

ANOMALY DETECTION IN THE TEMPERATURE OF AN AC MOTOR USING EMBEDDED MACHINE LEARNING

Ezzeldin Ayman Ibrahim Ismail, Mohd Ridzuan Ahmad*

Division of Control and Mechatronics Engineering, Faculty of Electrical Engineering, Universiti Teknologi Malaysia, 81310 UTM Johor Bahru, Johor, Malaysia

Article history

Received

25 October 2022

Received in revised form

29 May 2023

Accepted

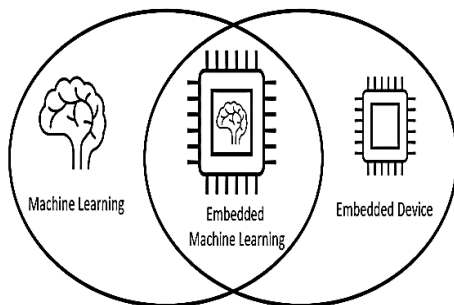
10 July 2023

Published Online

20 October 2023

*Corresponding author
mdridzuan@utm.my

Graphical abstract



Abstract

The integration of machine learning solutions is becoming more prominent in the industry. In industrial maintenance, new approaches categorized under predictive maintenance primarily use machine learning to identify patterns that could lead to machine failures. However, in most cases, implementing a machine learning approach is very expensive regarding resources and experienced personnel. Therefore, this approach is usually more costly in some machines than replacing these faulty machines instead. This paper proposes a low-cost machine-learning approach to detect anomalies in a rotary machine by monitoring its casing temperature using Edgelpulse to Train the model and a Raspberry Pico as the microcontroller. The project is divided into two phases. Data is collected to be used to train and test the model. The model is then deployed to the microcontroller and is connected to a sensor attached to the motor. The model developed showed promising results with an accuracy of 91% and a $f1$ score of 0.91.

Keywords: Machine learning, maintenance, anomaly detection, rotary machine, microcontroller

Abstrak

Penyepaduan penyelesaian pembelajaran mesin menjadi lebih menonjol dalam industri. Dalam penyelenggaraan industri, pendekatan baharu yang dikategorikan di bawah penyelenggaraan ramalan terutamanya menggunakan pembelajaran mesin untuk mengenal pasti corak yang boleh membawa kepada kegagalan mesin. Walau bagaimanapun, dalam kebanyakan kes, melaksanakan pendekatan pembelajaran mesin adalah sangat mahal berkaitan sumber dan kakitangan yang berpengalaman. Oleh itu, pendekatan ini biasanya lebih mahal dalam sesetengah mesin daripada menggantikan mesin yang rosak ini. Kertas kerja ini mencadangkan pendekatan pembelajaran mesin kos rendah untuk mengesan anomali dalam mesin berputar dengan memantau suhu selongsongnya menggunakan Edgelpulse untuk Melatih model dan Raspberry Pico sebagai mikropengawal. Projek ini dibahagikan kepada dua fasa. Data dikumpul untuk digunakan untuk melatih dan menguji model. Model itu kemudiannya digunakan pada mikropengawal dan akan disambungkan kepada penerima yang dipasang pada motor. Model yang dibangunkan menunjukkan hasil yang menjanjikan dengan ketepatan 91% dan skor $f1$ 0.91.

Kata kunci: Pembelajaran mesin, penyelenggaraan, pengesanan anomali, mesin berputar, pengawal mikro

© 2023 Penerbit UTM Press. All rights reserved

1.0 INTRODUCTION

Maintenance strategies will need to be modified as the industry transitions toward Industry 4.0 [1]. Currently implemented maintenance procedures have different types of problems. One of the most common maintenance strategies is corrective maintenance. Curative maintenance is simply the repair or replacement of defective or damaged equipment [2]. Although it is a simple approach, it has several flaws. Financially speaking, unplanned downtime may result in tremendous losses, even for a few hours. Furthermore, regarding energy consumption, a worn-but functional machine that is expected to malfunction or be damaged would consume more energy than a machine working normally [3].

Another type of maintenance strategy is preventive maintenance. Unlike corrective maintenance, this approach reduces unexpected downtimes of machines due to failures by using scheduled maintenance. By using statistical data, machine parts are replaced periodically. The advantage of using this approach is that the machine will have fewer unexpected breakdowns. However, the problem with this maintenance strategy is that it is not cost-effective. When parts are replaced after a specific period, the health of that part is not considered or assessed. Therefore, sometimes a perfectly working part may be replaced because of its scheduled maintenance due date, which wastes the remaining useful life of the part [4, 5].

