *University of Würzburg, Am Hubland, 97074 Würzburg, Germany*
[b] *Bosch Rexroth AG, Maria-Theresien-Str. 23, 97816 Lohr am Main, Germany*
[c] *University of Hohenheim, Fruwirthstr. 21, 70599 Stuttgart, Germany*

Abstract

Despite the increased sensor-based data collection in Industry 4.0, the practical use of this data is still in its infancy. In contrast, academic literature provides several approaches to detect machine failures but, in most cases, relies on simulations and vast amounts of training data. Since it is often not practical to collect such amounts of data in an industrial context, we propose an approach to detect the current production mode and machine degradation states on a comparably small data set. Our approach integrates domain knowledge about manufacturing systems into a highly generalizable end-to-end workflow ranging from raw data processing, phase segmentation, data resampling, and feature extraction to machine tool anomaly detection. The workflow applies unsupervised clustering techniques to identify the current production mode and supervised classification models for detecting the present degradation. A resampling strategy and classical machine learning models enable the workflow to handle small data sets and distinguish between normal and abnormal machine tool behavior. To the best of our knowledge, there exists no such end-to-end workflow in the literature that uses the entire machine signal as input to identify anomalies for individual tools. Our evaluation with data from a real multi-purpose machine shows that the proposed workflow detects anomalies with an average F1-score of almost 93%.

Keywords: Predictive Maintenance, Industry 4.0, Industrial Internet-of-Things, Clustering, Anomaly Detection

1. Introduction

The fast developments in the Internet-of-Things (IoT) as well as the continuous miniaturization of sensors lead to new applications in areas such as Industry 4.0 and Industrial IoT. In Industry 3.0, the machines are already automated, but the work steps are static. Therefore, fewer sensors are installed, which means that less data is available on the machines. In Industry 4.0, not only are the static work steps automated, but more intelligence

*Corresponding author
Email addresses: marwin.zuefle@uni-wuerzburg.de (Marwin Züfle), felix.moog@boschrexroth.de (Felix Moog), veronika.lesch@uni-wuerzburg.de (Veronika Lesch), christian.krupitzer@uni-hohenheim.de (Christian Krupitzer), samuel.kounev@uni-wuerzburg.de (Samuel Kounev)

is transferred to the machines, so that various and flexible work steps can be automated and adaptations can be made directly by the machines [1]. For this purpose, the machines are equipped with more sensors, which enables extensive monitoring of machines and collection of data. However, the pure monitoring and collection of data is only a first step. The key is the intelligent analysis of the collected data in combination with algorithms from the fields of machine learning and data mining. To contribute to these developments, various companies start to offer specific software for data analytics. Leading manufacturing companies also recognize the need to expand their portfolios to support data acquisition and analysis, for example, Bosch Rexroth [2].

One such application that benefits from intelligent use of data is predictive maintenance. According to the report "Industrial Internet of Things: Unleashing the Potential of Connected Products and Services" of the World Economic Forum [3], predictive maintenance enables 12% savings on scheduled repairs, 30% reduced maintenance costs, and 70% fewer breakdowns. One prominent example showing the significance of proactively identifying production issues is the case of Volkswagen from 2016 where production issues resulted in financial losses of up to 400 million Euros per week. Nowadays, many companies still follow a regular (periodic) maintenance approach. This often leads to waste of personnel and material, as in many cases, maintenance is not necessary and could be postponed. A study by Mobley identified that this waste is responsible for one third of all maintenance-related costs [4]. Besides, sudden machine defects can still occur, despite the regular maintenance, due to unexpected severe degradation.

Although these numbers clearly show the importance of automatic and early failure detection mechanisms, companies often lack an understanding of state-of-the-art machine learning approaches that can be used for predictive maintenance. Furthermore, approaches from other areas, such as time series forecasting, are sometimes applied but without adaptation to the specifics of the data from the industrial sector.

In contrast, predictive maintenance is a hot topic in academics where many researchers contribute with diverse papers. The following paragraphs present an overview of related work and delineate our work from existing approaches in the literature. In a previous work, we have conducted a survey on predictive maintenance based on a systematic literature review [5]. Using the methodical structure of predictive maintenance systems elaborated there, we have identified two main areas that are related to our contribution: (i) predicting the health of machines, and (ii) scheduling maintenance tasks. For both topics, an in-depth data analysis is required to predict health indices and to decide when to schedule maintenance tasks. First, we give a short overview of approaches for predicting machine health and then look at approaches for scheduling maintenance.

One category of methods that can be applied to detect faults of machines are the statistical models. Gebraeel, for example, focused on analyzing the degradation of components (based on vibration data) and predicting the health of machines [6]. He proposed a stochastic degradation modeling framework to model the remaining life of already partially degraded equipment. His model focused on exponentially degrading components. In contrast, Liao et al. focused on statistical pattern recognition for assessing a health index for machines [7]. They determined the health index by clustering the identified patterns and predicting the machine performance based on auto-regressive and moving average models. Cai et al. proposed a so-called similarity matching procedure that uses the kernel two-

sample test to find the most similar instances in the training data set [8]. Using this statistical matching, they provided an estimation of the remaining useful life of the examined machine along with its probability distribution by using Weibull analysis.

More recent approaches typically apply deep learning methods for machine fault detection. In their study, Hoang and Kang gave an overview of such recent techniques, which are specifically tailored to bearings [9]. In addition to their survey, Hoang and Kang also presented a convolutional neural network (CNN) that used vibration data to detect bearing failures [10]. Han et al. developed a transfer learning approach that used a CNN to diagnose related but unseen faulty machine conditions [11]. Another transfer learning approach was introduced by Sobie et al. [12]. They used a simulation tool to create a training data set and applied the learned model to real world data. To classify bearing failures, Sobie et al. compared different machine learning models, a CNN, and a dynamic time warping approach. While the most common artificial intelligence architecture for machine fault detection is the CNN, Yam et al. used recurrent neural networks (RNN) to add the capability of intelligent condition-based fault diagnosis to conventional condition-based monitoring approaches [13]. To this end, they predicted the trend of equipment deterioration. Other approaches for deep learning than those using CNNs or RNNs used for example auto-encoders. Haidong et al., for instance, presented a stacked transfer auto-encoder that uses particle swarm optimization to diagnose faults in various machines [14]. However, all these deep learning approaches require broad and large data sets to train the models. Therefore, Shah et al. compared the performance of deep learning and statistical approaches and found that in IoT-enabled intelligent manufacturing, statistical methods with feature engineering processes provide higher accuracy than the deep learning models [15]. To overcome the problem of massive training data sets, machine learning methods can be used instead of deep neural networks. However, similar to statistical approaches, when using machine learning methods instead of deep neural networks, the first step is to process the raw data and extract features. Knittel et al. proposed such an approach for the diagnosis of milling machines using typical feature extraction steps and a support vector machine [16].

However, all these approaches only directly consider the execution with a faulty machine part and completely disregard the segmentation of the raw signal. In practice, however, the machine usually provides data on the complete production process, i.e., across several work steps and thus several tools and operations. Therefore, we present an end-to-end approach, which segments the data of a multi-purpose CNC machine into the individual work steps, and thus into different machine tools, and then examines individual phases to detect specific anomalies. To the best of our knowledge, there does not yet exist such an end-to-end workflow that takes the entire machine signal as input in order to identify anomalies for individual tools.

Approaches for scheduling maintenance tasks include, for example, the work from You and Meng [17]. They proposed a modularized framework for scheduling predictive maintenance tasks. With their framework, they integrated real-time sensor-based prognostics information with classical preventive maintenance and condition-based maintenance scheduling. Ladj et al. proposed a prognostic health management (PHM) that interprets the results of existing PHM modules in a new way to identify the remaining useful life of a machine [18]. A genetic algorithm is used to find a cost-optimal scheduling of production and predictive maintenance tasks. Yang et al. also used a

genetic algorithm in their approach to schedule maintenance tasks with regards to the complex interaction between production processes and maintenance operations [19]. The approach from Liao and Wang focused on predicting the machine health as well as deterioration and, based on this, created a maintenance schedule [20]. To determine a machine's state, they used principal component analysis (PCA) and statistical pattern recognition (SPR).

Yet, the planning of maintenance tasks is out of the scope of the proposed workflow. In this work, the focus is only on the detection of anomalous machine tools.

All mentioned approaches from both research areas used diverse methods for identifying the machine's health index and determining the remaining useful life. Yet, the applicability of the mentioned approaches strongly depends on the machines under consideration and the available data. Therefore, it is difficult for many companies to find the right approach for their problem and to transfer the methods used in the literature to their individual machine. In this paper, a novel generalizable workflow for automatic anomaly detection for machines is introduced, which is tailored to the specific requirements of manufacturing machines. The proactive identification of such anomalies is a necessary step to predict degradation and to avoid machine downtimes. The workflow targets multi-purpose machines with multiple different tools and various sensors. More specifically, the contributions of this paper are threefold:

1. We propose a clustering-based approach for segmenting the raw data into the different work steps and thus several machine tools (i.e., phases).
2. We present a generic data preprocessing and oversampling workflow including the application of basic machine learning methods for learning the differentiation of normal and degraded behavior to predict the current degradation state of production machines using basic machine learning classifiers.
3. We evaluate the learned model with real world data from a multi-purpose machine to show its applicability for predictive maintenance. The experimental results show that by using only basic machine learning classifiers (i.e., in contrast to deep neural networks), the workflow is able to learn the distinction between the normal and abnormal state with only very few training examples. This is an important criterion for the practical application in Industry 4.0, which deep neural networks typically cannot fulfill as these require large training data sets.

The remainder of this paper is structured as follows: Section 2 presents the background on the data acquisition. In Section 3, we introduce the machine learning-based anomaly detection workflow in detail. Section 4 presents a broad experimental evaluation using real-world data of a CNC milling machine. Section 5 discusses the most important results of the experimental evaluation as well as threats to validity. Finally, Section 6 concludes the paper.

