Intelligent Bearing Fault Detection: Deep Learning Model Assessment and Embedded System Deployment

A. F. Cotrino Herrera, J. A. López Sotelo, *Senior Member, IEEE*, A. Toro Lazo, and J. C. Blandón Andrade

***Abstract*—Bearings are vital parts used in many industrial settings; however, their failures can greatly reduce system efficiency and operational reliability. In this context, AI-driven predictive maintenance is an effective approach for identifying and classifying bearing faults via vibration analysis. This study creates a custom dataset by recording vibration signals from a DC motor operating under various fault conditions (bearing without lubrication, bearing with one missing ball, and bearing with two missing balls), as well as under normal operation (bearing without failure), using an accelerometer and a controlled test bench. Furthermore, four neural network models — Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Transformer — were trained and evaluated based on accuracy, recall, and F1-score. The CNN model performed best, achieving a 99.95% accuracy on the validation dataset. This model was then implemented on an ESP32, reaching 94.2% accuracy during real-time testing. These results demonstrate that AI-based fault detection systems can be effectively deployed on resource-limited platforms, providing a promising solution for predictive maintenance and educational efforts to boost STEM skills.**

Link to graphical and video abstracts, and to code: https://latamt.ieeer9.org/index.php/transactions/article/view/10411

***Index Terms*—Artificial Intelligence, Ball Bearings, Data acquisition, Data analysis, Deep Learning, Spectral Analysis, TinyML, Vibration measurement.**

I. Introduction

Currently, bearings are essential components across various industrial sectors, performing critical functions such as reducing friction, supporting loads, and guiding the rotation of mechanical elements [1]. They are present in a wide range of applications, including electric motors and transportation equipment, where their proper operation is crucial for ensuring efficiency and operational safety. In fact, it is estimated that two out of three motor failures originate from damage initiated in the bearings [2]. Common failures include wear, imbalance, damage to the raceways, and lack of lubrication, among others [3].

Today, emerging technologies such as the Internet of Things (IoT) and artificial intelligence (AI) offer new tools to address these challenges [2][4][5]. Data analysis techniques enable the detection of bearing anomalies, mitigate risks, and optimize maintenance strategies. This leads to predictive maintenance, which employs non-invasive methods to detect faults before they occur [6]. In this context, neural networks have proven valuable for identifying and classifying vibration patterns associated with such conditions [7].

Over the years, significant advances have been made in developing robust models that adapt to diverse conditions. This progress has enabled the implementation of real-time monitoring systems, even on embedded platforms, bringing these solutions closer to more accessible and dynamic industrial environments.

This work aims to contribute to the development of AI- and embedded-system-based solutions for the detection and classification of bearing faults, promoting their efficient and accessible application across various settings, primarily educational environments. The approach encompasses data acquisition, dataset creation, neural network training, and deployment to validate system performance.

This paper is organized as follows. Section II presents the related work, summarizing the most relevant studies on vibration analysis and bearing fault diagnosis using artificial intelligence. Section III details the proposed methodology, including data acquisition, dataset preparation, neural network architectures, and model deployment on an embedded system. Section IV discusses the results of the experiments, evaluating the performance of the implemented models and analyzing real-time validation results on the embedded device. Finally, section V concludes the paper by highlighting the main findings, identifying the limitations encountered, and suggesting future research directions for extending this work.

II. Related Work

Since the emergence of neural networks, numerous

The associate editor coordinating the review of this manuscript and approving it for publication was Guillermo Valencia-Palomo (*Corresponding author: Andres Felipe Cotrino Herrera*).

This article is a result of the research project "Application of Internet of Things and Artificial Intelligence Technologies to Predictive Maintenance", internal code CI-023-01, carried out at the Universidad Católica de Pereira.

Andres Felipe Cotrino Herrera, and J. A. López Sotelo are with the School of Engineering and Basic Sciences, Universidad Autónoma de Occidente, Cali, Colombia (e-mails: andres.cotrino@uao.edu.co, and jalopez@uao.edu.co).

A. T. Lazo, and J. C. B. Andrade are with Systems and Telecommunications Engineering Program, Universidad Católica de Pereira, Pereira, Colombia (e-mails: alonso.toro@ucp.edu.co, and juanc.blandon@ucp.edu.co).

experiments, applications, and prototypes have been developed for vibration analysis, driven by the recognized potential of these algorithms compared to traditional methods for the same task, such as time-domain and frequency-domain analyses. Moreover, AI-related methods are easier to implement and do not require extensive expertise or specialized knowledge [8].

As a result, deep learning-based methods have been developed to diagnose bearing faults, overcoming the limitations of traditional approaches that rely heavily on feature extraction and other data processing techniques. Among these, convolutional neural networks (CNNs) stand out for their ability to directly process vibration signals and produce optimal classifications. This approach eliminates the need for manual feature extraction techniques, achieving high accuracy and robustness even under noisy conditions and varying loads [9].

An innovative approach to machine fault diagnosis and monitoring is the development of a non-invasive IoT system that identifies and predicts fault severity in real time. In this study [10], machine learning techniques such as Support Vector Machines (SVMs), K-Nearest Neighbors (KNNs), and Random Forests were applied, with the latter achieving an accuracy of 81.41%. The methodology enables classification of operating conditions, such as normal state, imbalance, and misalignment, and integrates a wireless monitoring and alert system that interacts with mobile devices.