The disadvantages of preventive maintenance mentioned above are why the industry is moving towards predictive maintenance [4]. Predictive maintenance revolves around the use of monitoring devices to observe system parameters related to the health status of the machine or parameters that, if changed, could negatively affect the machine's performance. The change in the data sent by monitoring devices can be called deviations or anomalies. A predictive maintenance system detects these anomalies before they cause machine failures. After that, maintenance can be performed to find the leading cause of the anomalies and fix them [6].

However, the conventional approach to predictive maintenance uses cloud computing to process the data. Nevertheless, using cloud computing servers and IoT devices in anomaly detection machine learning models causes large amounts of data transmission. This leads to high power consumption, bandwidth requirements, and low latency [4, 7]. Edge computing was introduced to solve many problems related to using cloud servers in data transmission. This is because computation at the edge has a better response time, less energy consumption, and is more secure in transferring data [8, 9]. Microcontrollers can be used to implement edge computation. Although there are few papers related to the performance of machine learning algorithms implemented on edge devices (microcontrollers), microcontrollers (μ C) have shown

promising results due to their fast response and energy efficiency [10-13].

The predictive maintenance that is focused upon is the one that uses a machine learning model to detect the anomalies of the machine. The first step is to collect data about the parameter that will be monitored. After collecting enough data, a feature selection process is performed to try and find patterns within the dataset gathered. After that, a machine learning algorithm is chosen based on the patterns found within the dataset, and then the model is trained using the dataset and then deployed to be ready to work.

An ML model could be deployed on three types of platforms. Cloud, Mobile, and TinyML. Cloud ML is the conventional approach when it comes to predictive maintenance plans. However, in this project, the TinyML approach is used. It is essential to understand the difference between Cloud ML and TinyML. Cloud ML uses cloud computing, where the ML model is deployed to. Data is transmitted through the internet since the sensor is planted on the machine or the asset to be monitored. On the other hand, TinyML uses a μ C for the ML model to be deployed. By connecting the sensor to μ C, the data is sent locally within the μ C itself [11].

TinyML was introduced in the first place because of the problems that the conventional ML -Cloud ML-platform faces. These problems were related to energy consumption, internet connectivity, and privacy. Since the ML model is deployed in the cloud, data are sent from the sensor to the model to filter, process, and consume significant amounts of energy. Furthermore, with the increase of industrial technologies that use Cloud computation and IoT devices, transmitting data in real-time from sensors to the cloud would require unnecessary bandwidth allocation. Moreover, latency could occur during the transmission of data. Finally, by sending the data to the cloud, there is a risk of having the data compromised or hacked [8, 11].

On the other hand, TinyML uses a μ C where the ML model is deployed and connected to the sensor locally. Using μ C eliminates the energy consumption problem since these devices tend to consume power within the range of milliwatts. It is well established that internet connectivity is not required by using μ Cs to host the ML model, thus eliminating the connectivity and privacy problems [11, 12].

Anomaly detection generally means identifying data that deviates or shows significant differences compared to standard data within a dataset. In machine learning, anomaly detection is the classification of data as "normal" or "abnormal". 8 Many algorithms can be used to train an ML model to identify anomalies. However, some algorithms could be extremely challenging to use in the TinyML platform because of the high computational operations of these algorithms, like neural networks [11].

Although some solutions show better results, it does not mean they should be implemented in all applications. For example, replacing a machine may

not be as costly as applying predictive maintenance for that machine [14]. For this reason, some rotary machines break down quite significantly due to outdated maintenance plans [15, 16].

In conventional machine learning approaches, it is common to utilize multiple parameters in a single model for anomaly detection. However, in the context of embedded machine learning, resource constraints often limit the usage to only one parameter. In this project, the temperature parameter has been chosen for anomaly detection due to its ability to accurately reflect the motor's operational state. Employing vibrations of the motor as a parameter for anomaly detection can pose challenges since it requires the motor to remain stationary. Even slight changes in displacement can result in inaccurate data collection, making temperature a more suitable and reliable parameter for this purpose.