2. Background

One of the most fundamental parts of every data mining application is the data acquisition. In the field of machine tools and automation, it is often challenging to obtain relevant data. Machines typically rely on sensor

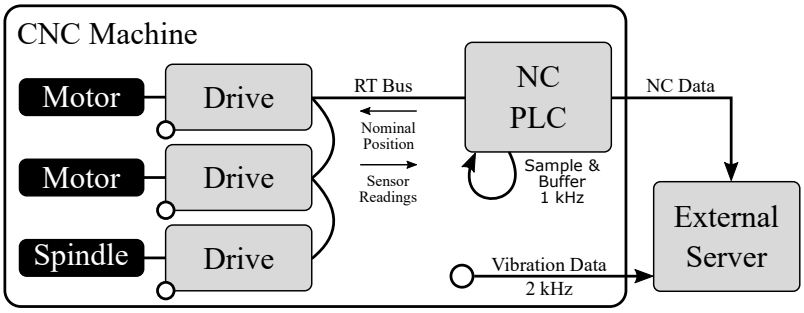


Figure 1: Data flow inside of and out of a CNC machine. The NC/PLC controller provides the nominal position of each drive. The drives are connected via a real-time bus. Drive-internal sensor readings are transferred back to the controller where they are buffered and then transmitted to an external server. This server also samples the vibration sensor mounted to the machine.

data for their operation. However, providing a convenient interface to access sensor data from external devices has not been in the focus of manufacturers. Only in recent years, with the increasing overall interest in data analytics, motivation has increased to provide access to sensor data and dedicate some of the limited computing capacity to utilize the available data sources.

A subclass of manufacturing machines are computerized numerical control (CNC) machines, which provide multiple potential data sources. The general components of such machines with relevant sensors and the resulting data flow is illustrated in Figure 1. The numerical control (NC) is the core of the machine. It controls the movement by following the programmed instructions and is usually accompanied by a programmable logic controller (PLC). Together they form the central point for all data in the machine. The controller is connected to the individual drive controllers via a real-time bus system (RT Bus), such as EtherCAT or SERCOS. Each drive is then connected to one or two motors, usually synchronous servo motors. The drive controllers deliver current to the motors according to the axis position demanded by the controller. They are also responsible for other motion related tasks, like acceleration and deceleration ramps, and can also interpolate the movement. The motors are mechanically connected to the movable parts of the machine. Internal sensors provide the drive with readings required for controlling the movement. This data can usually be transferred to the controller (NC/PLC) via the bus system, but the ability to extract such high-frequency data from the controller is highly vendor-specific, if it exists at all.

For this work, we had access to a five axis CNC milling machine. Besides allowing X, Y, and Z movements, it also has a rotating vice and a pivoting table. A tool magazine with 40 slots allows for automatic tool changes. The machine is equipped with Bosch Rexroth components, such as controller, drives, and motors. In addition, Bosch Rexroth also provides a software tool called efficiency workbench (EWB) for data collection with its NC controllers. This tool allows us to collect drive internal data, from which we selected position and velocity of the axes as well as torque and power draw from the DC intermediate circuit and temperature of the motors. The selection of these signals is based on the fact that they are standard parameters that should exist in most of today's machines. The maximum sampling rate is limited by the interpolator's cycle time and the amount of data to be recorded. In each

cycle, 10 data points can be sampled. If more than 10 parameters have to be sampled, multiple cycles are required and the sampling rate for each data point is reduced accordingly. The recorded data is buffered by the controller and transferred after the measurement is completed. Triggering can be done from within the NC program, which is also recorded synchronously. This means that the program instructions can be directly associated with measurement values of physical signals.

The recording of synchronous NC commands simplifies the phase detection, but it is difficult to apply this to different domains or applications where such data is not accessible, like different vendors or motion controllers without numerical control capabilities. In this paper, we address the problem of phase detection without relying on a recorded NC program (see Section 3.1). The NC program is only used as gold standard data to validate the results.

In addition to the machine-internal sensors, an external vibration sensor is also applied. Since vibration is bound to occur in any rotary machinery, this seems like a natural choice. The used sensor (called CISS) is a multi-sensor specialized for industrial applications. It offers easy connectivity and various integrated sensors at a relatively low cost. It features 3-axis inertial sensors like an accelerometer, a gyroscope, and a magnetometer, as well as several environmental sensors (i.e., temperature, humidity, atmospheric pressure, light, and a microphone). The sensor provides integrated analog-to-digital conversion and can be connected either via Bluetooth or USB. The maximum sampling rate is 1 kHz. There is a special mode available where only the accelerometer is active. This allows a sampling rate of 2 kHz. For our use case, rotational movement of the main spindle with less than $20\,000\,\mathrm{min}^{-1}$ (about 333 Hz) can be expected, so the sampling rate of 2 kHz is sufficient to capture the expected vibrations and harmonics.

We chose this sensor over more sophisticated measurement equipment, because it delivers a reasonable good resolution and accuracy. The low price and ease of application make it a good fit for retrofitting existing machines. Besides, an exact acoustic analysis is not in the scope of this work. For mounting the sensor, a position close to the main spindle of the machine was chosen. The machine can be seen in Figure 2a with the sensor position circled. Figure 2b shows a closer view of the mounting position. The part on which the sensor is mounted moves with the spindle in X, Y, and Z directions.

3. Proposed Approach

We propose a novel workflow to automatically identify anomaly effects in CNC milling machines. Figure 3 depicts a simplified overview of this approach. First, the data acquisition takes place (see Section 2) followed by a data conversion. Data conversion parses the raw data extracted from the machine to a readable format for the analysis. The remaining procedure of this workflow is two-fold: (1) dividing the raw machine signals into different phases, each representing a certain production step, and (2) testing the individual phases for anomaly effects. Here,

(a) CNC milling machine. Circle: Position of the additionally applied vibration sensor on top of the machine.

(b) Closer view on the mounting position of the vibration sensor. Compared to Figure 2a, the z-axis is rotated by 90 degrees.

Figure 2: CNC machine with sensor mounting position.

the first part itself consists of two steps: (1) an on/off[1] recognition, and (2) a production step identification for the on phases. Since the proposed approach is an automated workflow, the output of each step is directly used as input for the next step. The operator only needs to set one parameter which is the number of on phases.

3.1. Phase Detection

The aim of this procedure is to split the raw data streams into signals that can be unambiguously assigned to specific work steps in the manufacturing process. This step must be carried out in order to achieve comparability of individual production steps across several manufacturing processes. As the overall objective is to detect anomaly effects for individual tools in the CNC milling machine, a comparison of entire data streams of the manufacturing processes does not provide useful results, because the variations within the signals are too small and vanish into the data set. The procedure of dividing the raw data streams into phases again requires two main steps: (1) splitting

[1] An on phase represents the usage of a tool. In an off phase, the rotation of the spindle is stopped and the tool change takes place.

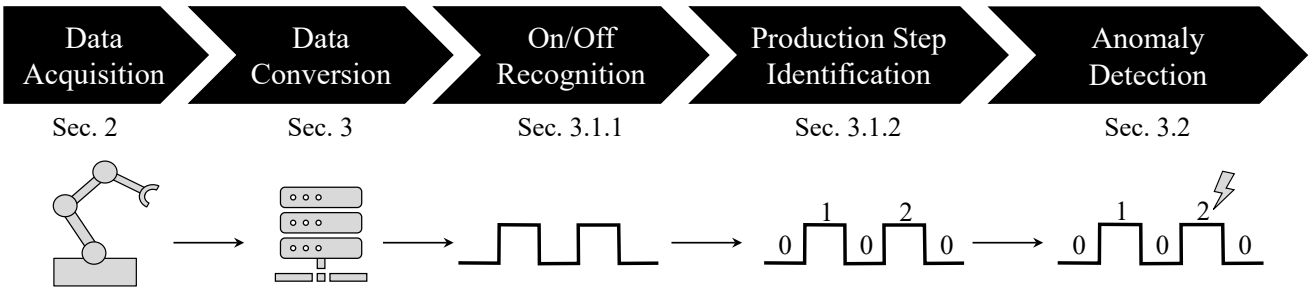


Figure 3: The simplified overall workflow. First, the data is retrieved directly from the machine and an additional vibration sensor. Second, an optional data conversion step is applied if the raw data is not available in a format readable for analysis. Third, the workflow divides the data into on and off phases and subsequently maps the phases that perform the same manufacturing process to a common identifier. Finally, the workflow trains a machine learning model for each type of on phase to detect the anomalies.

the raw data into on/off phases, and (2) mapping each on phase to a common identifier. These mechanisms are necessary because anomalies can only be detected for individual tools with their respective manufacturing processes. As mentioned previously, it is not feasible to extract the phases from the NC program steps, as this would limit the applicability of the approach. This means that it would require domain knowledge and would make it impossible to transfer the approach to different domains or use cases where this data is not accessible.

3.1.1. On/Off Recognition of Tool Change Phases

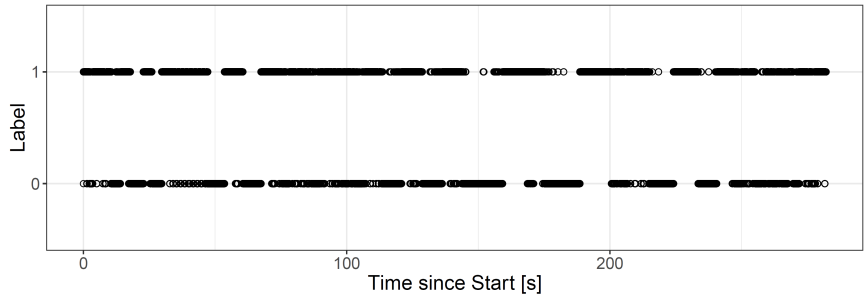
As anomaly effects of individual tools will not cause changes in the machine signals during the tool change phases, these need to be filtered out. For this purpose, the raw machine signals are passed to this stage of the algorithm. Then, this data is used to apply a *k-means* clustering [21]. Here, k is set to two, as this component of the workflow is only supposed to identify on and off phases. This means that each measurement timestamp is either mapped to cluster label 1 (on phase) or cluster label 0 (off phase) according to its machine signals. However, using *k-means* on the noisy raw data results in many mislabelings. Figure 4a shows this behavior. The horizontal axis shows the time since measurement start. The vertical axis shows the cluster label for each timestamp, that is, either 1 or 0 representing on or off respectively. Each circle in the figure indicates whether the respective set of features is clustered as an off or on phase. In an optimal setting, there should be long sequences of equal cluster labels without any breaks. These long sequences would represent an individual phase. Yet, there are many breaks (i.e., fast changes between the cluster labels) within the long sequences that indicate mislabeling. Thus, the proposed workflow comprises a systematic threshold-based smoothing of the provided cluster labels.