In response to the high demand for fault diagnosis solutions in industrial plants—aimed at reducing unscheduled downtime, performance degradation, among other issues—a study [11] proposed an end-to-end methodology based on a Convolutional Recurrent Neural Network (CRNN). This method processes raw vibration data directly, without preprocessing or feature extraction. It is distinguished by its ability to manage large volumes of data in real time, as well as its computational and training efficiency, making it highly suitable for industrial applications. The method achieved high accuracy across datasets, reaching 97.13% on the IMS (Intelligent Maintenance Systems) dataset and 99.77% on the Case Western Reserve University (CWRU) dataset.

Another notable study proposed a methodology for bearing fault detection and diagnosis, combining Variational Mode Decomposition (VMD) with one-dimensional Convolutional Neural Networks (1D-CNNs) [12]. In this research, VMD was employed as a filtering technique to highlight fault-related components in signals contaminated by harmonics. Subsequently, the extracted features were processed through a 1D-CNN to classify and diagnose faults with various levels of severity. Results obtained with the CWRU dataset demonstrated the superior performance of the proposed model (measured by accuracy), outperforming traditional methods such as Empirical Mode Decomposition (EMD) and other deep learning approaches (MLP, RNN, and LSTM).

An advanced approach to rolling bearing fault diagnosis is presented through the use of deep learning models capable of handling non-stationary and time-varying vibration signals. In this study [13], an optimized self-adaptative Deep Belief Network (SADBN) is proposed, where the network parameters are tuned with the Slap Swarm Algorithm (SSA). Experimental validation using the CWRU dataset demonstrates that the proposed model outperforms conventional methods and standard DBN architectures, achieving an accuracy above 94%.

Additionally, a recent study [14] evaluated six supervised learning models—Decision Tree, Discriminant Analysis, Naive Bayes, SVM, KNN, and Ensemble Classifier—on vibration data from the CWRU dataset. Notably, the Gaussian Naive Bayes model achieved 97.5% accuracy in fault classification. This superior performance can be attributed to the application of Principal Component Analysis (PCA)-based feature selection, which reduces feature correlation and aligns well with the conditional independence assumption of Naïve Bayes. Furthermore, the relatively small dataset size likely contributed to the effectiveness of the probabilistic Gaussian modeling. In addition to its high accuracy, Gaussian Naive Bayes demonstrated lower training time and high prediction speed compared to other models with similar accuracy, reinforcing its suitability for real-time fault diagnosis applications.

III. Methodology

To perform the vibration analysis of a bearing driven by a DC motor, this study proposes evaluating different neural network architectures, encompassing data acquisition, training, and validation, to select the most suitable model for deployment on an embedded device and verify its real-time performance.

A. Types of Vibrations to be Classified

Based on the research conducted [3], five vibration types were identified to form the dataset: normal bearing, bearing without lubrication, bearing missing one ball, bearing missing two balls, and a final class corresponding to the motor off. Previous studies [11],[12],[13],[14], have shown that bearing faults generate characteristic frequency signatures associated with specific geometric defects, such as inner race, outer race, and rolling element faults. However, the proposed scheme is not intended to discriminate between standardized race fault types, but rather to detect abnormal bearing conditions within a controlled experimental and educational setup. For this reason, the fault scenarios considered in this work were selected based on their ease of physical implementation and repeatability in a low-cost test bench, enabling the generation of distinguishable vibration patterns in a simple and reproducible experimental environment.

A previously developed test bench was used for this purpose, consisting of a base that supports the motor and bearing, and a protoboard for connecting the sensor that captures vibration data. Fig. 1 shows the complete test bench setup.

The design shown in Fig. 1 allows the user to easily interchange the bearing, facilitating the collection of samples for each fault type using the same setup and thus ensuring

uniformity in the experimental conditions [15].

It is important to note that the proposed test bench was intentionally designed as a controlled experimental setup and does not aim to fully replicate the mechanical complexity of an industrial environment. The objective of this configuration

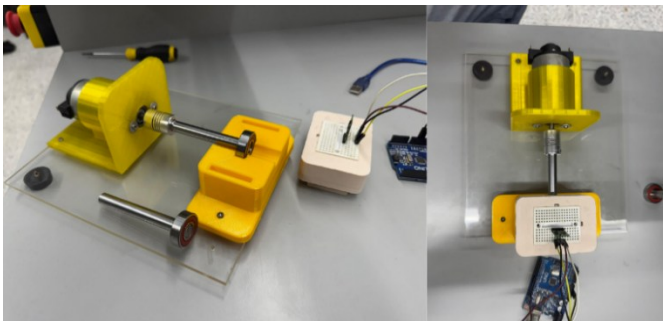

Fig. 1. Testing bench (Motor, Sensor Setup, Bearing).

is to isolate bearing-related vibration patterns and ensure repeatable data acquisition, rather than to capture all possible mechanical perturbations present in real industrial machinery.

Based on this, vibration data were collected using the Pololu MinIMU-9 v5 inertial sensor, which served as the primary sensing device throughout the study. This accelerometer was employed to acquire tri-axial acceleration signals (X, Y, and Z), expressed in units of gravitational acceleration (g), from different bearings and fault conditions. The accelerometer orientation defines how vibration components are projected onto the X, Y, and Z axes. In this study, the sensor can be mounted in any orientation, provided it remains consistent throughout data acquisition and deployment. Preserving the orientation does not affect classification performance, as tri-axial signals are jointly used, whereas changing the orientation between measurements may introduce discrepancies in the results.

