Random Forest on an Embedded Device for Real-time Machine State Classification

Fabian Küppers
Ruhr West University of Applied Sciences
Bottrop, Germany
fabian.kueppers@hs-ruhrwest.de

Jonas Albers
Lenord, Bauer & Co. GmbH
Oberhausen, Germany
jalbers@lenord.de

Anselm Haselhoff
Ruhr West University of Applied Sciences
Bottrop, Germany
anselm.haselhoff@hs-ruhrwest.de

***Abstract*—Heavy machine tools are used in numerous industrial manufacturing processes. Avoiding unplanned maintenance time is crucial and can be achieved by continuous condition monitoring. A more targeted condition monitoring is possible if the operating state of a machine tool is known (e.g. *standstill*, *neutral* or *cut*). However, many manufacturers of control units do not release any information about the machine's operating state. For this reason, we investigated a system-independent approach to determine the operating state in real-time in less than 1 ms. This low runtime is necessary for machine state classification. Since most machine tools vary in individual components, the proposed state detection uses learning algorithms based on the machine's vibration characteristics.**

In this work we propose the Random Forest algorithm because of its reliable classification performance while keeping explainability of each prediction. Facing the real-time requirements, the Random Forest model has been adapted to an embedded device with very limited resources. Thus, the model prediction has to work with low resource consumption in a very low runtime and without loss in accuracy. We show that this approach is suitable for determining machine operating states in real-time in less than 700 $\mu\mathrm{s}$ with an average accuracy of 96 %.

***Index Terms*—Random Forest, Machine Tool, Classification, State Estimation, Real-Time**

I. Introduction

INdustrial manufacturing processes are highly optimized and automated work flows for the production of various goods. In particular, the production lines of automobile manufacturers are precisely matched to the individual work steps. Heavy machine tools are used in these manufacturing processes e.g. to mill an engine block from a single piece of metal.

Unplanned maintenance is costly due to high supply chain integration and must be kept to a minimum. This can be achieved by continuous condition monitoring. The target is to make reliable statements about the current machine condition in order to plan the next maintenance periods.

However, for targeted condition monitoring it is very important to know the current operating state of a machine. For example, an industrial milling machine might have various operating states such as *standstill* or *tool change*, *rapid traverse with rotating shaft* (denoted by *neutral*) and *incision in work piece* (*cut*). When determining these states, however, it is rarely possible to get these information by the machine's control unit since many manufacturers of control units do not release any information about the machine's operating state.

We investigate how the machine's operating state can be determined in real-time using the vibration signal of a sensor mounted directly above the rotating shaft of a milling spindle. This low runtime is necessary for targeted condition monitoring. The rotation signal is also taken into account. Since most machine tools, especially in larger production plants, vary in individual components, different machines have different vibration characteristics. For this reason, we considered machine learning algorithms to learn a vibration model for a particular milling machine. In order to meet the real-time requirements, such a model must be able to perform all decisions in less than 1 ms on an embedded device with very limited resources. We used the Random Forest algorithm based on Decision Trees [1] to determine the machine state because of its reliable classification performance while keeping explainability of each prediction. However, interpreting machine learning models is a wide area of research. Thus, we did not focus on explaining model decisions in this work. For classification, we identified and implemented appropriate features based on the vibration signal. Afterwards, we implemented and examined the Random Forest algorithm to run properly and fault-tolerant on an embedded device with limited memory and computational resources in real-time.

II. Related Work

Machine state classification is a wide area of research and used in various domains [2]–[6]. There are different approaches to perform adaptive condition monitoring on industrial milling machines [7], [8]. For example, Widodo & Yang [9] examine different machine learning approaches and use Support Vector Machines (SVM) [10] for condition monitoring and fault diagnosis. Prieto et al. [11] use a neural network to detect bearing degradation relying on time-based vibration features. Our approach does not describe condition monitoring itself, but rather a more targeted detection of the machine state so that condition monitoring might be carried out much more precisely with the aid of specialized machine learning models.