In order to remove mislabelings, first the derivative x' of the cluster labels is calculated.

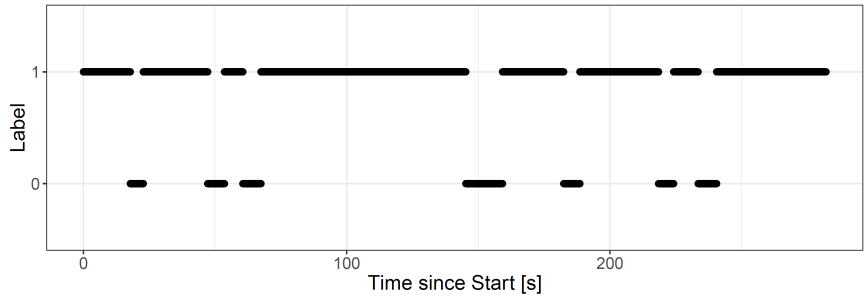
$$x' = \frac{dx}{dt} \tag{1}$$

Since the signal is a discrete time sequence, the derivative x' equals the first order difference of the time series x:

$$x'_i = x_{i+1} - x_i, \text{ for } i \text{ in } 1, ..., n-1 \tag{2}$$



(a) Clustering with many mislabelings (interrupted sequences of the same cluster label).



(b) Clustering without mislabelings (clear boundaries between sequences of different cluster labels).

Figure 4: One manufacturing process divided into on (1) and off (0) phases according to the cluster labels.

Afterwards, the indices of timestamps with a derivative not equal to zero are extracted. This ordered set of indices p indicates potential label changes:

$$p = \{0\} \cup \{i | x'_i \neq 0, \text{ for } i \text{ in } 1, ..., n-1\} \cup \{n\} \tag{3}$$

Further, it is necessary to check whether the potential cluster label change might occurred due to a new phase or due to incorrect labeling during a phase. For this purpose, the distance between the indices is calculated. If the distance is higher than a certain threshold t, a new phase is estimated. Otherwise, the label changes are treated as mislabelings. The ordered set of final cluster label changes is calculated as:

$$c = \{p_j | \, (p_j - p_{j-1}) > t, \text{ for } j \text{ in } 2, ..., |p|\} \tag{4}$$

The threshold t has a default value of 100 ms, but it can also be set by the operator in case the tools are changed or the data is recorded with another frequency. In the case of incorrect cluster labels, i.e., p_j that are not included in c, the label of the next accepted cluster is assigned. Figure 4b illustrates the resulting on/off recognition after applying this procedure. The scattered cluster labels are now merged to longer sequences of identical cluster labels.

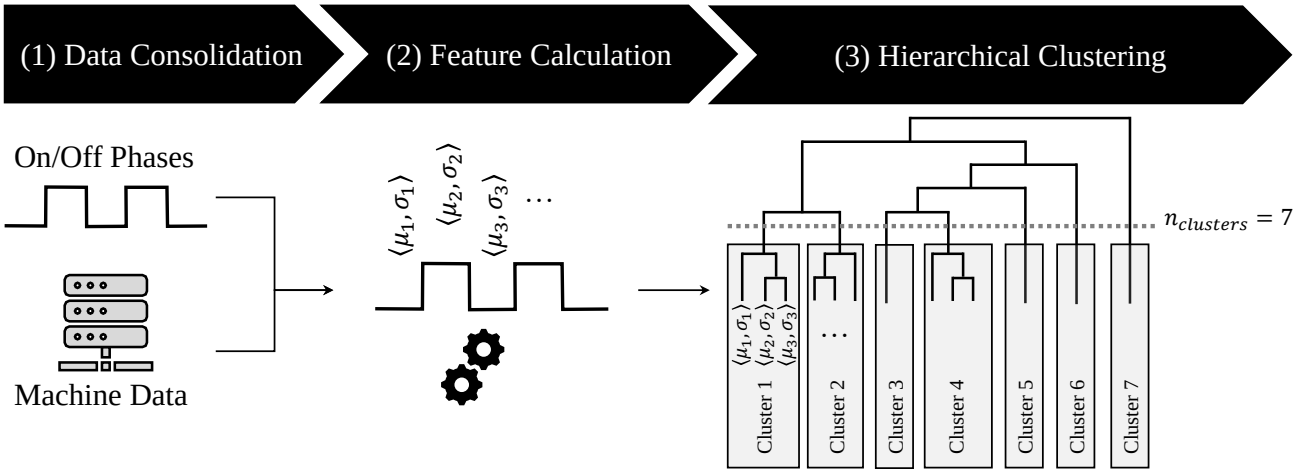


Figure 5: Simplified production step identification and mapping using hierarchical clustering (see Section 3.1.2). First, the workflow merges the machine data and timestamps of the on and off phases. Then, it derives features of the resulting machine data segments and generates a dendrogram from them. Depending on the number of on phases, the dendrogram is cut into different clusters, each representing a specific manufacturing step.

3.1.2. Production Step Identification and Mapping

After splitting the raw signal into on/off phases, the workflow needs to map the individual production steps to each other. Here, each production step is assigned a specific tool with its specific production sequence. Given that an anomaly in a machine tool typically does not affect the production steps of other tools, the workflow only identifies anomalies for such individual steps. Based on this mapping, the specific phases can be grouped across the entire manufacturing processes. Figure 5 schematically illustrates the workflow of the production step identification and mapping procedure. As input, the algorithm receives the raw machine signals and the cutting points between on and off phases in order to separate the machine data phasewise into segments (see step (1) in Figure 5). Then, features are calculated for each individual phase (see step (2) in Figure 5). Preliminary tests showed the best results for the measures mean and standard deviation as features. Finally, the workflow applies *hierarchical clustering* [22] on the feature data to map similar phases to a common identifier (see step (3) in Figure 5). Hierarchical clustering generates a tree structure, also called dendrogram, that either: (1) starts with all instances in one cluster and then splits each cluster into two clusters until all clusters consist of one instance only (divisive, top-down), or (2) begins with an own cluster for each instance and then merges two clusters into one until only one cluster is left (agglomerative, bottom-up). Here, the workflow applies the bottom-up approach with centroid linkage as agglomeration method. This means that the centroid of each cluster is determined and the distance between centroids a and b of two clusters A and B is calculated as $D_{\text{centroid\_linkage}}$:

$$D_{\text{centroid\_linkage}}(A, B) = d(a, b) \tag{5}$$

Therefore, the workflow employs the Euclidean distance as distance metric d, where l is the dimension of the feature space:

$$d\left(a,b\right) = ||a-b||_2 = \sqrt{\sum_{i=1}^{l}\left(a_i - b_i\right)^2} \tag{6}$$

However, a cutting point for the dendrogram is required that specifies the desired number of clusters. This parameter needs to be set by the operator. Yet, for a particular manufacturing process, the production steps are known in advance and thus, the number of clusters is equal to the number of production steps plus one for the tool change phase.

3.2. Machine Learning-based Anomaly Detection Approach

Based on the preprocessed signals, we learned multiple machine learning models within the workflow to detect the anomaly effects and compare their performance against each other for this task. As an alternative, we also implemented a typical order analysis method, namely resampling the signal to the order domain before computing the Fourier transformation [23]. However, this common procedure did not provide satisfactory results (see Section 4.3.1). For this reason, the focus is on machine learning methods, which are integrated into the workflow by applying these algorithms to the individual phases identified by the previously presented methods. The purpose of this step is to learn the correlation between anomaly effects and intrinsic characteristics of the machine signals. However, a typical problem for machine learning and artificial intelligence applications is the amount of data. The training of these methods strongly depends on the amount and variety of data, which in turn is the basis for a good generalizability and predictive power of the resulting model. Thus, the workflow first enlarges the data set to increase the amount of instances for training and applies the machine learning afterwards. In the following paragraphs, these steps are explained in more detail.

In the workflow, the vibration signals are used for the detection of anomaly effects. An example of the recorded signals from the used three axis vibration sensor is depicted in Figure 6. This sensor has a comparably high sampling rate of 2 kHz which allows a resampling of the measurements. This means that the original vibration data streams of the individual phases are split into r resampled data streams. Each new data stream ds_i starts at the i-th position, $i = 1, \ldots, r$. After adding a value v_i of the original data stream to the resampled data stream, $r-1$ values are skipped until the next value is added. This resampling strategy can be described as:

$$ds_i = \left(v_i, v_{i+r}, v_{i+2r}, \ldots\right) \quad , \quad i = 1, \ldots, r \tag{7}$$

Then, these resampled data streams of the individual phases are used to calculate a variety of statistical features describing the intrinsic behavior of the machine. For this purpose, the following characteristics were selected:

- Mean: represents a base level around which the vibration signal varies.