The recorded data were transmitted to the Edge Impulse platform for organization and labeling, resulting in a structured dataset organized by class. To ensure robust data, 150 samples (30 per class) were collected, each lasting 20 seconds and sampled at 50 Hz.

The embedded device itself does not intrinsically modify the sensor's sampling frequency; instead, the effective sampling rate is defined by the data acquisition strategy implemented in software. In this work, the microprocessor controls the timing of accelerometer readings through its internal timers, ensuring a fixed sampling frequency of 50 Hz during data acquisition.

The proposed scheme is therefore dependent on the sampling frequency, since changes in the acquisition rate alter the temporal resolution and frequency content of the vibration signals. For this reason, the sampling frequency was kept consistent during dataset creation, training, validation, and embedded deployment. Any change in the sampling frequency would require signal resampling or model retraining to preserve classification performance.

Subsequently, the dataset was divided into training (70%), validation (20%), and test (10%) subsets. The data splitting was performed at the sample level, ensuring that complete 20-second recordings were assigned exclusively to one subset, thereby preventing data leakage between them. Fig. 2 presents the acceleration data for each class, obtained from the sensor.

A high-quality dataset is essential to ensure that a deep learning model is accurate, generalizable, and truly representative of the real-world problem. Poorly labeled,

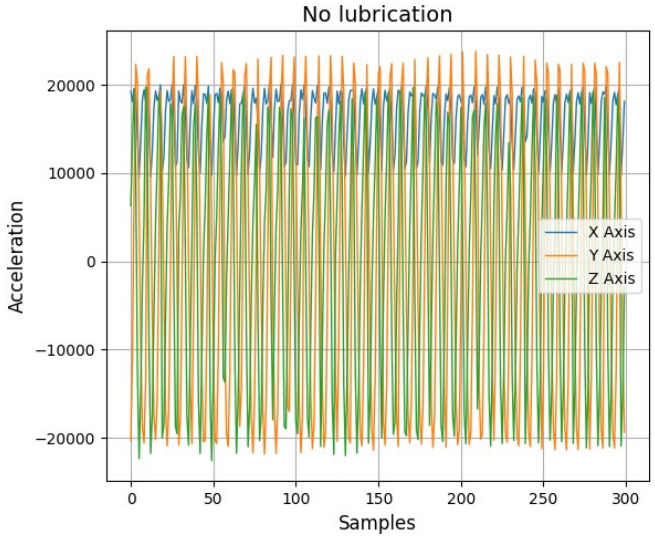

Fig. 2. Raw acceleration data collected per class (Ex: No lubrication).

biased, or insufficient data can lead to overfitting, inaccurate predictions, or the perpetuation of biases, affecting both the model's performance and its practical utility [16][17]. Moreover, a balanced and diverse dataset enables the capture of relevant patterns and proper model evaluation, helping to avoid unnecessary iterations and reducing both development time and costs.

B. Neural Networks to be Implemented

Currently, a wide variety of artificial intelligence models are applied to the classification of acceleration data, ranging from Transformers to multilayer perceptrons (MLPs) [5][18]. Each of these models involves a distinct type of data processing and presents advantages and disadvantages when deployed on embedded devices [19], which is the focus of this research.

One of the main advantages of a multilayer perceptron (MLP) is its simplicity and the speed of its implementation and training. This architecture allows for efficient deployment in terms of computational resource consumption (on embedded or low-cost platforms). Since it requires fewer parameters than more complex models, MLP training is typically faster and requires less processing power and memory. However, to optimize performance in classification and pattern recognition tasks, data preprocessing is often needed. A common example is spectral feature extraction; a technique used on platforms such as Edge Impulse [20]. By highlighting the relevant features of raw data, this preprocessing step facilitates the network's learning, enabling

it to identify complex patterns more effectively [12][21].

On the other hand, more complex models such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and Transformers can work directly with raw sensor data. These models learn hierarchical representations from the original data, allowing for more precise identification of complex patterns. However, this approach comes at the cost of higher computational resource consumption during both training and inference, which limits its deployment on devices with memory, processing, or energy constraints [12][22][23].

Based on this, four types of neural networks have been selected: multilayer perceptron (MLP), convolutional network (CNN), Long Short-Term Memory (LSTM), a type of Recurrent Neural Network (RNN), and Transformer. Simple models were constructed to achieve high accuracy, optimizing performance without overloading the embedded device where the model will be deployed.

The selection of the neural network architectures was guided by the objective of comparing representative learning paradigms commonly used in vibration-based fault diagnosis. Specifically, the selected models cover a spectrum ranging from feature-based learning (MLP) to local pattern extraction from raw signals (CNN), temporal dependency modeling (LSTM), and global context modeling through attention mechanisms (Transformer), which enables the modeling of long-range temporal dependencies within vibration signals. This is particularly relevant for bearing fault diagnosis, where impulsive and periodic patterns may occur at non-local time positions within a signal window. This choice enables a systematic evaluation of how different signal-processing and learning strategies impact classification performance, computational cost, and deployability on low-resource platforms.