In recent years, Neural Networks and Convolutional Neural Networks (CNN) in particular enjoy growing popularity. Both Ince et al. [12] and Abdeljaber et al. [5] use one-dimensional CNNs for motor fault detection. The data stream of the vibration signal is processed in time-based domain directly. A major disadvantage of conventional Neural Networks, CNNs

978-9-0827-9703-9/19/$31.00

and other black box methods like SVMs is their explainability. Given an input signal X a prediction Y is generated. However, it is not clear why the model comes to its result. This topic has already been subject of research in the past with rule-based Neural Networks [13]. Nevertheless, these metrics are unsuitable for reliable machine state prediction due to the much more complex implementation and computation effort.

We propose the Random Forest algorithm [1] to perform state recognition. A Random Forest consists of an ensemble of Decision Trees [14]. In the field of state detection and condition monitoring of wind turbines, the work of Kusiak et al. [3] already shows that the Random Forest algorithm offers high accuracy compared with other machine learning algorithms. For simple use cases, Neural Networks and Decision Trees offer nearly the same performance in accuracy. However, Decision Trees are much easier to explain and debug [15] by simply examining the "decision path" computed for a single prediction. For an ensemble method like Random Forest, the feature contributions can be computed to determine the influence of each variable [16].

The original Decision Tree and thus the Random Forest algorithm are binary classifiers. However, the application of classifying machine states needs more different states than two and thus leads to a multi class classification problem. This is addressed in a one vs. all manner. The forest for each class either accepts or declines its class by a majority voting [17].

The Random Forest model must fit to the real-time requirements. In a time frame of max. $1\ \mathrm{ms}$ (from arrival of the first sample) the model should make a decision about the current machine state. In general, it is possible to process data in real-time with Random Forests [18], [19].

III. Random Forest for real-time State Estimation

A Random Forest consists of an ensemble of multiple uncorrelated Decision Trees. A Decision Tree is a non-metric learning model which creates rules during training to map the input to a specific classification result. The Random Forest predicts a class y given feature vector $\vec{x} = (x_1, ..., x_K)^T$ with K features by the highest class vote of its trees. Each tree is trained by a subset D_s that is randomly sampled with replacement from the whole training set $D = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), ...\}$ with $D_s \subseteq D$. Each subset is sampled so that the amount of classes are balanced. The greater the subsets for each Decision Tree the higher the representational power of each model gets. However, large training subsets result in a high training duration. Thus, the size of each subset is part of our examinations.

For continuous features, the nodes in a Decision Tree divide the input space by a given feature $x_k \in \vec{x}$ into two subsets which are either greater or lower than a specific threshold. The threshold as well as the index k of the feature for a node are determined by the information gain IG [20], [21] with

$$\underset{k}{\operatorname{argmax}} \quad IG(D_s, k). \tag{1}$$

The information gain is the criterion for building up all branches in a tree until either the maximum tree depth is reached or no more information gain can be achieved. In these two cases, a leaf is created at the end of the branch. This leaf contains the class that occurs most frequently in the remaining subset at the end of the branch. We use Reduced Error Pruning (REP) in order to avoid overfitting [14], [22]. Starting at the bottom of a tree, each node is replaced with the majority class vote of subsequent leaves. If the prediction accuracy on a dedicated test set is not affected then the change is kept. This training method is useful for setting up a Decision Tree which is able to distinguish between two classes $+1$ and -1. For multiple classes, the one vs. all method is applied. The used Random Forest model is shown in Fig. 1.

Features

The state detection bases on the vibration signal of the machine. The vibration signal is sampled at a frequency f so that f samples (denoted by s) are recorded per second. However, the Random Forest algorithm needs dedicated features in order to determine the machine state. Thus, all samples s are windowed in consecutive time slots t with a window length n (denoted by $\vec{s}_t$). For each time slot t, a feature vector $\vec{x}_t$ is computed. This is demonstrated in Fig. 2.