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{8}$$

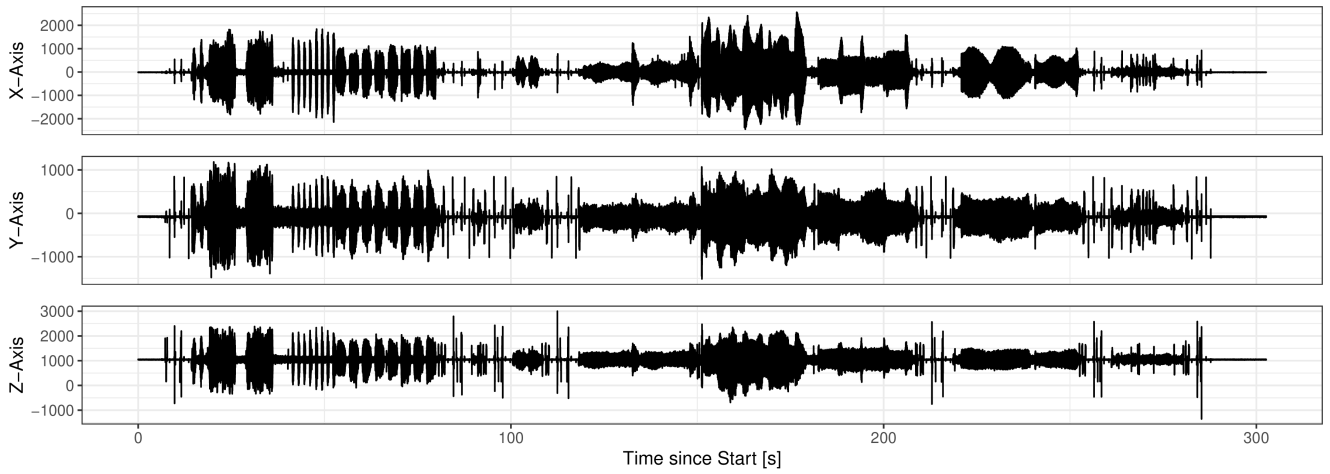


Figure 6: An example of the vibration signals of the three axis vibration sensor.

- Median: similar to the mean but less outlier-sensitive. A large difference between the mean and the median implies high outliers.

$$\tilde{x} = \frac{1}{2}\left(x_{\lfloor (n+1)/2 \rfloor} + x_{\lceil (n+1)/2 \rceil}\right) \tag{9}$$

- Standard deviation: describes the amount of variation within the signal.

$$\tilde{s} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2} \tag{10}$$

- Skewness: measures the asymmetry of a probability distribution function around its mean value.

$$\tilde{\mu}_3 = \frac{m_3}{m_2^{3/2}} \quad \text{with} \quad m_k = \frac{1}{n}\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^k \tag{11}$$

- Kurtosis: similar to the skewness, the kurtosis describes the shape of the probability distribution function. Instead of the asymmetry, the kurtosis quantifies the steepness of the probability distribution function.

$$\tilde{\mu}_4 = \frac{m_4}{m_2^{4/2}} \tag{12}$$

- Root mean square (RMS): the square root of the mean of the squared values of the measured signal is directly related to the energy content of the vibration.

$$\mathrm{RMS} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} \tag{13}$$

- Crest factor: measures the extremeness of peaks in a waveform. A large value indicates high peaks compared to the RMS of the waveform.

$$C = \frac{\max(|x|)}{\text{RMS}} \tag{14}$$

- Gradient: β_1 represents the steepness of the measured signals. Therefore, a linear model $f(t_i; \beta_0; \beta_1)$ of the measured signals is fitted on the measurement time t_i.

$$\beta_1 \quad \text{with} \quad f(t_i; \beta_0; \beta_1) = \beta_0 + \beta_1 t_i \tag{15}$$

- Peak to peak (PTP): provides the total range between minimum and maximum value.

$$\text{PTP} = \max(x) - \min(x) \tag{16}$$

The output of this step is a vector of features for each resampled data stream of each individual phase. Paired with a class label indicating whether an anomaly is present or not, such a vector makes up a single instance used for training the machine learning techniques. The proposed approach assumes labeled training data since it is based on supervised machine learning methods. In terms of evaluating the approach, the class label is to be predicted by the machine learning method, so only the features without labels are passed to the method. Afterwards, the predicted class label is compared to the actual one. As machine learning methods, we selected naive Bayes, decision tree, k-nearest neighbor [24], random forest [25], support vector machine [26], and XGBoost [27], while a simple logistic regression model serves as a basis for comparison.

3.3. Implementation

The workflow is implemented in `R` functions that invoke each other. For this purpose, the output of each component of the workflow is passed on to the next component, resulting in an automatic execution. `R` provides many useful libraries for data processing and machine learning. The libraries used for the machine learning models are presented in Section 4.3.2. Also, the machine learning methods applied are easily interchangeable in the code.

4. Case Study and Experimental Results

To assess the performance of the proposed automatic anomaly detection workflow, we conducted a real-world case study. First, Section 4.1 presents the experimental setup. Then, Section 4.2 shows results on the phase detection step followed by Section 4.3, which provides an evaluation of the anomaly detection part. Since, to the best of our knowledge, there is no end-to-end workflow that takes the entire machine signal as input and detects anomalies for individual tools, we do not compare the proposed workflow to related work. However, we analyze the suitability of different machine learning methods for the application within our end-to-end workflow with respect to their detection quality.

Table I: Overview over the phases in the recorded NC program. Phases 3 and 5 are used for the unbalance detection.

Phase	Description
-	Initial tool pickup: Cutter head ⌀ 63 mm
P1	Surface milling, circular deepening, rectangular deepening (3200 min^{-1})
-	Tool change: Insert drill ⌀ 24 mm
P2	Drill start bore for milling circular pocket (3000 min^{-1})
-	Tool change: Slot mill ⌀ 20 mm
P3	**Widening start bore (**2800 min^{-1}**)**
-	Tool change: Roughing cutter ⌀ 32 mm
P4	Pre-milling circular pocket, roughing rectangular deepening (2800 min^{-1})
-	Tool change: Slot mill ⌀ 20 mm
P5	**Roughing circular pocket (**4500 min^{-1}**)**
-	Tool change: HSC end mill ⌀ 20 mm
P6	Contour edges (17 000 min^{-1})
-	Return last tool to magazine

4.1. Experimental Setup and Data Generation

To simulate a faulty tool, we attached a small weight to a drill in radial direction. Although it does not interfere with the drilling process, it causes additional vibrations in the entire machine by acting as an imbalance in the rotation of the tool. For higher rotation speeds, this resulted in an audible oscillation, distinct from the usual operating noise of the machine. Unbalance effects cause increased wear and defects and should therefore be avoided. While sophisticated systems for the detection of tool wear and breakage are available on the market, the effort to apply them is usually high. The systems need to be integrated in the machine and require detailed process information. Therefore and for the straightforward way of emulation, an unbalance can be considered suitable as an example defect which allows us to examine a loosely coupled detection system with little included process knowledge.

As for the process, an NC program was created that cuts a 5 mm deep shape into the surface of an aluminium block. This process was chosen, as it can easily be repeated without requiring large amounts of material by simply lowering the zero reference by 5 mm between the runs. The prepared faulty tool is used in two drilling operations with two different rotation speeds (i.e., phases P3 and P5). In total, 5 different tools are used in 6 process steps. In these phases, the main spindle is rotating and they are referred to as on phases. In between these on phases, automatic tool changes take place where the spindle is stopped. Correspondingly, these are called off phases. The on phases in which the tool with the prepared unbalance is active are termed unbalance phases. Table I gives an overview over the phases in the recorded NC program.

For our experiments, two blocks of aluminum were used and in total we recorded data from 30 runs of the described machining process. Of these, 13 had the unbalance attached, as described above. Additionally, we recorded 7 runs (including 3 with an unbalance) where no material was processed. This means that the tools were moving through air only. The data acquired during the case study and used for the evaluation of the proposed

approach will be published on Zenodo[2] upon acceptance.

4.2. Accuracy of the Phase Detection Component

Here, the accuracy of the phase segmentation (i.e., on/off phase recognition) and production step identification and mapping are assessed. The evaluation of the on/off phase recognition algorithm is done by comparing the timestamps of the phase edges provided by our algorithm to the timestamps captured within the NC data. We capture the NC data only as a gold standard to evaluate the accuracy of the on/off recognition algorithm, as narrowing the proposed workflow to scenarios where NC data is available would limit the applicability of the approach (see Section 2). The parameter for the threshold-based smoothing was set to $t = 100\,\text{ms}$. This parameter was empirically determined in order to allow a meaningful distinction between on and off phases. Table II depicts the results in terms of time differences. Each column represents a change between two phases and a phase change ID with an asterix symbolizes that the phase changed from an on phase to an off phase. In contrast, phase change IDs without an asterix show phase changes from off phases to on phases. For each of these phase change IDs, Table II shows the mean time difference, the standard deviation of the difference, and the interquartile range of the difference. All units are depicted in milliseconds. Here, a positive average time difference means that the time calculated by the algorithm is later than the timestamp from the NC data, while a negative value indicates the opposite. Regarding the mean difference, a clear distinction between phase changes from on to off phases versus phase changes from off to on phases can be seen. Phase changes from off to on phases lie in a border of around 300 ms to 360 ms. In contrast, phase changes from on to off phases only show a difference of about 65 ms to 140 ms, except for the last phase change, which has an average time difference of almost 550 ms. Obviously, these deviations are not sufficient for a clear distinction between different phases. However, the standard deviation as well as the interquartile range show that the variation between the runs is very small. Contrary to the mean values, the standard deviation and the interquartile range exhibit smaller values for phase changes from off to on phases. The interquartile range is of special interest since it represents that over all phase changes, half of all values lie in a range of at max 35.5 ms. For 5 out of the 12 phase changes, this border is even much smaller, that is, at max 2 ms. This shows that the proposed on/off recognition algorithm identifies the timestamps very well. Yet, the identified timestamps differ by a rather fixed offset compared to the timestamps of the NC data. However, this fixed offset can be determined during a calibration phase and therefore does not affect the quality of the on/off recognition. Moreover, the offset can be explained by the fact that the NC data might use another point in time as phase change than the algorithm.

The production step identification and mapping algorithm, which is executed upon the detected on and off phases, even achieves a perfect matching with an accuracy of 100%.

[2]Link to the data set: `https://zenodo.org` (data will be uploaded upon acceptance)

Table II: Mean [ms], standard deviation [ms], and interquartile range [ms] of the deviation between the timestamps of the predicted phase change and recorded phase change. A * indicates that the phase changed from a producing step (on) to a tool change phase (off).