C. Data Preprocessing: Spectral Feature Extraction and Windowing

Although frequency-domain analysis is effective for detecting specific bearing faults under well-defined conditions, it cannot be able to easily address all fault scenarios, especially when fault characteristics evolve or overlap across classes. In this work, artificial intelligence is therefore used as a complementary approach to automatically learn discriminative patterns from spectral features, temporal variations, and multi-axis vibration signals without relying on predefined thresholds or explicit fault models [24].

To successfully train a multilayer perceptron, it is necessary to preprocess the dataset through spectral analysis of the vibration signals. This technique allows the signal to be examined in terms of frequency, making it easier to identify patterns, especially when working with accelerometer data. Fig. 3 shows a graphical representation of how a time-domain signal is transformed into the frequency domain.

The data obtained from this type of analysis is varied; however, for this study, the following features are calculated: dominant frequency, total energy, low-, medium-, and high-band energy, average amplitude, and amplitude standard deviation. Each of these variables is computed for each acceleration axis (x, y, z), yielding a total of 21 spectral features for network training.

Fig. 4 shows a graphical representation of the spectral features calculated for a random sample from each class of the dataset. The graph shows the different values and behaviors depending on the type of fault in the bearing.

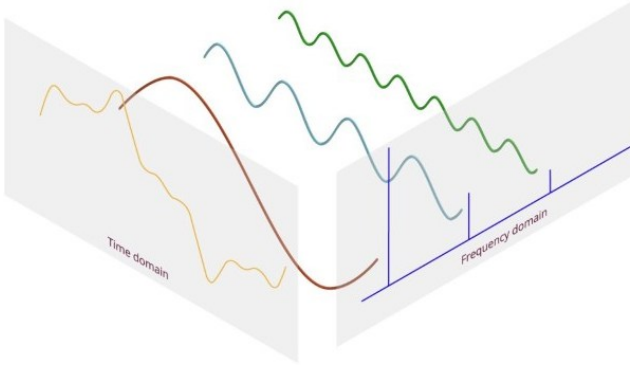

Fig. 3. Spectral Analysis representation: Time domain to Frequency domain [25].

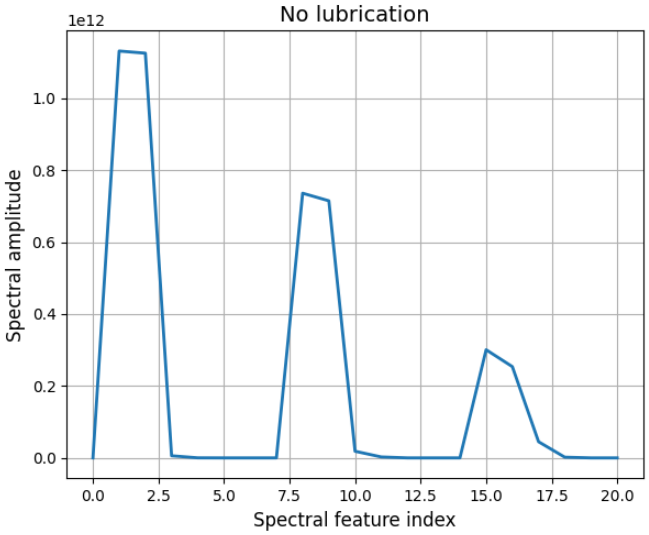

Fig. 4. Graphical representation of Spectral features (Ex: No lubrication).

Additionally, to improve training performance, a sliding window scheme was applied to the data. This method involves dividing the vibration signals into smaller segments (1-second windows) that slide across the sample with a fixed time step. In this way, a better representation of dynamic behavior is achieved, and the amount of data available for model training is increased.

The data processing flow for the implemented networks is presented in Fig. 5.

As shown in Fig. 5, preprocessing using sliding windows is an essential step in data preparation and is crucial to the performance of all the models implemented in this study. Although the sliding-window segmentation is applied uniformly, the multilayer perceptron (MLP) relies on an additional stage of spectral feature extraction from each

window. In this work, these processed spectral features were used to reduce the computational load of the MLP and to provide a more compact, descriptive representation of the vibration signal, given its limited ability to capture temporal dependencies directly from raw sequences.