The vibration signal is expected to change depending on the operating state of the milling machine (*standstill*, *neutral*, *cut*). We observed homogeneous and uniform vibrations with states *standstill* and *neutral* and irregular deflections when cutting into a work piece. These vibration characteristics can be measured with the polygon length L of the signal in a time window with

$$L(\vec{s}_t) = \sum_{i=1}^{n-1} \sqrt{(s_i - s_{i+1})^2 + 1} \tag{2}$$

and the amount of sign changes related to a zero level s_0 with

$$\mathrm{numsgn}(\vec{s}_t) = \sum_{i=1}^{n-1} \begin{cases} 1 & \text{if } (s_i - s_0)(s_{i+1} - s_0) < 0 \\ 0 & \text{else} \end{cases}. \tag{3}$$

We expect these features to vary most between different machine states. The measured engine rotation speed (denoted as rpm) as well as the mean μ and variance σ^2 of the vibration signal are also taken into account and added to the feature vector $\vec{x}_t$

$$\vec{x}_t = (\mathrm{rpm}, \quad \mu, \quad \sigma^2, \quad L, \quad \mathrm{numsgn})^T. \tag{4}$$

For computing the polygon length L of a signal, it is necessary to perform a square root operation for each consecutive samples (cf. (2)). This causes $n - 1$ square root operations for a sample window of size n. Facing the real-time requirements, a floating point square root on an ARM® Cortex®-M4 with a dedicated Floating Point Unit (FPU) e.g. needs 14 instruction cycles while an addition operation only needs 1 cycle. On other devices without a dedicated FPU, the amount of cycles needed for a square root operation (integer or float) is considerably larger. Due to the high computational