Phase Change ID	CH1	CH2*	CH3	CH4*	CH5	CH6*	CH7	CH8*	CH9	CH10*	CH11	CH12*
Mean	356.957	98.957	313.913	78.087	301.348	66.870	301.000	112.522	301.217	137.174	305.696	-548.826
Standard Deviation	18.190	16.339	1.041	13.588	0.982	8.081	1.243	14.235	1.043	10.836	0.974	11.472
Interquartile Range	35.5	32.5	2.0	26.0	1.0	14.5	2.0	27.5	1.0	16.5	1.0	22.0

4.3. Prediction Quality of the Anomaly Detection Component

To identify the degradation effects caused by the attached unbalance, we applied two approaches. First, we used acoustic analysis methods to examine the oscillations. However, since the results were not convincing, we applied a second approach using machine learning techniques (see Section 3.2). In the following, the results of the order analysis are discussed first, followed by the results of the machine learning methods.

4.3.1. Acoustic Analysis

Unbalance, as an example for an anomaly, often results in audible effects, which we also experienced during the data acquisition for our experiments. The use of acoustical methods seems natural. We therefore also aimed to apply a basic order spectrum analysis to find a differentiation between good and anomalous measurements.

The occurring frequencies in the signal greatly depend on the rotational speed of the main spindle of the investigated machine. A more general measure for varying speeds than the frequency spectrum is the order analysis [23]. Here, the energy of the signal is displayed over the orders and not the frequency. The orders are multitudes of the rotational frequency and provide a normalization for varying rates of rotation. While order analysis is often applied for changing rotation speeds to get a spectrogram over an acceleration ramp, it can still be helpful for normalizing different constant rotational speeds. The transformation can be simplified in such constant cases. In our application, the spindle is only accelerated once in every phase, after picking up the tool, and it then maintains the programmed speed. The variation we see in this speed is negligible with a deviation of only about $\pm 5\,\mathrm{min}^{-1}$. Thus, assuming a stationary case is valid.

To apply this, we compute the fast Fourier transformation (FFT) and apply Welch's method [28] to average out noise effects. We use a window size of 1000 samples, which equals 0.5 s, with an overlap of 50% and apply the Hanning window function. The frequency values are transformed to orders by division through the fundamental order f_0, which is the average rotational frequency of the spindle for this segment.

In case of an unbalance, we expect that the effect is also visible in the spectrum. Specifically, we expect the energy of the first order to be significantly higher [29]. In Figure 7, an overlay of all spectra for the good and the unbalance measurements for the second unbalance phase (i.e., the fifth on phase P5) can be seen. The comparison shows that the shapes of the good and the unbalance curves are similar. While we see no prominent difference in the first order, it is evident that the overall level is increased. The first unbalance phase (i.e., the third on phase P3) exhibits no difference at all and it is therefore not depicted here.

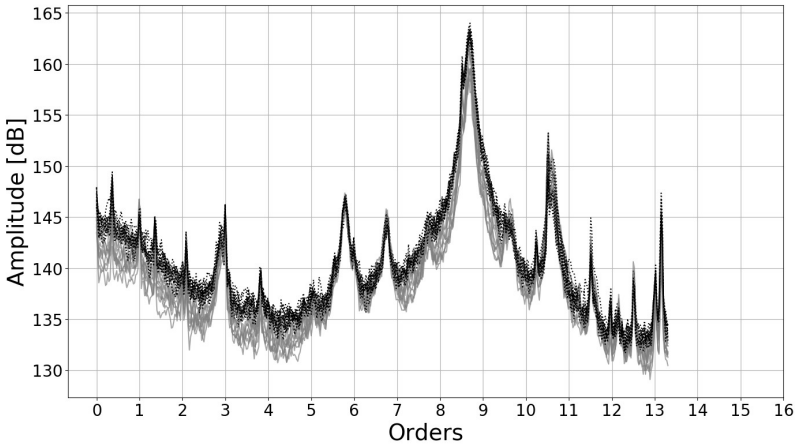


Figure 7: Order spectra of all measurements in X direction for the second unbalance phase (i.e., phase P5). Measurements with an attached unbalance are displayed in black, measurements without unbalance in gray.

Differences between the two unbalance phases could be explained by the higher rotation speed in the second unbalance phase, as we expect in general higher excitation levels and, therefore, a greater impact of the unbalance. The lack of an increased first order might indicate that the unbalance, while having an audible effect, is not impacting the structure-borne sound as much. While the slight effect is visible in the plot, it is difficult to derive criteria that separate both classes from it.

With this result, it can be seen that order analysis as a standalone classifier or as a preprocessing step is not as sensitive to the anomaly as expected. For different anomaly effects, it might be even less effective. We conclude that the acoustical approach is not suitable for our simplified means of data acquisition and that a more generic approach would benefit the applicability.

4.3.2. Comparison of the Machine Learning Methods

To expand the data set, we resample the original vibration signal to a lower sampling rate (see Section 3.2). For our experiments, we set the resampling factor r to 4. Therefore, the actual sampling rate of the vibration sensor is reduced from 2 kHz to 500 Hz. Since we focus on automatic anomaly detection, the machine learning task is a binary classification. Hence, we have selected accuracy, precision, recall, F1-score, and Matthews correlation

coefficient (MCC) [30] as evaluation metrics:

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}$$
$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$
$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$
$$\text{F1-score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$
$$\text{MCC} = \frac{(\text{TP} \cdot \text{TN}) - (\text{FP} \cdot \text{FN})}{\sqrt{(\text{TP} + \text{FP}) \cdot (\text{TP} + \text{FN}) \cdot (\text{TN} + \text{FP}) \cdot (\text{TN} + \text{FN})}}$$

Here, TP represents the number of instances correctly classified as anomalous, while TN shows the number of instances correctly classified as normal. Instead, FP and FN reflect the number of instances falsely classified as anomalous and normal, respectively. To examine the predictive power of the learned machine learning models, we split the data into a training and a testing set. For this purpose, we used 70% of the data for model learning and the remaining 30% of the data to validate the respective model performance. We are aware that a single split between training and testing data can lead to an arbitrary ranking of the machine learning models. Therefore, we performed the experiment 100 times with random splits between the training and testing sets. However, we ensured that resampled signals originating from the same base signal are either used for training or testing. Otherwise, the results would be biased since the resampled vibration signals are very similar to the original signal. As introduced in Section 3.2, logistic regression, naive Bayes, decision tree, k-nearest neighbor, random forest, support vector machine, and XGBoost are applied on the data to detect the unbalance. The used parametrizations of the algorithms as well as the utilized `R` libraries are listed in Table III.

The results for the first phase with an unbalance (i.e., phase P3) are shown in Figure 8. Since we conducted 100

Table III: Parametrization and used libraries for the machine learning methods. The workflow is implemented in `R`.

Method	Library	Parameters
Logistic Regression	`stats::glm`	family = binomial(link = "logit")
Naive Bayes	`e1071` [31]	laplace = 1
Decision Tree	`tree` [32]	split = "gini", method = "recursive.partition"
k-Nearest Neighbor	`class` [33]	k = 5
Random Forest	`randomForest` [34]	ntree = 500, mtry = 10
Support Vector Machine	`e1071` [31]	scale = FALSE, kernel = "polynomial", degree = 3, coef0 = 0, cost = 5, shrinking = TRUE, epsilon = 0.1
XGBoost	`xgboost` [35]	booster = "gbtree", eta = 0.1, max_depth = 10, gamma = 0.5, subsample = 0.75, colsample_bytree = 1, objective = "binary:logistic", eval_metric = "error", eval_metric = "logloss", nrounds = 1000, watchlist = list(train, val)

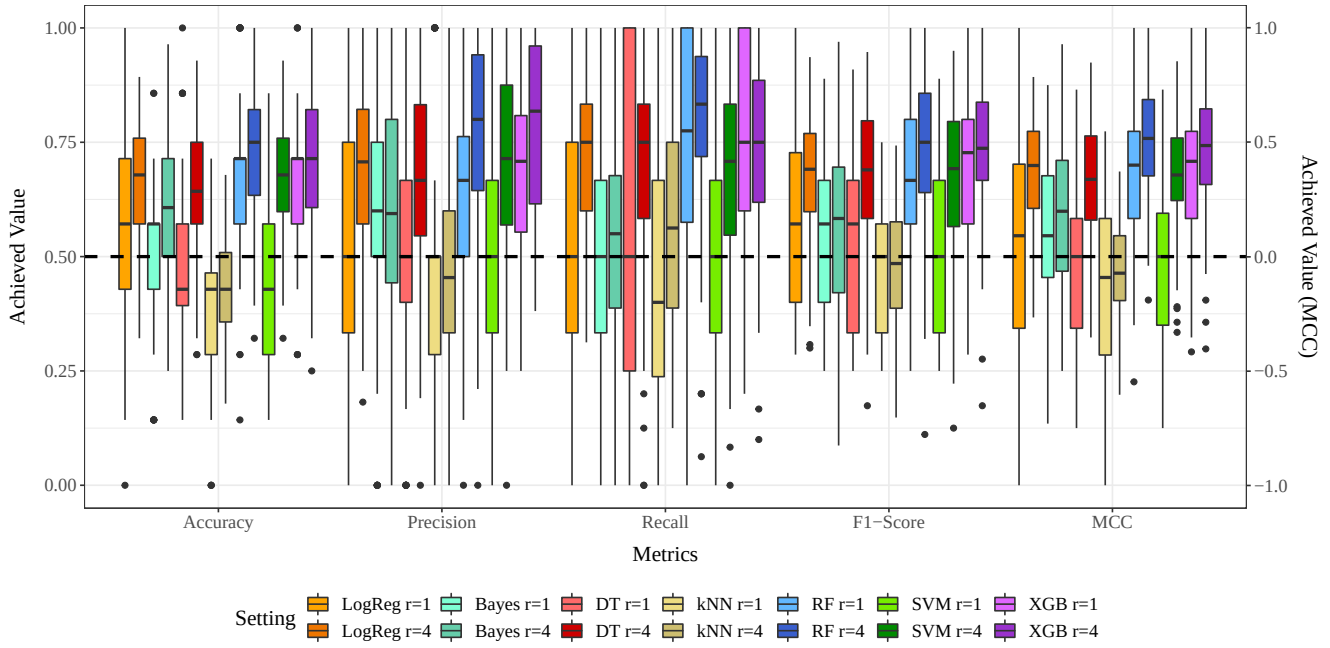


Figure 8: Achieved values of the machine learning methods logistic regression (LogReg), naive Bayes (Bayes), decision tree (DT), k-nearest neighbor (kNN), random forest (RF), support vector machine (SVM), and XGBoost (XGB) for each evaluation metric for the first unbalance phase (i.e., phase P3). The value of r indicates the resampling factor. That is, for $r = 1$, no resampling is used. The dashed horizontal line shows the theoretical baseline of the application of random guess.