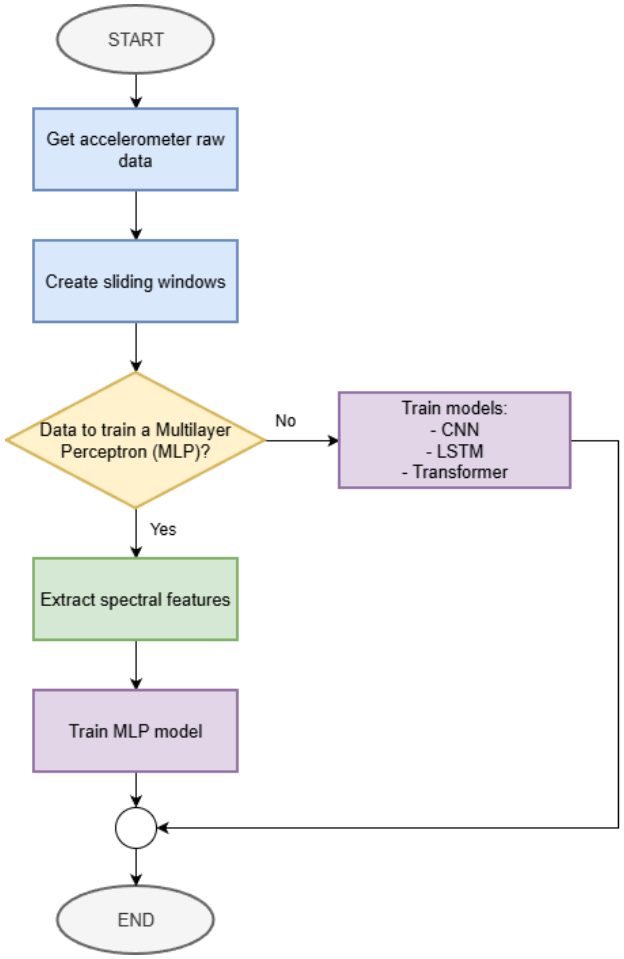

Fig. 5. Process flowchart for model training.

In contrast, the CNN, LSTM, and Transformer architectures are trained directly on the segmented vibration data generated by sliding windows, without further spectral preprocessing. In other words, while the MLP operates on processed features, these deep learning models work with windowed raw data, allowing them to learn meaningful temporal patterns directly from the signal. The use of sliding windows not only standardizes the input representation but also increases the amount of available training data, which is beneficial for deep architectures that typically require larger datasets. Additionally, dividing the signal into shorter, fixed-length windows makes the training process more computationally manageable for the models, improving efficiency while preserving the signal's essential temporal dynamics.

D. Neural Network Models Construction

When working with neural network models, it is possible to go from the simplest to the most complex by varying the number of layers, neurons, and additional operations. Based on this, it is essential to consider that the best model should be implemented on a device with limited resources (e.g., an ESP32), so it is necessary to balance high accuracy and model performance while minimizing the resources required for execution.

Based on the discussion in this section and using the TensorFlow library, the proposed architecture approaches are described below.

1) Multilayer Perceptron (MLP)

The neural network receives a set of 21 spectral features as input. To construct the model layer by layer, the Sequential API in Keras was used. The designed architecture consists of three hidden layers with 20, 15, and 10 neurons, respectively, followed by an output layer with five neurons, corresponding to the five classes of the classification problem. The neural network architecture is shown in Table I.

TABLE I
LAYER-BY-LAYER ARCHITECTURE OF THE PROPOSED MULTILAYER PERCEPTRON (MLP)

Stage	Layer Type	Parameters / Details	Output Shape
Input	Input	Features = 21	*(21)*
Dense Layer 1	Dense	Units = 20 Activation = ‘relu’	*(20)*
Dense Layer 2	Dense	Units = 15 Activation = ‘relu’	*(15)*
Dense Layer 2	Dense	Units = 10 Activation = ‘relu’	*(10)*
Output layer	Dense	Units = 5 Activation = 'softmax'	*(5)*

2) Convolutional Neural Network (CNN)

This network takes as input the time windows generated during data preprocessing, which have a size of (64, 3). Based on this, and once again using the Sequential API, a 1-dimensional convolutional layer with 32 filters of size 3 (kernel size) is added, followed by a dimensionality-reduction layer, a dense layer with 32 neurons, and a 5-neuron output layer for classification. The architecture is presented in Table II.

TABLE II
LAYER-BY-LAYER ARCHITECTURE OF THE PROPOSED CONVOLUTIONAL NEURAL NETWORK (CNN)

Stage	Layer Type	Parameters / Details	Output Shape
Input	Input	Sequence length = 64 Features = 3	*(64, 3)*
Convolution	Conv2D	Filters = 32 Activation = ‘relu’ Kernel Size = 3	*(62, 32)*
Global Pooling	GlobalAveage-Pooling2D		*(32)*
Dense Layer	Dense	Units = 32 Activation = ‘relu’	*(32)*
Output layer	Dense	Units = 5 Activation = 'softmax'	*(5)*

3) Long Short-Term Memory (LSTM)

Similar to the CNN architecture, this network takes as input the time window sizes from preprocessing. To obtain a lightweight model, the Sequential API of Keras is used, with an LSTM layer with 64 units followed by a dense layer with five neurons for classification. Network architecture is shown in Table III.

TABLE III
LAYER-BY-LAYER ARCHITECTURE OF THE PROPOSED LONG SHORT-TERM MEMORY (LSTM) NETWORK

Stage	Layer Type	Parameters / Details	Output Shape
Input	Input	Sequence length = 64 Features = 3	*(64, 3)*
Recurrent Layer	LSTM	Units = 64	*(62)*
Output layer	Dense	Units = 5 Activation = 'softmax'	*(5)*

4) Transformer Architecture

To implement a simple architecture, a model uses a single multi-head attention layer with only two attention heads. This is followed by dimensionality reduction, a dense layer of thirty-two neurons, and a 5-neuron output layer for multi-class classification. Due to model length, the architecture is presented in Table IV.