2.0 METHODOLOGY

The project workflow is outlined and adhered to as depicted in Figure 1. The circuit schematic is designed to detect the motor's casing temperature. After that, the components are connected by following the schematic that has been created. Following that, the code required to read the temperature data from the sensor is developed; although the full version of the code is available, the code could be modified to fill the need of this project. After that, the data collection process starts using the system discussed above.

The data collected is uploaded to Edgimpulse platform to process the data and find features to learn. After that, the machine learning model is trained and tested before downloading it to be further modified to directly receive the data from the sensor without manually feeding the model with data to make predictions. Finally, the full ML model is deployed into the μC to be tested.

While collecting the data to train the model, the Circuit will consist of the μC , the breakout board, and the sensor. However, after deploying the model into the μC , the circuit will consist of the components mentioned above and extra components that will work as an indicator for the model's output. Table 1 provides a comprehensive list of all parts, their respective functions, and their corresponding prices.

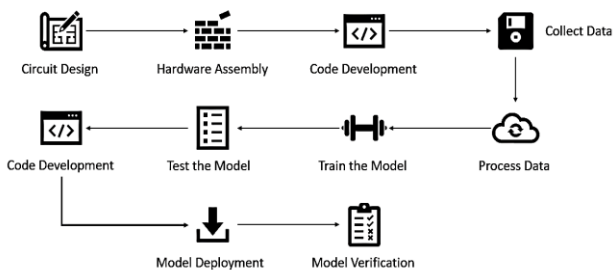








Figure 1 Flow of the project

At the beginning of the project, the Arduino IDE program will connect the μC to the computer and modify and update the code required to read the temperature from the sensor using the breakout board. Once the code preparation is complete, it is uploaded to the μC and subsequently evaluated.

Following this process, the probe of the sensor is placed on the motor's casing, and the data collection process is started. The motor will operate until it reaches its maximum temperature level, which typically takes approximately 20 minutes. Therefore, the collected data is categorized into four distinct datasets based on the motor's operational conditions. Table 2 shows the four dataset labels and their explanation. Figure 2 illustrates the expected graphical representation of the data collected for the four labels.

Table 1 Summary of components used and their functions

Name	Picture	Function
Raspberry Pi Pico		Connects the ML model to the sensor
MAX 31855 Breakout Board		ADC
K-Type Thermocouple Sensor		Analog temperature sensor
LED Lights x3		Will indicate the stress level of the motor by turning on or blinking
Active Buzzer		Will make noise when an anomaly is detected
Liquid Crystal Display		Will display the motor's casing temperature

The data is collected and uploaded to the edge impulse platform using the command prompt and the edge impulse CLI tool. After uploading the data to the edge impulse platform, the data is processed, and features are extracted from the data to train the ML model. The model will mainly be trained on detecting

anomalies. However, a classifier function will also be added to the model to give users more clarity on which stage the motor is running at.

Table 1 Explanation of dataset labels

Label Name	Description	Duration	Temperature Range (Celsius)
Idle	When the Motor is not running	-	30 - 33
Minimum	When the motor is running in minimum temperature range	5 minutes (After Running with one load connected)	33 - 40
Medium	When the motor is running in medium temperature range	10 minutes (with one load connected)	40 - 56
Maximum	When the motor is running in maximum temperature range	5 minutes (with one load connected)	56 - 65

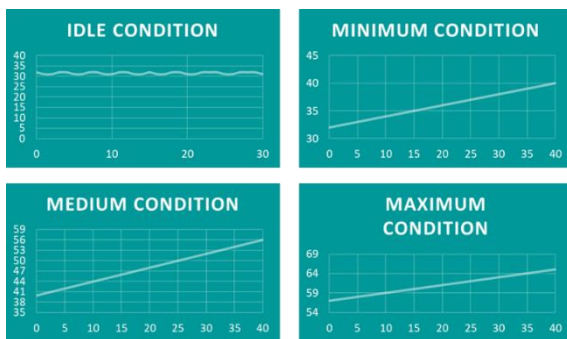


Figure 2 Expected graphical representation of different labels

The primary objective of the model training is to detect anomalies within the motor's operation. Simultaneously, the classifier component of the model will enable classification of the motor's state. An embedded ML approach is introduced in [7],

incorporates the TEDA algorithm to detect anomalies; However, this algorithm can learn independently; it requires strict prior knowledge of multidimensional variables [17, 18]. The model is trained on anomaly detection using a K-means clustering approach.