experiments per machine learning method, the figure depicts box plots of the distributions of the achieved values. The horizontal axis presents the different evaluation metrics, while the left vertical axis shows the achieved results for the metrics accuracy, precision, recall, and F1-score. In contrast, the right vertical axis is used for Matthews correlation coefficient, as this metric ranges from -1 to 1 instead of 0 to 1. Similar to the other four metrics, a larger value of MCC indicates a better prediction. That is, an MCC of 1 corresponds to a perfect prediction and -1 represents the worst possible prediction. Each of the seven different colors represents one of the machine learning methods: logistic regression (orange), naive Bayes (aquamarine), decision tree (red), k-nearest neighbor (golden), random forest (blue), support vector machine (green), and XGBoost (purple). Moreover, the brightness of the color illustrates whether the resampling method was utilized or not. To this end, lighter shades stand for $r = 1$ and, thus, the resampling method was not applied, whereas the darker shades represent the results achieved using a resampling factor of $r = 4$. The medians of accuracy, precision, recall, and F1-score range from about 0.40 up to about 0.80 (left y-axis). For the Matthews correlation coefficient, the achieved medians lie between -0.10 and 0.50 (right y-axis). First, it can be seen that the k-nearest neighbor classifier performed worst. Next, logistic regression, naive Bayes, decision tree, and support vector machine without resampling achieved rather arbitrary results. That is, their achieved values are close to the random guess (0.50 for accuracy, precision, recall, and F1-score and 0.00 for Matthews correlation coefficient). In contrast, random forest and XGBoost without resampling already provided reasonable outcomes that differ significantly from the random guess. As first insight, the detection performance of random forest and XGBoost prove that the unbalance detection is possible. Second, Figure 8 shows that applying the resampling approach with a factor of $r = 4$ improves the performance of all machine learning methods for all evaluation metrics. This can be seen by the fact that the median values achieved by the approaches using resampling are almost always higher compared to the respective approach without resampling. Only in the cases of precision when applying naive Bayes and F1-score when using k-nearest neighbor classifier, the median detection quality slightly decreased. Moreover, the 25th percentiles (i.e., the lower end of the box) enhance for all methods when using resampling. Regarding the 75th percentiles (i.e., the upper end of the box), resampling outperformed the original signal in 31 out of 35 cases. Only the metric recall achieved a higher third quartile when applying decision tree, random forest, or XGBoost without resampling and the metric Matthews correlation coefficient for the application of k-nearest neigbor classification. Furthermore, the application of resampling improves the arbitrary results of logistic regression, decision tree, and support vector machine to results that differ significantly compared to random guesses. Nevertheless, they do not achieve the performance of random forest and XGBoost with resampling.

To compare the performance of the machine learning methods in numbers, Table IV summarizes the average values achieved for each setting and each evaluation metric. Again, the results show that resampling the vibration signal improves the average performance of all machine learning methods for all evaluation aspects, besides recall when using XGBoost as classification method. However, the degradation is only marginal. The best performing method for detecting the anomaly in the first unbalance phase is random forest. It outperforms all other machine learning methods in terms of accuracy, recall, F1-score, and Matthews correlation coefficient. Yet, regarding precision, XGBoost achieves a higher value than random forest. The overall worst performance is shown by the

k-nearest neighbor classifier, followed by naive Bayes. The single decision tree, logistic regression, and support vector machine achieve comparable mediocre results that cannot compete with random forest and XGBoost in this experiment.

Table IV: Mean μ of the machine learning metrics for resampled ($r = 4$) and non-resampled ($r = 1$) signals of the first unbalance phase (i.e., phase P3). The best values of each metric are printed in bold.

Model	**LogReg**		**Bayes**		**DT**		**kNN**		**RF**		**SVM**		**XGB**	
Resamp. Factor	$r = 1$	$r = 4$	$r = 1$	$r = 4$	$r = 1$	$r = 4$	$r = 1$	$r = 4$	$r = 1$	$r = 4$	$r = 1$	$r = 4$	$r = 1$	$r = 4$
Accuracy [μ]	0.529	0.671	0.499	0.591	0.474	0.649	0.391	0.438	0.661	**0.725**	0.461	0.669	0.667	0.717
Precision [μ]	0.532	0.688	0.596	0.602	0.505	0.675	0.446	0.479	0.665	0.761	0.523	0.705	0.702	**0.781**
Recall [μ]	0.534	0.709	0.479	0.540	0.552	0.698	0.444	0.563	0.769	**0.786**	0.482	0.674	0.762	0.752
F1-Score [μ]	0.569	0.672	0.531	0.554	0.541	0.669	0.455	0.481	0.677	**0.736**	0.505	0.670	0.693	0.735
MCC [μ]	0.069	0.360	0.062	0.193	-0.036	0.323	-0.128	-0.069	0.393	**0.505**	-0.042	0.352	0.363	0.464

Besides the average prediction performance, the robustness of the learned models is also of high importance. For this purpose, the variation in the predictive power over the randomized experiments is considered. The advantage of more robust models is that the quality of the prediction can be trusted more, whereas less robust models show a larger variation in their prediction quality. Table V presents the standard deviations of the prediction performance over all 100 experiments for each machine learning method. Here, k-nearest neighbor with resampling achieves the lowest standard deviation on the metrics accuracy, precision, F1-score, and MCC. Only for recall, logistic regression with resampling reaches a smaller standard deviation. However, as presented in Table IV, the overall predictive power of k-nearest neighbor and logistic regression cannot compete with random forest and XGBoost in our use case. Moreover, the table shows that the resampling technique reduces or at least maintains the standard deviation for all machine learning methods and evaluation metrics. In some cases, the reduction is even substantial. Therefore, it can be concluded that the proposed resampling method not only improves the results on the first unbalance phase on average, but also makes the machine learning models more robust.

Table V: Standard deviation σ of the machine learning metrics for resampled ($r = 4$) and non-resampled ($r = 1$) signals of the first unbalance phase (i.e., phase P3). The best values of each metric are printed in bold.

Model	**LogReg**		**Bayes**		**DT**		**kNN**		**RF**		**SVM**		**XGB**	
Resamp. Factor	$r = 1$	$r = 4$	$r = 1$	$r = 4$	$r = 1$	$r = 4$	$r = 1$	$r = 4$	$r = 1$	$r = 4$	$r = 1$	$r = 4$	$r = 1$	$r = 4$
Accuracy [σ]	0.196	0.126	0.166	0.152	0.183	0.151	0.173	**0.105**	0.173	0.153	0.178	0.130	0.152	0.147
Precision [σ]	0.314	0.192	0.273	0.273	0.231	0.207	0.282	**0.185**	0.226	0.221	0.264	0.212	0.200	0.189
Recall [σ]	0.299	**0.182**	0.258	0.243	0.339	0.223	0.308	0.208	0.262	0.205	0.260	0.217	0.239	0.195
F1-Score [σ]	0.181	0.143	0.169	0.201	0.187	0.163	0.148	**0.130**	0.179	0.167	0.169	0.166	0.165	0.147
MCC [σ]	0.417	0.231	0.331	0.328	0.358	0.281	0.394	**0.209**	0.325	0.253	0.364	0.251	0.313	0.276

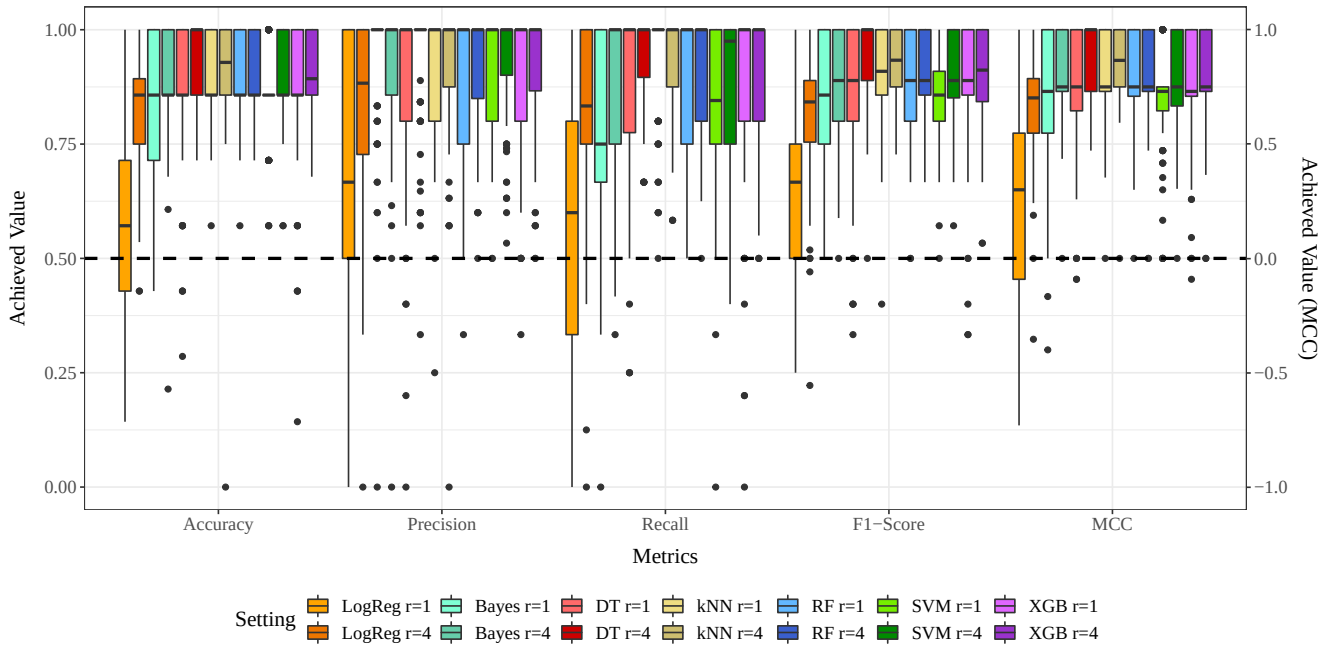


Figure 9: Achieved values of the machine learning methods logistic regression (LogReg), naive Bayes (Bayes), decision tree (DT), k-nearest neighbor (kNN), random forest (RF), support vector machine (SVM), and XGBoost (XGB) for each evaluation metric for the second unbalance phase (i.e., phase P5). The value of r indicates the resampling factor. That is, for $r = 1$, no resampling is used. The dashed horizontal line shows the theoretical baseline of the application of random guess.