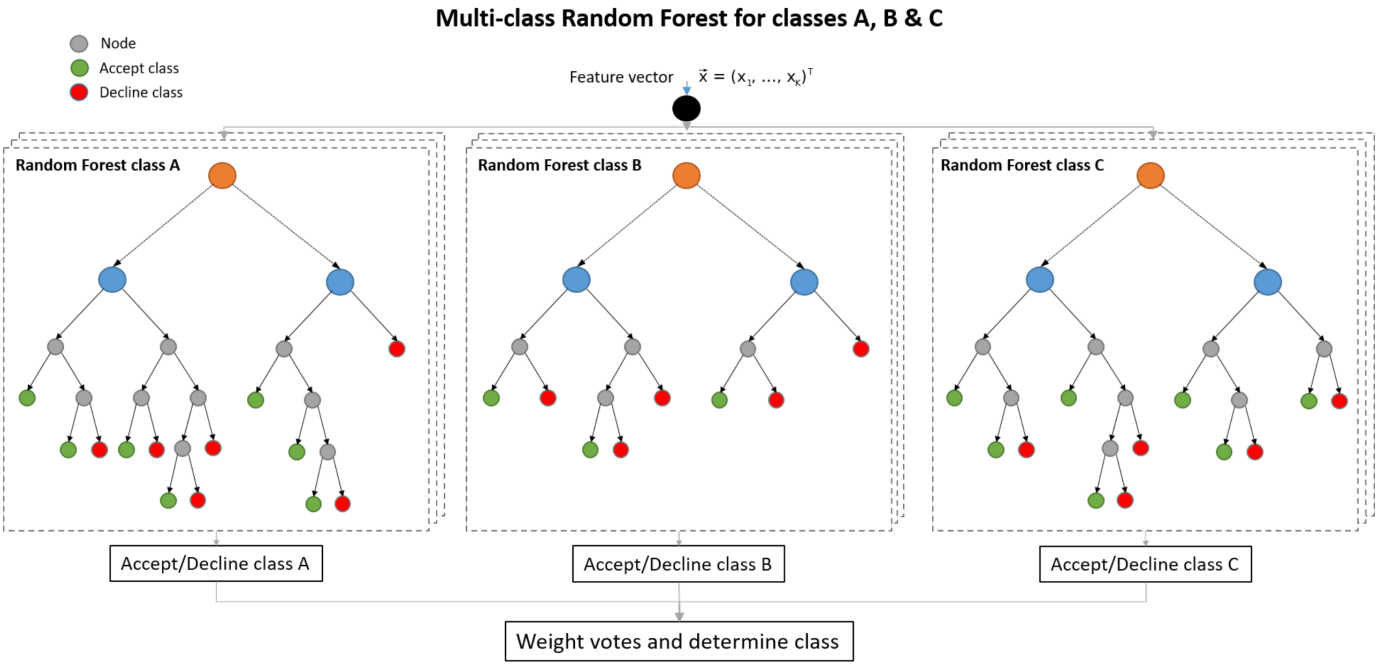

Fig. 1. Random Forest model to distinguish between multiple classes by a given feature vector $\vec{x}$. Each forest either accepts or declines the class it was trained for. The class with the majority of votes is accepted.

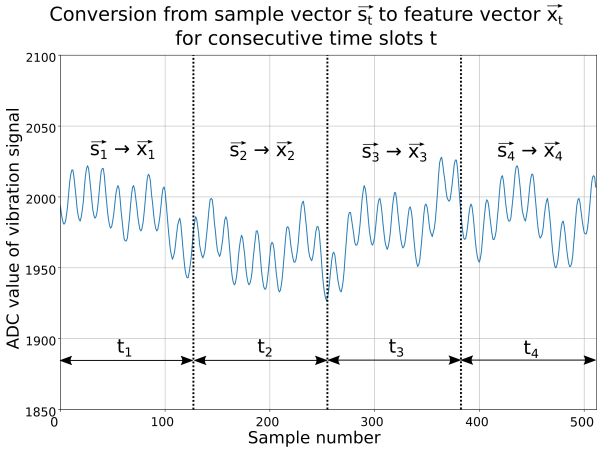

Fig. 2. Vibration signal sampled by an Analog-Digital Converter (ADC). The signal is separated into consecutive time slots of size n. For each time slot t, a feature vector $\vec{x}_t$ is computed.

costs of such an operation, the used polygon length L for $\vec{x}_t$ is approximated for large differences $s' = |s_i - s_{i+1}|$ by

$$\lim_{s' \gg 1} L(\vec{s}_t) = \sum_{i}^{n-1} |s_i - s_{i+1}|. \tag{5}$$

Features in the frequency domain like Power Spectral Density (PSD) computed by a Fast Fourier Transformation (FFT) could also be taken into account. FFT algorithms have a very low runtime and are suitable for real-time applications. For windowed data, a Short Time FFT (SFFT) or a Wavelet Transformation (e.g. Gabor-Wavelet) are common concepts to get frequency features while preserving time information. However, internal investigations based on a Principal Component Analysis (PCA) have revealed no additional information content of these frequency features compared to the previously mentioned time based features.

IV. Empirical Evaluation

A. Data Recording and Hyperparameter Setup

We performed several vibration measurements of a milling spindle and classified them by their states (*standstill*, *neutral*, *cut*) to get a training data set. State transitions or the combination of different machine configurations are not considered. The recorded data is considered to be a homogeneous and equally distributed set of these different states to represent real operation mode. For this purpose, measurements with rotation speeds at 0 (*standstill* only) and 5,000 rpm have been recorded. The sampling frequency was 1 MSample/s (1,000,000 samples per second). The whole setup is shown in Table I and Fig. 3.

To get the best hyperparameters (model configuration parameters) for the model, the amount of trees for each class, the size of training subsets D_s for each Decision Tree and the window size n are varied and examined by their influence on the resulting prediction accuracy. On the basis of these parameters, we performed a Grid Search hyperparameter optimization (training and evaluation of several predefined combinations of hyperparameters) and a 5x2 cross validation for each configuration setup.

We measure a correlation of 0.81 between the size of training subsets D_s and prediction accuracy (with a p-value

TABLE I
EXPERIMENTAL SETUP OF MICROPROCESSOR AND RANDOM FOREST CONFIGURATION.