Model accuracy can be defined as the ratio of correctly predicted classifications by the model to the overall number of predictions made. Furthermore, the f1 score provides a balanced evaluation of a model's performance by combining precision and recall into a single metric in classification tasks. Equation 1 shows the formula for calculating the f1 score.

$$F1\ Score = 2 \times \frac{recall \times precision}{recall + precision} \tag{1}$$

3.0 RESULTS AND DISCUSSION

In this section, the results of the experiment are going to be discussed. Also, the detailed design of the circuit and the codes used to monitor the temperature and connect the model with the sensor. The results of the model training process using the Edge Impulse platform will also be mentioned.

To collect the temperature data, a circuit that contains the μ C, the sensor, and the breakout board - which works as a bridge between the μ C and the sensor- is built. Figure 3 shows the schematic for the circuit. As shown, the sensor is connected to the breakout board, and the board is connected to the μ C. The sensor probe is placed on the motor's casing Figure 4.

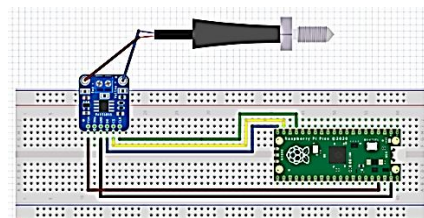


Figure 3 Circuit schematic for the data collection circuit

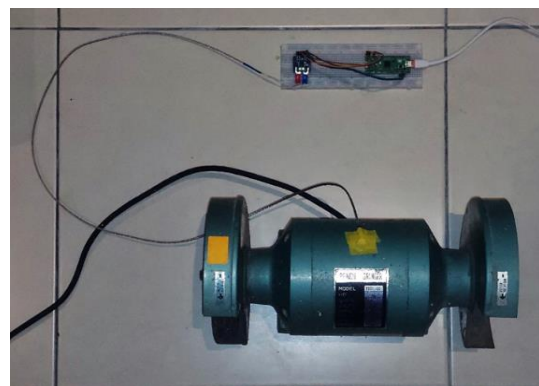


Figure 4 Data collection circuit with the probe attached to the motor

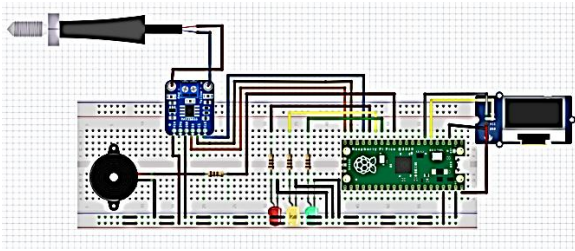


Figure 5 Full circuit schematic



Figure 6 Full circuit connection with the sensor probe attached to the motor

After the deployment of the model, a new circuit is required to make the system more independent. Although the data collection circuit can be used to detect anomalies, it is necessary to connect it to a serial monitor (i.e., ARDUINO IDE Serial Monitor) to be able to observe the output of the ML model. Figure 5 shows the complete circuit schematic. Three LED lights are used to show the prediction output of the model. The buzzer will alert users if an anomaly is detected. Finally, the LCD will show the current temperature of the motor's casing.

After preparing the circuit, a code is uploaded to the μC to read the temperature data from the sensor. Figure 6 shows the experimental setup. The rate of data read is one per second or 1 Hz. Hence the delay is 1000 ms.

The data is collected using the command prompt and Edge Impulse CLI tools. Using the "edge-impulse-data-forwarder" command line, the μC is connected directly to the edge impulse platform. The length of each dataset is set in the Edge Impulse platform. Figure 7 shows the graphical representation of the collected data for the four labels.

The data is divided into 80% and 20% for training and testing, respectively. The training dataset is used to train the model, whereas the testing dataset is used to test the model after training. Figure 8 shows the confusion matrix for the training result of the classifier. The model's accuracy is about 95%, and its average $f1$ 0.94. after training the classifier.

In Figure 9, the blue circled shapes represent the training data, whereas the purple oval shapes are called clusters. In K-means clustering, the data is placed into groups or clusters; when new data is

detected, the anomaly detection model will calculate the distance between the new data point and the edge of the closest cluster to that data point. This distance is the anomaly score.

Sometimes, the anomaly score is calculated in a negative value because sometimes the data is within the cluster. Thus, the distance is calculated in negative. Following the training phase, the testing data is used to test the model accuracy and $f1$ score.