TABLE IV
LAYER-BY-LAYER ARCHITECTURE OF THE PROPOSED TRANSFORMER-BASED MODEL

Stage	Layer Type	Parameters / Details	Output Shape
Input	Input	Sequence length: 64 Features: 3	*(64, 3)*
Input projection	Dense	Units = 32	*(64, 32)*
Multi-head attention	MultiHeadAttention	num_heads = 2 key_dim = 32	*(64, 32)*
Normalization	Layer Normalization	Epsilon = 1e-6	*(64, 32)*
Global pooling	GlobalAveragePooli-ng1D	Averages across the time steps	*(32)*
Dense hidden layer	Dense	Units =32 Activation = 'relu'	*(32)*
Regularization	Dropout	Rate = 0.5	*(32)*
Output layer	Dense	Units = 5 Activation = 'softmax'	*(5)*

To mitigate overfitting, the models were deliberately designed with lightweight architectures that limit the number of trainable parameters. In addition, regularization strategies such as dropout and early stopping were applied during training. Sliding-window segmentation was performed after dataset partitioning, increasing the effective number of training instances while preserving class separation and preventing data leakage.

Overall, the resulting architectures are considerably lighter than those typically used in high–computational-capacity environments. This design choice is essential for deployment on low-power embedded devices, such as the ESP32, where processing and memory resources are limited, while still maintaining competitive classification performance. The results from the training and validation process are presented in the next section.

E. Exporting the Selected Model to an Embedded Device

Once the model to be implemented on the embedded system — in this case, an ESP32 — is selected, the next step is to export it in the appropriate format. The choice of this device is justified by its low cost, low energy consumption, adequate processing power to run lightweight artificial intelligence models, and its integrated Wi-Fi connectivity, which enables future IoT projects.

To facilitate the model integration into the development environment, the Edge Impulse platform [20] provides a Python API that allows exporting the model as a C++ library for importing into the Arduino IDE. It is important to note that the model must first be converted to the TensorFlow Lite (TFLite) format, which imposes certain restrictions on the types of operations and network structures allowed due to compatibility limitations with microcontrollers.

While Edge Impulse imposes constraints on model size, memory usage, and supported operations, these limitations are aligned with the objectives of this work, which focus on lightweight neural network architectures for reliable real-time execution on resource-constrained embedded platforms.

Once converted to TFLite, the code provided by Edge Impulse [26] is used to generate a compressed ZIP file. This file contains the necessary library, which must be imported into the Arduino IDE. From this interface, it is possible to perform tests to validate the model's accuracy using metrics such as Accuracy, Recall, and F1-Score, and to compare the results with the data obtained during development in Python.

Fig. 6 shows the steps required for implementation on the embedded device.

Once the model is loaded onto the ESP32 and the sensor is connected, the validation process for detecting multiple fault types can begin.

IV. RESULTS AND DISCUSSION

To evaluate each model's performance, post-training tests are conducted on test data. For each model, metrics such as accuracy, recall, and F1 Score are obtained, as they are key indicators for assessing performance and selecting the optimal model for deployment on the embedded device.

The number of correct predictions relative to the total number of forecasts measures accuracy. This metric helps to understand how accurate the classifications are in general. On the other hand, recall measures the model's ability to identify true positives correctly. Regarding the F1 Score, it is the harmonic mean of precision and recall, providing a balance between the two.

Fig. 7 presents a graph showing the results obtained for the

four trained neural network models, comparing the three metrics mentioned.

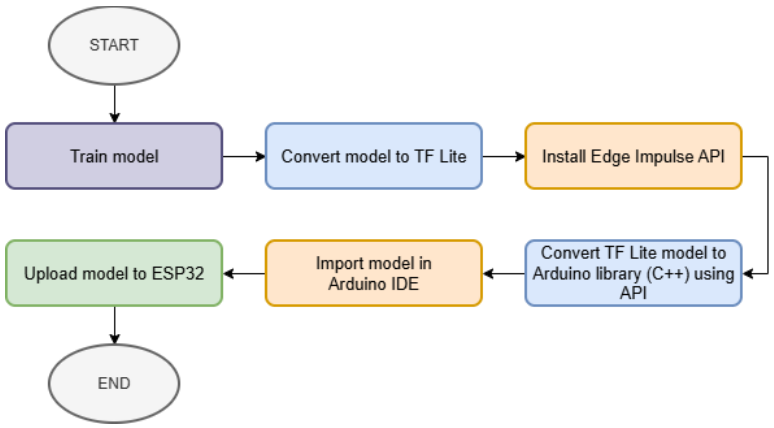

Fig. 6. Steps for embedded device implementation.

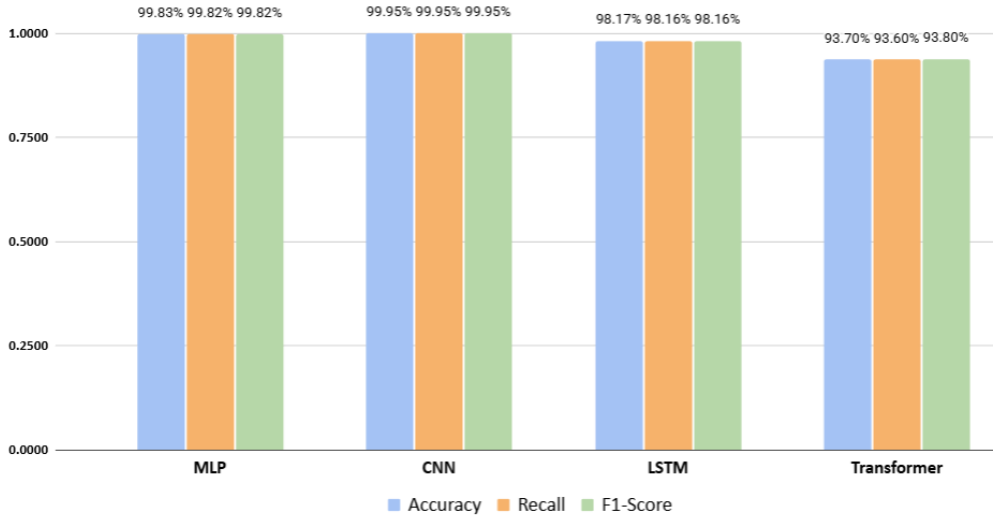

Fig. 7. Evaluation metrics for trained models.