Device	STM32F207 Nucleo-144
Processor	ARM® Cortex®-M3 @ 120 MHz
Power supply	5V with max. 300 mA
A/D converter	3 SAR A/D @ 15 MHz
A/D resolution	12 bit
A/D record	3 record cycles with DMA Double Buffer Mode enabled
Vibration sensor	ADXL 1002 MEMS @ 3.3V
Measurement card	NI PC 6115 @ $\pm$10V
Rotation speed	5,000 rpm
Size of D_s	750 samples
Amount of trees	9 per class (27 total)
Window size n	512

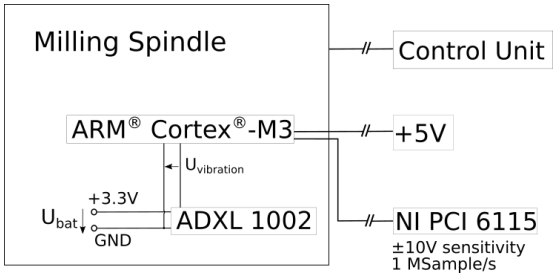

Fig. 3. Setup with milling spindle, a microprocessor to detect machine states and related measurement devices.

of > 0.01 %). This offers a significant and strong positive correlation and thus leads to the assumption that the overall prediction accuracy highly depends on the amount of training data. The amount of trees per class and the window size n have a low correlation of 0.34 and 0.28 to prediction accuracy (with p-values of 1 % and 3 %) respectively. Their impact is not as important as the size of each training subset. In other terms, the used Random Forest model does neither need a huge amount of trees nor a big (and thus large) window size n to get a fairly good trade-off between prediction accuracy and size and thus runtime.

B. Accuracy & Runtime on the Embedded Device

Based on the hyperparameter investigations, we trained a Random Forest model with 9 Decision Trees per class, a window size n of 512 and a training subset size of 750 samples per tree. This model was flashed onto an embedded device (cf. Table I) and should determine the states of a milling machine during operation in real-time. For each machine state, we configured a digital output of the microcontroller and connected it to the NI PC 6115 measuring card in order to observe the currently predicted state. The vibration signal was recorded with the first Analog-Digital Converter (ADC) connected to the board's DMA with double buffer mode enabled (recording and computing data simultaneously with two separated data buffers). The examinations are splitted into runtime and performance measurements in the following.

1) Runtime examinations: The ADC is driven at 15 MHz sampling frequency with 12 bit resolution and 3 cycles per sample. At a window size of 512, this leads to a theoretical sampling time of $t_{\text{ADC}} = 512\,\mu$s. The DMA overhead is not included. Because of the DMA's double buffer mode, it is possible to record the signal while processing the previous window. The feature calculations and the state classification must be finished before the new window is recorded completely. Otherwise, the feature calculations are inconsistent. Therefore, the algorithm has to be in compliance with the following constraints:

$$t_{\text{total}} = t_{\text{ADC}} + t_{\text{features}} + t_{\text{classification}} \tag{6}$$

$$\text{subject to} \quad t_{\text{features}} + t_{\text{classification}} \leq t_{\text{ADC}}. \tag{7}$$

To measure the required conversion time of the ADC for a single time window, a trigger pulse is applied to an output of the microcontroller after each recorded time window. With this trigger, we measured a frequency of 0.97 kHz for sampling time t_{ADC}. One period thereby corresponds to two recorded time frames. Thus, the real sampling time including overhead is given by $t_{\text{ADC}} = 515.46\,\mu$s, which is equivalent to a sampling frequency of approx. 1 MSample/s.

As a next step, we repeated this procedure to determine the required time for calculating the features of a time frame. This was measured with ADC disabled. The evaluation of the trigger for computing the features has shown a frequency of 3.15 kHz. Therefore, the feature calculation time is given by $t_{\text{features}} = 158.73\,\mu$s, while the trigger impulse for the pure classification time oscillates at 20.9 kHz. The classification duration is thus given by $t_{\text{classification}} = 23.92\,\mu$s, so that $t_{\text{features}} + t_{\text{classification}} = 182.65\,\mu$s , which fulfills the restriction of (7). The total time from the start of the time frame sampling until the state decision is therefore given by $t_{\text{total}} = 698.11\,\mu$s. After classification, the processor waits approx. 332 μs for a new time frame. This leads to 1,940 model decisions per second and also allows larger Random Forest models with a higher amount of Decision Trees to reduce the CPU idle time.