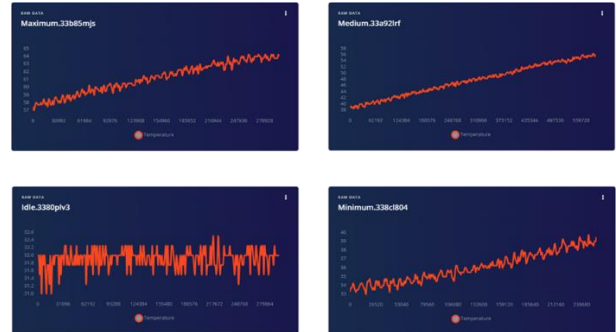


Figure 7 Graphical representation of actual collected data for the four labels

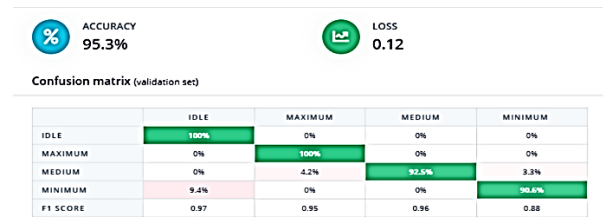


Figure 8 Resultant confusion matrix after the NN classifier training process

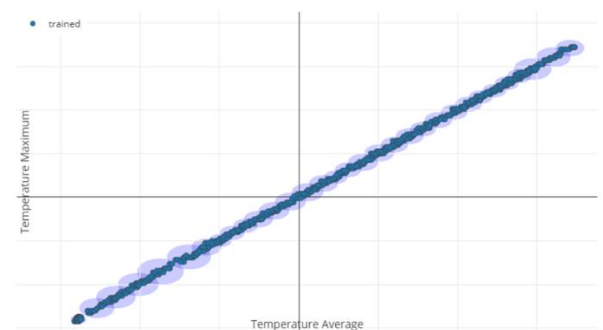


Figure 9 Anomaly detection explorer window

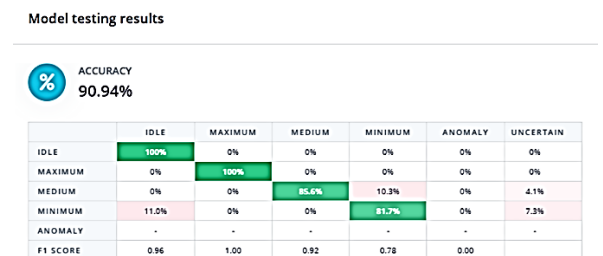


Figure 10 Resultant confusion matrix after testing the ML model

Table 2 Circuit output for each predicted output

Predict Label	Corresponding Output	Temperature Range (Celsius)
Idle	No output	30 - 33
Minimum	Green LED Constant Light	33 – 40
Medium	Orange LED Constant Light	40 – 56
Maximum	Red LED Constant Light	56 - 65
Anomaly	Red LED Blink Buzzer Turns On	Anything outside of the Normal Data

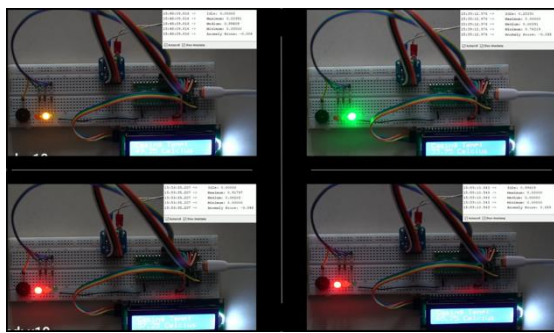


Figure 11 Different stages of the output of the circuit based on the temperature

Figure 10 shows the confusion matrix of the model after using the test data, the model accuracy is about 90%, and its average *f1* score is 0.91.

After testing the model, it is downloaded as an Arduino IDE library. After adding it to the Arduino IDE program, the library is added along with the breakout board library, and the code is modified so that the sensor can directly send the data to the model. The model is trained to collect data for up to 15 seconds; after that, it will process the data to extract features like average, RMS, and maximum temperature values.

Figure 11 shows the output of the circuit at different stages; Table 3 explains each step and its corresponding output. After deploying the model on the μC , it was noticed that most misclassified data are usually when the motor is running at maximum temperature. This is because when the motor approaches its maximum temperature, the increase in temperature starts to reduce, and the temperature change in time almost equals zero. Therefore, the model sometimes misclassifies maximum stress level as idle, Figure 12.