Table V presents the values of the metrics shown in Fig. 7.

TABLE V
COMPARISON OF EVALUATION METRICS OBTAINED BY THE TRAINED MODELS

Model \ Metric	MLP	CNN	LSTM	Transformer
Accuracy	0.9983	**0.9995**	0.9817	0.937
Recall	0.9982	**0.9995**	0.9816	0.936
F1-Score	0.9982	**0.9995**	0.9816	0.938

The results indicate that the convolutional neural network (CNN) achieved the highest performance across all evaluation metrics, with the multilayer perceptron (MLP) showing comparable results. Although the LSTM and Transformer models also achieved scores above 90%, their performance was noticeably lower than that of the CNN and MLP. In particular, the Transformer model yielded the lowest metrics among all architectures. This behavior can be explained by several factors: the simplicity of the implemented Transformer architecture (a single multi-head attention layer with two heads) and the limited dataset size (150 total samples, 30 per class). Additionally, the Transformer design was intentionally kept lightweight due to the computational constraints imposed by the embedded deployment target, which required fewer parameters and layers. These limitations collectively restrict the advantages typically offered by Transformers in modeling long-range dependencies, preventing the model from reaching its full potential in this application.

From a signal-processing perspective, the superior performance of the CNN can be attributed to its ability to exploit local correlations and quasi-stationary properties of vibration signals. The convolutional layers act as adaptive filter banks, enabling efficient extraction of localized temporal and frequency-related patterns that are characteristic of bearing faults. By leveraging weight sharing and translation invariance, the CNN achieves an effective balance between classification accuracy and computational complexity, which is particularly advantageous for short, fixed-length vibration windows and embedded deployment constraints.

Given the model with the best overall performance, its deployment was conducted in C++ to package it as a library for import into the Arduino IDE and subsequently upload the

classifier to the ESP32 microcontroller. Once the sensor and test bench were connected, data were captured as time windows, maintaining the same size used during the preprocessing stage (64 samples per axis, resulting in a 64×3 matrix). Fig. 8 presents the setup of the final embedded model implementation.

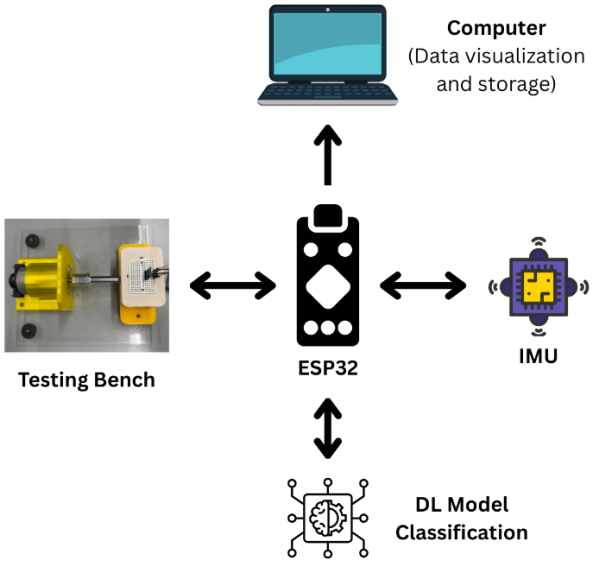

Fig. 8. Setup of final embedded model implementation.

Tests were performed across all five classes present in the dataset. For each class, more than one hundred samples were collected as the test bench operated and the bearing rotated. Based on the data collected, the success rate for each category was calculated. Fig. 9 presents a graph summarizing these results.

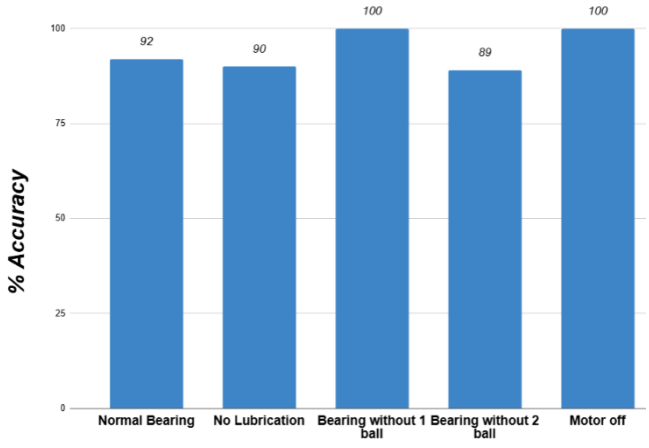

Fig. 9. Validation results per class on ESP32.

Additionally, Table VI presents the obtained values for each evaluation metric used in the model assessment. This can be used for a direct comparison with the results previously shown in Table V for the CNN model.