Figure 9 presents the achieved machine learning metrics for the second phase an unbalance is attached to (i.e., phase P5). The layout and color-coding of the figure is similar to Figure 8. However, compared to Figure 8, the performance is much better and in most cases, the 75th percentile even reaches 1.00. Similar to the first unbalance phase, the predictive power of logistic regression learned without resampling achieves is rather poor. In fact, logistic regression yields by far the worst results for this unbalance phase. Yet, for this unbalance phase, naive Bayes and support vector machine are able to keep up with random forest and XGBoost. Furthermore, the single decision tree and k-nearest neighbor do not only keep up with random forest and XGBoost, but they even outperform them. The overall best method for the second unbalance phase is therefore represented by decision tree with resampling, yielding the highest value for all evaluation metrics. Again, applying the resampling approach performs better or at least as well as the approach that simply uses the original vibration data in most cases. There are only few cases where the original signal yields better results, namely the 25th percentile of naive Bayes when considering precision, k-nearest neighbor in terms of recall, and XGBoost regarding F1-score. The reason for the significantly improved unbalance detection is most probably the increased rotational speed. In contrast to the phase depicted in Figure 8, the rotational speed was increased from $2800\,\mathrm{min}^{-1}$ to $3500\,\mathrm{min}^{-1}$.

Table VI shows the mean values achieved on each evaluation metric for all machine learning methods with and without resampling. In most cases, the proposed resampling increases the average predictive power significantly. Yet, the k-nearest neighbor classifier does not seem to benefit from the resampling, although it increases its precision, F1-score, and MCC by a small amount. However, the accuracy is slightly decreased and the recall drops by a significant amount. The only other case of a negative effect employed by resampling is the precision when applying naive Bayes. As already mentioned for Figure 9, k-nearest neighbor and decision tree surpass random forest and XGBoost for the second unbalance phase. More precisely, k-nearest neighbor without resampling achieves the highest recall, while the single decision tree outperforms all other methods regarding accuracy, precision, F1-score, and MCC. In contrast to the first unbalance phase, naive Bayes and SVM also reach competitive results and keep up with random forest and XGBoost. Only logistic regression cannot compete with the other six methods.

Table VI: Mean μ of the machine learning metrics for resampled ($r = 4$) and non-resampled ($r = 1$) signals of the second unbalance phase (i.e., phase P5). The best values of each metric are printed in bold.

Model	**LogReg**		**Bayes**		**DT**		**kNN**		**RF**		**SVM**		**XGB**	
Resamp. Factor	$r=1$	$r=4$	$r=1$	$r=4$	$r=1$	$r=4$	$r=1$	$r=4$	$r=1$	$r=4$	$r=1$	$r=4$	$r=1$	$r=4$
Accuracy [μ]	0.593	0.817	0.833	0.890	0.857	**0.927**	0.914	0.912	0.879	0.908	0.860	0.890	0.857	0.907
Precision [μ]	0.623	0.834	0.940	0.915	0.882	**0.939**	0.912	0.923	0.875	0.926	0.908	0.933	0.889	0.921
Recall [μ]	0.586	0.827	0.771	0.886	0.890	0.940	**0.945**	0.925	0.908	0.917	0.845	0.879	0.883	0.910
F1-Score [μ]	0.609	0.813	0.833	0.885	0.863	**0.929**	0.915	0.919	0.875	0.909	0.860	0.891	0.869	0.902
MCC [μ]	0.214	0.649	0.687	0.804	0.742	**0.861**	0.838	0.839	0.770	0.809	0.699	0.776	0.739	0.805

The standard deviations for all settings on the second unbalance phase are depicted in Table VII. Similar to the first unbalance phase, the resampling technique increases the robustness of the machine learning methods which is shown by the reduction of the standard deviation for almost all settings. The only exceptions are shown for naive Bayes when considering precision and k-nearest neighbor regarding accuracy and precision. Contrary to the previous evaluations, the methods exhibiting the smallest standard deviation are more diverse here. While the single decision tree still yields the smallest standard deviation for accuracy and recall, the support vector machine reaches the lowest variation regarding precision. Finally, the smallest standard deviations for F1-score and MCC are achieved by the k-nearest neighbor classifier. Compared to the first imbalance phase, the standard deviations for all methods decreased, except for logistic regression regarding all evaluation metrics. The only other exception is k-nearest neighbor without resampling with regard to accuracy.

To summarize, the average anomaly detection performance increased from the first unbalance phase to the second unbalance phase as well as the standard deviation decreased. This behavior might be due to the fact that the rotational speed was increased from $2800\,\mathrm{min}^{-1}$ (first unbalance phase P3) to $3500\,\mathrm{min}^{-1}$ (second unbalance phase P5). In addition, the proposed resampling technique increases the predictive power and reduces the variations in predictive power significantly. While random forest with resampling reached the best overall predictive power on

Table VII: Standard deviation σ of the machine learning metrics for resampled ($r = 4$) and non-resampled ($r = 1$) signals of the second unbalance phase (i.e., phase P5). The best values of each metric are printed in bold.

Model	**LogReg**		**Bayes**		**DT**		**kNN**		**RF**		**SVM**		**XGB**	
Resamp. Factor	$r = 1$	$r = 4$	$r = 1$	$r = 4$	$r = 1$	$r = 4$	$r = 1$	$r = 4$	$r = 1$	$r = 4$	$r = 1$	$r = 4$	$r = 1$	$r = 4$
Accuracy [σ]	0.216	0.115	0.135	0.116	0.156	**0.085**	0.093	0.116	0.106	0.086	0.099	0.092	0.151	0.098
Precision [σ]	0.300	0.196	0.146	0.157	0.200	0.129	0.141	0.146	0.170	0.130	0.138	**0.119**	0.165	0.143
Recall [σ]	0.311	0.185	0.225	0.173	0.187	**0.107**	0.118	0.108	0.141	0.123	0.186	0.147	0.198	0.144
F1-Score [σ]	0.197	0.130	0.145	0.115	0.156	0.091	0.102	**0.075**	0.121	0.088	0.103	0.097	0.143	0.112
MCC [σ]	0.428	0.224	0.267	0.182	0.270	0.160	0.182	**0.154**	0.207	0.196	0.236	0.196	0.249	0.215

the first unbalance phase, the single decision tree with resampling outperformed the other methods for the second unbalance phase. However, random forest still performed well also on the second unbalance phase. Finally, the experiment has shown that the proposed workflow is able to accurately detect anomalies in machine tools when proper resampling is applied.

5. Discussion

To summarize the most important results, Section 5.1 briefly recapitulates the evaluation findings. While our goal was to introduce a most optimal and universally applicable solution, we are aware of some limitations of this work. These are discussed in Section 5.2.

5.1. Evaluation Findings

As a brief summary of the experimental real-world case study, the key takeaways are the following:

(I) Given the measured machine internal data and vibration signal, common acoustic analysis does not detect the anomalies well. This is mostly rooted in the requirements of these methods to receive data with high sampling rate and signal resolution, which further requires expensive sensors. The minor differences were not reliable enough to function as a classifier without human assessment. However, our goal was to achieve automatic detection with standard machine data and inexpensive sensors.

(II) The proposed machine learning-based workflow for automatic anomaly detection is able to decompose the data into segments for each production step and detect the anomalies with an F1-score of up to 92.9%. In addition, the case study shows that the proposed resampling technique of the original signal significantly improves the quality of the prediction.

(III) The proposed automatic workflow detects anomalies in machine tools with high accuracy even with little training data. This shows that the workflow can be quickly integrated into industrial production processes.

(IV) An increased rotational speed results in more accurate anomaly detection rates. This is consistent with the physical properties of an unbalance since higher speeds mean greater resulting forces, which in turn affect the measured signals. Yet, the means of detection might not be suitable for phases with slower rotational speeds.

(V) While only the ensemble learners random forest and XGBoost have shown a good predictive power on the first unbalance phase (i.e., the unbalance phase with lower rotational speed), the single decision tree and k-nearest neighbor even surpassed random forest and XGBoost with increased rotational speed. However, random forest and XGBoost still achieved a high predictive power for the increased rotational speed. Therefore, when choosing a single machine learning method, random forest should be preferred, as this method robustly allows accurate detection of machine tool anomalies.

5.2. Threats to Validity

The goal of this paper was not to develop a novel machine learning approach but to combine several existing base-level machine learning techniques in an intelligent way to automatize the procedure of anomaly detection in machine monitoring data, especially with base-level signals and inexpensive sensors. By demonstrating the applicability and effectiveness of such an automatic workflow based on rather simple machine learning techniques, we want to bring academic theory and industrial practice closer together. The proposed automatic workflow consists of three main steps: 1) on/off recognition, 2) production step identification and mapping, and 3) anomaly detection.

We are aware that 30 measurement runs only provide a small data set for machine learning models. In practice, however, it is also an important criterion that methods can be applied quickly without long measurement and training time. To achieve this good performance despite the few training data, the workflow integrates the resampling technique to artificially increase the amount of training data. The experimental results show that the automatic workflow already achieves good performance even with these few training data. However, increasing the amount of data would most likely further improve the model.

The main problem with most anomaly detection mechanisms for machine data is the reusability and transferabiltiy since they require profound domain knowledge and specific data that cannot be captured by other practitioners. However, the proposed workflow requires only one input by the practitioner, that is, the number of different production steps. In addition, the proposed automatic workflow only uses standard physical quantities (see Section 2) that can be measured on any comparable machine and therefore the approach is transferable.