Moreover, anomaly usually occurs after the motor reaches its maximum stress level. However, the model detected an anomaly when the motor was not running (in idle condition) because of a temperature reading lower than usual, Figure 13.

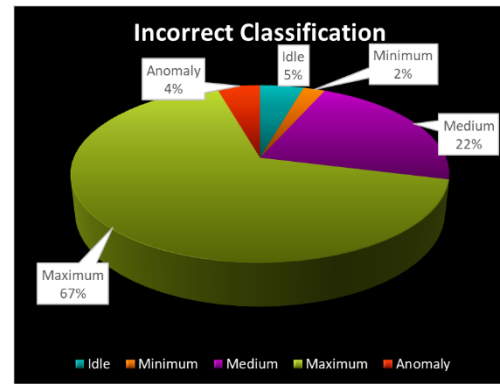


Figure 12 Percentage of incorrect classification for all outputs

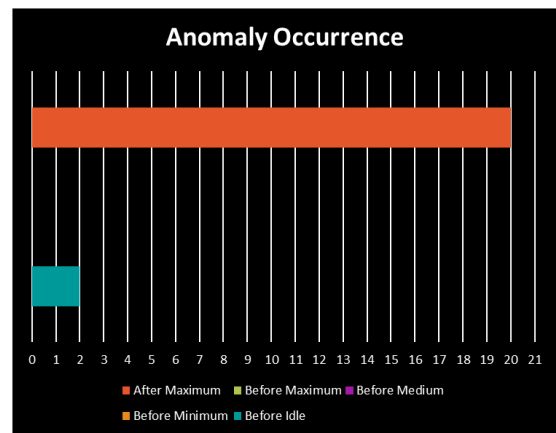


Figure 13 State of Motor at which anomaly occurs

4.0 CONCLUSION

The developed ML model returned an accuracy of approximately 90% and an average *f1* of 0.91. Also, some outputs were added to the circuit to increase the project's independence.

To increase the accuracy and reliability of the project, another ML model could be developed to detect anomalies in other parameters, like the motor's vibrations, similar to [19]. Furthermore, a control circuit could be designed to make the project fully unmanned so that when an abnormality is detected, the control circuit will automatically shut down the motor and rerun it when an anomaly is no longer occurring, as in [20].

Finally, the project can be applied to many industrial machines due to its low cost, allowing the industry to use predictive maintenance plans without spending tremendous amounts of resources to implement them.

Conflicts of Interest

The author(s) declare(s) that there is no conflict of interest regarding the publication of this paper.

Acknowledgment

This work has been supported by Universiti Teknologi Malaysia through Hi-Tech (F4) Q.J130000.4623.00Q15 grant.