TABLE II
CONFUSION MATRIX AND ACCURACY OF EACH STATE.

expected: / actual:	standstill	neutral	cut
amount of time frames	37,204	43,078	41,635
standstill	37,204	0	0
neutral	0	43,044	4,951
cut	0	34	36,680
unknown	0	0	4
accuracy	1.0	0.999	0.881

2) Accuracy: We have recorded a test data set with the previously described setup (cf. Fig. 3 and Table I) and measured the accuracy of the Random Forest model in online operation mode The detailed classification and accuracy results are presented as a confusion matrix in Table II (predicted states over expected states). The average accuracy is given with 96 %. Nearby none of the recorded samples are classified as unknown. While the detection of the states *standstill* and *neutral* have a very high accuracy, approx. 11.9 % of the recorded *cut* samples are still classified as *neutral*.

By examining the recorded classification trigger it is remarkable that most of the misclassified states are not clustered but equally distributed over the recording time. Thus, it might be advantageous to issue a state transition only after more than one consecutive model predictions. We performed new measurements with this method and achieved an average accuracy of 99.8 % with the drawback of a higher runtime of $1.214\,\mathrm{ms}$ which however violates the real-time constraints. The amount of misclassified *cuts* is mainly lower with only 0.57 % while none of the time frames were rejected as unknown.

In summary, the prediction of a machine state could be computed in less than 700 μs. This is significantly lower than the restriction of max. $1\,\mathrm{ms}$. We achieved a mean accuracy of 96 % which is sufficient for this application.

V. Conclusion

It is important to know the operating state of machine tools for targeted condition monitoring. In this work, determining the current state should be done for milling machines based on their rotation and vibration signals in real-time in less than $1\,\mathrm{ms}$. However, each machine has its own characteristics due to design, size and miscellaneous features. Therefore, we trained a Random Forest model onto a milling machine to perform the operating state detection. The major benefits of this algorithm are its explainability of decisions for each underlying Decision Tree, fast computations and ease of implementation.

For this purpose, we identified and examined several features of the vibration input signal due to runtime and information content. We identified the Random Forest method as a promising approach and modified it due to the requirements of a low runtime and resources consumption.

The examinations have shown a prediction runtime of below 700 μs with an average accuracy of 96 % on a STM32F207 Nucleo-144 with ARM® Cortex®-M3 at 120 MHz and a sampling frequency of 1 MSample/s. It was possible to improve the average accuracy up to over 98 % with the drawback of a higher runtime of $1.2\,\mathrm{ms}$. These results show that our approach is convenient for a quick and precise machine state detection even at safety-critical processes.

VI. Outlook

We use a Random Forest in a one vs. all fashion. It is also possible to directly use Decision Trees that are able to perform a multi-class classification. This might enhance prediction speed because less trees are necessary. Our approach of using a Random Forest in a one vs. all fashion might be used to identify critical machine states. If each forest declines the state it was trained for, commanding an emergency shutdown should be possible. However, this scenario was not tested in order not to damage the experimental setup. Thus, this aspect has to be investigated in further examinations. Another interesting approach would be this Random Forest method in combination with the Isolation Forest algorithm. An Isolation Forest is a modification of the Random Forest algorithm for anomaly detection [23]. The combination of these methods might be used to identify critical states in a reliable manner.

References

[1] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, October 2001.

[2] S. Abbasion, A. Rafsanjani, A. Farshidianfar, and N. Irani, "Rolling element bearings multi-fault classification based on the wavelet denoising and support vector machine," *Mechanical Systems and Signal Processing*, vol. 21, no. 7, pp. 2933 – 2945, 2007.

[3] A. Kusiak and A. Verma, "A data-mining approach to monitoring wind turbines," *IEEE Transactions on Sustainable Energy*, vol. 3, pp. 150–157, January 2012.