For the data acquisition, the usage of the EWB tool also has some downsides. First of all, it is not designed for automatic data acquisition and user interaction is required for every measurement run. While we are aware that a productive use requires a fully automatic solution, this was not in the scope of this paper but can be implemented easily in the future. The approach should be easy to migrate to an automatic system, once these are available. Additionally, the EWB tool is specific for this vendor. To transfer this approach to other vendors, equivalent access to machine internal data is required. However, the focus of this work is on providing a generally applicable workflow and, therefore, the automatic extraction of signals is set as a requirement.

The evaluation also included sampling analog input signals via the machines PLC as an alternative to the sensors with integrated analog-to-digital conversion. While possible in principle, this has some major downsides. Obviously, the PLC code would have to be adjusted. This can already be prohibitive for existing production machines for liability reasons. Besides, sampling data at a high frequency puts additional load on the controller. Depending on the application of the machine, e.g., controlling a high number of axes in real time, the computational resources might already be exhausted. Finally, the sampling rate would depend on the PLC's cycle time. Typically, the fastest possible cycle times, depending on model and brand, range from 1 ms to 10 ms, that is, 1 kHz to 0.1 kHz. In practical applications, the cycle time is often deliberately relaxed, e.g., to 10 ms or 20 ms, to avoid stressing the system when the fast cycles are not required. Input modules with oversampling capabilities could make the sampling more independent from the PLC cycle time but require additional hardware. Hence, using the PLC directly is not optimal for our scenario to acquire additional high frequency data of external sensors.

While the resolution of the used vibration sensor proved to be not sufficient for a detailed acoustical analysis, a more sophisticated measurement setup would have produced a better signal and subsequently potentially clearer results for the acoustical features. We did not pursue this further in favor of a measurement- and preprocessing-wise simpler solution. This helps to keep the approach straightforward and the costs low, while also maintaining a sufficient performance.

Although more runs were recorded, only the runs processing real material were used for the experimental results. For the runs where no material was processed, the vibration signals as well as some machine internal data, such as torque and DC link power, contain much less information, resulting in different data patterns. Thus, the learned models were not applicable on these runs.

6. Conclusion

Sudden machine tool failures can lead to downtimes of entire factories and thus to enormous costs. Therefore, the premature detection of machine tool anomalies is an essential task to avoid such downtimes. In this paper, we present a highly generalizable end-to-end workflow for detecting machine tool anomalies that requires only basic machine data (e.g., velocity, position, and vibration signals) and a comparably small training data set. Our approach processes the data in an automated workflow consisting of two main steps: phase detection and anomaly detection. First, the raw machine data is clustered into production and tool change phases and similar phases are matched using hierarchical clustering. Second, supervised classification models are applied to detect present degradations in the phases. By applying a resampling strategy, our approach requires only small training data sets, which is a crucial requirement for a successful application in the industrial context. We evaluate our workflow using a real-world data set and apply seven machine learning models to identify anomalies. The results show that the phase detection is very accurate. In addition, resampling the vibration signal improves the performance of all machine learning methods, whereby random forest achieves the most robust high values for the evaluation metrics, while a single decision tree even surpasses random forest for a high rotational speed.

As future work, the simulation of further anomalies in different states of degradation are planned to assess the performance and sensitivity of the proposed automated workflow in other scenarios. Furthermore, the transfer of the approach to other domains is also considered. In addition, we plan to integrate time series predictions from our earlier work [36] to predict the characteristics of the next production iterations and provide this information to the machine learning methods in order to not only detect the anomalies, but also to predict them at an even earlier stage.

Acknowledgments

Acknowledgements will only be added in the final version due to the double-blind review process.

References

[1] Vogel-Heuser B, Hess D, Guest editorial industry 4.0–prerequisites and visions, IEEE Trans on Automation Sci and Engin 13 (2) (2016) 411–413.

[2] Bosch Rexroth, Simply collecting data is not enough, accessed on 05.03.2020.
URL `https://www.boschrexroth.com/en/xc/company/press/index2-23360`

[3] World Economic Forum, Industrial Internet of Things: Unleashing the Potential of Connected Products and Services, Tech. rep. (January 2015).

[4] Mobley RK, An Introduction to Predictive Maintenance, Elsevier, 2002.

[5] Krupitzer C, Wagenhals T, Züfle M, Lesch V, Schäfer D, Mozaffarin A, Edinger J, Becker C, Kounev S, A survey on predictive maintenance for industry 4.0 (2020). `arXiv:2002.08224`.

[6] Gebraeel N, Sensory-Updated Residual Life Distributions for Components With Exponential Degradation Patterns, IEEE Trans Automation Sci Eng 3 (4) (2006) 382–393.

[7] Liao W, Wang Y, Pan E, Single-machine-based predictive maintenance model considering intelligent machinery prognostics, Internat Jrnl Adv Manuf Technol 63 (1-4) (2012) 51–63.

[8] Cai H, Jia X, Feng J, Li W, Pahren L, Lee J, A similarity based methodology for machine prognostics by using kernel two sample test, ISA Tans (2020).

[9] Hoang DT, Kang HJ, A survey on Deep Learning based bearing fault diagnosis, Neurocomputing 335 (2019) 327–335.

[10] Hoang DT, Kang HJ, Rolling element bearing fault diagnosis using convolutional neural network and vibration image, Cognitive Syst Res 53 (2019) 42–50.

[11] Han T, Liu C, Yang W, Jiang D, Learning transferable features in deep convolutional neural networks for diagnosing unseen machine conditions, ISA Trans 93 (2019) 341–353.

[12] Sobie C, Freitas C, Nicolai M, Simulation-driven machine learning: Bearing fault classification, Mech Syst and Signal Processing 99 (2018) 403–419.

[13] Yam R, Tse P, Li L, Tu P, Intelligent Predictive Decision Support System for Condition-Based Maintenance, Internat Jrnl Adv Manuf Technol 17 (5) (2001) 383–391.

[14] Haidong S, Ziyang D, Junsheng C, Hongkai J, Intelligent fault diagnosis among different rotating machines using novel stacked transfer auto-encoder optimized by PSO, ISA Trans (2020).

[15] Shah D, Wang J, He QP, Feature Engineering in Big Data Analytics for IoT-Enabled Smart Manufacturing–Comparison between Deep Learning and Statistical Learning, Computers & Chem Engin (2020) 106970.

[16] Knittel D, Makich H, Nouari M, Milling diagnosis using artificial intelligence approaches, Mech & Ind 20 (8) (2019) 809.

[17] You MY, Meng G, A modularized framework for predictive maintenance scheduling, Proc Inst Mech Engineers, Part O: Jrnl Risk Reliability 226 (4) (2012) 380–391.

[18] Ladj A, Varnier C, Tayeb FBS, IPro-GA: an integrated prognostic based GA for scheduling jobs and predictive maintenance in a single multifunctional machine, IFAC-PapersOnLine 49 (12) (2016) 1821–1826.

[19] Yang ZM, Djurdjanovic D, Ni J, Maintenance scheduling in manufacturing systems based on predicted machine degradation, Jrnl Intell Manuf 19 (1) (2008) 87–98.

[20] Liao W, Wang Y, Data-driven Machinery Prognostics Approach using in a Predictive Maintenance Model, Jrnl Computers 8 (1) (2013) 225–231.

[21] Lloyd S, Least Squares Quantization in PCM, IEEE Tans Information Theory 28 (2) (1982) 129–137.

[22] Rokach L, Maimon O, Clustering Methods, in: Data Mining and Knowledge Discovery Handbook, Springer, 2005, pp. 321–352.

[23] Uchtmann K, Wirth R, Maschinendiagnose an drehzahlveränderlichen Antrieben mittels Ordnungsanalyse, Antriebstechnik 38 (5) (1999) 44–49, [Available in German only].

[24] Fix E, Discriminatory analysis: nonparametric discrimination, consistency properties, USAF school of Aviation Medicine, 1951.

[25] Breiman L, Random Forests, Mach Learning 45 (1) (2001) 5–32.

[26] Cortes C, Vapnik V, Support-Vector Networks, Mach learning 20 (3) (1995) 273–297.

[27] Chen T, Guestrin C, XGBoost: A Scalable Tree Boosting System, in: KDD'16, ACM, 2016, pp. 785–794.

[28] Welch P, The Use of Fast Fourier Transform for the Estimation of Power Spectra: A Method Based on Time Averaging Over Short, Modified Periodograms, IEEE Trans Audio and Electroacoustics 15 (2) (1967) 70–73.

[29] Wirth R, Maschinendiagnose an Industriegetrieben Teil II: Signalidentifikation in der Praxis, Antriebstechnik 37 (11) (1998) 77–81, [Available in German only].

[30] Matthews BW, Comparison of the predicted and observed secondary structure of T4 phage lysozyme, Biochimica et Biophysica Acta (BBA)-Protein Structure 405 (2) (1975) 442–451.

[31] Meyer D, Dimitriadou E, Hornik K, Weingessel A, Leisch F, Chang CC, Lin CC, Misc Functions of the Department of Statistics, Probability Theory Group (2019).
URL `https://cran.r-project.org/web//packages/e1071/e1071.pdf`

[32] Ripley B, Classification and Regression Trees (2019).
URL `https://cran.r-project.org/web/packages/tree/tree.pdf`

[33] Ripley B, Venables W, Functions for Classification (2021).
URL `https://cran.r-project.org/web/packages/class/class.pdf`

[34] Breiman L, Cutler A, Liaw A, Wiener M, Breiman and Cutler's Random Forests for Classification and Regression (2018).
URL `https://cran.r-project.org/web/packages/randomForest/randomForest.pdf`

[35] Chen T, He T, Extreme Gradient Boosting (2019).
URL `https://cran.r-project.org/web/packages/xgboost/xgboost.pdf`

[36] Züfle M, Bauer A, Lesch V, Krupitzer C, Herbst N, Kounev S, Curtef V, Autonomic Forecasting Method Selection: Examination and Ways Ahead, in: Proceedings of the 16th IEEE International Conference on Autonomic Computing (ICAC), IEEE, 2019.