References

- [1] R. A. Câmara, H. S. Mamede, and V. D. d. Santos. 2019. Predictive Industrial Maintenance with a Viable Systems Model and Maintenance 4.0. *2019 8th International Conference On Software Process Improvement (CIMPS)*, 23-25 Oct. 2019. 1-8. Doi: 10.1109/CIMPS49236.2019.9082435.
- [2] J. Zenisek, F. Holzinger, and M. Affenzeller. 2019. Machine learning Based Concept Drift Detection for Predictive Maintenance. *Computers & Industrial Engineering*. 137: 106031. Doi: <https://doi.org/10.1016/j.cie.2019.106031>.
- [3] J. Rabatel, S. Bringay, and P. Poncelet. 2011. Anomaly Detection in Monitoring Sensor Data for Preventive Maintenance. *Expert Systems with Applications*. 38(6): 7003-7015. Doi: 10.1016/j.eswa.2010.12.014.
- [4] S. K. Bose, B. Kar, M. Roy, P. K. Gopalakrishnan, and A. Basu. 2019. AdepoS: Anomaly Detection based Power Saving for Predictive Maintenance using Edge Computing. *Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC*. 597-602. Doi: 10.1145/3287624.3287716.
- [5] I. T. Christou, N. Kefalakis, A. Zalonis, and J. Soldatos. 2020. Predictive and Explainable Machine Learning for Industrial Internet of Things Applications. *2020 16th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 25-27 May 2020. 213-218. Doi: 10.1109/DCOSS49796.2020.00043.
- [6] Q. Cao et al. 2022. KSPMI: A Knowledge-based System for Predictive Maintenance in Industry 4.0. *Robotics and Computer-Integrated Manufacturing*. 74: 102281. Doi: 10.1016/j.rcim.2021.102281.
- [7] P. Andrade et al. 2021. An Unsupervised TinyML Approach Applied for Pavement Anomalies Detection under the Internet of Intelligent Vehicles. *2021 IEEE International Workshop on Metrology for Industry 4.0 and IoT, MetroInd 4.0 and IoT 2021 - Proceedings*. 642-647. Doi: 10.1109/MetroInd4.0IoT51437.2021.9488546.
- [8] F. Sakr, F. Bellotti, R. Berta, and A. De Gloria. 2020. Machine Learning on Mainstream Microcontrollers. *Sensors (Switzerland)*. 20(9): 2638. Doi: 10.3390/s20092638.
- [9] H. Kayan, Y. Majib, W. Alsafery, M. Barhamgi, and C. Perera. 2021. AnoML-IoT: An End to End re-configurable Multi-protocol Anomaly Detection Pipeline for Internet of Things. *Internet of Things (Netherlands)*. 16: 100437. Doi: 10.1016/j.iot.2021.100437.
- [10] L. Buonanno, D. Di Vita, M. Carminati, and C. Fiorini. 2020. A Directional Gamma-Ray Spectrometer with Microcontroller-Embedded Machine Learning. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 10(4): 433-443. Doi: 10.1109/JETCAS.2020.3029570.
- [11] C. Banbury et al. 2021. Micronets: Neural Network Architectures for Deploying TinyML Applications on Commodity Microcontrollers. *Proceedings of Machine Learning and Systems*. 3.
- [12] B. Sudharsan et al. 2021. TinyML Benchmark: Executing Fully Connected Neural Networks on Commodity Microcontrollers. *2021 IEEE 7th World Forum on Internet of Things (WF-IoT)*, 14 June-31 July 2021. 883-884. Doi: 10.1109/WF-IoT51360.2021.9595024.
- [13] C. Banbury et al. 2021. *MLPerf Tiny Benchmark*.
- [14] A. Acernese, C. Del Vecchio, M. Tipaldi, N. Battilani, and L. Glielmo. 2021. Condition-based Maintenance: An Industrial Application on Rotary Machines. *Journal of Quality in Maintenance Engineering*. 27(4): 565-585. Doi: 10.1108/JQME-10-2019-0101.
- [15] S. Givnan, C. Chalmers, P. Fergus, S. Ortega-Martorell, and T. Whalley. 2022. Anomaly Detection Using Autoencoder Reconstruction upon Industrial Motors. *Sensors (Basel)*. 22(9). Doi: 10.3390/s22093166.
- [16] X. Zhao, Z. Liu, T. Wang, J. Bin, and M. Jia. 2020. Unsupervised Fault Diagnosis of Machine via Multiple-Order Graphical Deep Extreme Learning Machine. *2020 Asia-Pacific International Symposium on Advanced Reliability and Maintenance Modeling (APARM)*, 20-23 Aug. 2020. 1-6. Doi: 10.1109/APARM49247.2020.9209447.
- [17] C. Lou and X. Li. 2018. Unsupervised Fault Detection based on Laplacian Score and TEDA. *Proceedings of 2018 IEEE 7th Data Driven Control and Learning Systems Conference, DDCLS 2018*. 267-270. Doi: 10.1109/DDCLS.2018.8515956.
- [18] B. S. J. Costa, C. G. Bezerra, L. A. Guedes, and P. P. Angelov. 2015. Online Fault Detection based on Typicality and Eccentricity Data Analytics. *2015 International Joint Conference on Neural Networks (IJCNN)*, 12-17 July 2015. 1-6. Doi: 10.1109/IJCNN.2015.7280712.
- [19] T. K. Nguyen, I. Azarkh, B. Nicolle, G. Jacquemod, and E. Dekneuve. 2018. Applying NIALM Technology to Predictive Maintenance for Industrial Machines. *2018 IEEE International Conference on Industrial Technology (ICIT)*, 20-22 Feb. 2018. 341-345. Doi: 10.1109/ICIT.2018.8352201.
- [20] H. Lu, Y. Li, S. Mu, D. Wang, H. Kim, and S. Serikawa. 2018. Motor Anomaly Detection for Unmanned Aerial Vehicles Using Reinforcement Learning. *IEEE Internet of Things Journal*. 5(4): 2315-2322. Doi: 10.1109/JIOT.2017.2737479.