[4] M. Zhao, X. Jin, Z. Zhang, and B. Li, "Fault diagnosis of rolling element bearings via discriminative subspace learning: Visualization and classification," *Expert Systems with Applications*, 2014.

[5] O. Abdeljaber, O. Avci, S. Kiranyaz, M. Gabbouj, and D. J. Inman, "Real-time vibration-based structural damage detection using one-dimensional convolutional neural networks," *Journal of Sound and Vibration*, vol. 388, pp. 154 – 170, 2017.

[6] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao, "Deep learning and its applications to machine health monitoring," *Mechanical Systems and Signal Processing*, 2019.

[7] P. Prickett and C. Johns, "An overview of approaches to end milling tool monitoring," *International Journal of Machine Tools and Manufacture*, vol. 39, no. 1, pp. 105 – 122, 1999.

[8] B. Kilundu, P. Dehombreux, and X. Chiementin, "Tool wear monitoring by machine learning techniques and singular spectrum analysis," *Mechanical Systems and Signal Processing*, vol. 25, pp. 400 – 415, 2011.

[9] A. Widodo and B.-S. Yang, "Support vector machine in machine condition monitoring and fault diagnosis," *Mechanical Systems and Signal Processing*, vol. 21, no. 6, pp. 2560 – 2574, 2007.

[10] V. N. Vapnik and A. J. Chervonenkis, *Theory of Pattern Recognition*. Moscow: Nauka, 1974.

[11] M. D. Prieto, G. Cirrincione, A. G. Espinosa, J. A. Ortega, and H. Henao, "Bearing fault detection by a novel condition-monitoring scheme based on statistical-time features and neural networks," *IEEE Transactions on Industrial Electronics*, vol. 60, pp. 3398–3407, August 2013.

[12] T. Ince, S. Kiranyaz, L. Eren, M. Askar, and M. Gabbouj, "Real-time motor fault detection by 1-d convolutional neural networks," *IEEE Transactions on Industrial Electronics*, vol. 63, pp. 7067–7075, November 2016.

[13] R. M. Goodman, C. M. Higgins, J. W. Miller, and P. Smyth, "Rule-based neural networks for classification and probability estimation," *Neural Computation*, vol. 4, no. 6, pp. 781–804, 1992.

[14] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, vol. 19. Wadsworth and Brooks, 1984.

[15] I. S. Markham, R. G. Mathieu, and B. A. Wray, "Kanban setting through artificial intelligence: a comparative study of artificial neural networks and decision trees," *Integrated Manufacturing Systems*, vol. 11, no. 4, pp. 239–246, 2000.

[16] A. Palczewska, J. Palczewski, R. M. Robinson, and D. Neagu, "Interpreting random forest models using a feature contribution method," in *2013 IEEE 14th International Conference on Information Reuse & Integration (IRI)*, pp. 112–119, IEEE, 2013.

[17] M. N. Adnan and M. Z. Islam, "One-vs-all binarization technique in the context of random forest." European Symposium of Artificial Neural Networks, Computational Intelligence and Machine Learning, Bruges, Belgium, April 2015. Conference Paper.

[18] V. Lempitsky, M. Verhoek, J. A. Noble, and A. Blake, "Random forest classification for automatic delineation of myocardium in real-time 3d echocardiography," in *Functional Imaging and Modeling of the Heart*, pp. 447–456, 2009.

[19] G. Fanelli, J. Gall, and L. V. Gool, "Real time head pose estimation with random regression forests," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2011*, pp. 617–624, June 2011.

[20] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81–106, March 1986.

[21] T. M. Mitchell, *Machine Learning*. New York, NY, USA: McGraw-Hill, Inc., 1 ed., 1997.

[22] J. R. Quinlan, "Simplifying decision trees," *International journal of man-machine studies*, vol. 27, no. 3, pp. 221–234, 1987.

[23] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *ACM Trans. Knowl. Discov. Data*, March 2012.