TABLE VI
PERFORMANCE METRICS OF THE CNN MODEL AFTER DEPLOYMENT ON THE ESP32

Model \ Metric	CNN - Deployed
Accuracy	**0.942**
Recall	**0.942**
F1-Score	**0.941**

Table VII presents embedded implementation metrics, regarding inference latency, memory usage and power consumption.

TABLE VII
COMPUTATIONAL PERFORMANCE OF THE DEPLOYED CNN MODEL ON THE ESP32

Metric	Value
Latency	**1 ms**
Memory usage	**398 kB**
Power consumption	**260 - 400 mW**

The deployed CNN model achieves an inference latency of 1 ms on the ESP32 microcontroller. The compiled firmware occupies 398 kB of memory, corresponding to approximately 30% of the available program storage and power consumption is estimated with the ESP32 datasheet. The device typically consumes between 80 and 120 mA at 3.3 V during active CPU operation, corresponding to an estimated power consumption of approximately 260 – 400 mW.

Finally, the confusion matrix generated during validation is presented, providing a more detailed overview of the model's behavior by highlighting the classes where misclassifications occurred. This matrix is shown on Fig. 10.

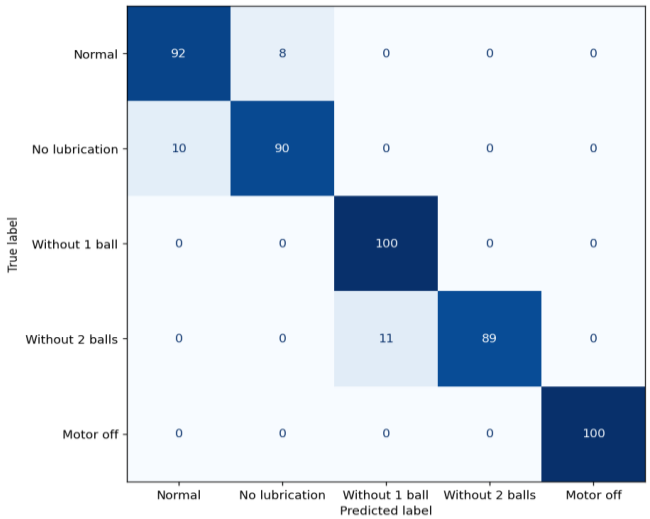

Fig. 10. Confusion Matrix of the deployed CNN model.

From the confusion matrix, it can be seen that the "motor off" condition and the "bearing missing a ball" fault were best identified by the model. On the other hand, in the "normal bearing" and "bearing without lubrication" classes, a higher degree of confusion was observed, which can be explained by the similarity of their vibration patterns.

Despite these challenges, the model achieved high accuracy across all classes, 90% or higher. Furthermore, the final model accuracy obtained during real-time validation was 94.2%. However, this value differs by almost six percentage points from the accuracy achieved during offline testing in Python.

These differences in the model's accuracy may be attributed to several factors. The following were identified.

1) Performance Degradation during Model Deployment

Due to the conversion process from a TensorFlow model to TensorFlow Lite and then to a C++ library, performance may decrease slightly when running in an embedded environment rather than on a more powerful machine used for training. Certain operations or computations essential for achieving optimal results are restricted or simplified during this transition.

2) Bearing Support Wear

During data acquisition for dataset creation and throughout extensive testing phases for validation and deployment, the 3D-printed bearing support experienced significant friction. This exposure led to an increase in temperature at the bearing's external surface, which was in contact with a Polylactic Acid (PLA) component (sensitive to elevated temperatures). Consequently, deformations such as support widening occurred, reducing grip and generating unexpected vibrations or patterns not accounted for during model training.

Based on these observations, the decrease in model accuracy is explained. Nevertheless, achieving an accuracy above 90% remains a highly promising result given the device constraints and the methodology employed.

As a complementary consideration, it is important to highlight that the mechanical issues observed in the bearing support could be mitigated in future iterations of the prototype. Since PLA is sensitive to temperature and prone to deformation under prolonged friction, other materials such as Polyethylene Terephthalate Glycol-modified (PETG) or Acrylonitrile Butadiene Styrene (ABS) could provide superior thermal resistance and greater stability during extended tests or operations. These enhancements could minimize mechanical variability and undesired vibration patterns, thereby improving the model's real-time accuracy.

From a system-level perspective, the robustness of the proposed scheme is influenced by the operating conditions under which the vibration data are acquired. If additional mechanical perturbations are introduced, such as external vibrations, load fluctuations, structural resonances, or misalignment, the vibration signal characteristics may change, potentially affecting classification performance. Since the models were trained using data acquired under controlled experimental conditions, strong perturbations not represented in the training dataset may lead to a reduction in accuracy due to distribution shift.

V. Conclusions

This study demonstrated the feasibility of implementing a bearing fault diagnosis system using deep learning models deployed on low-cost embedded systems. A proprietary dataset was developed from vibration measurements of a DC motor under different fault conditions, using an accelerometer and a controlled test bench. Experimentally, the quality and representativeness of the dataset were key factors in achieving effective model training and reliable validation on the embedded device.

Four neural network architectures (MLP, CNN, LSTM, and Transformer) were trained and evaluated using accuracy, recall, and F1-score metrics to compare their performance. The convolutional neural network (CNN) model achieved the best results, achieving 99.95% accuracy on the test data during the validation phase.

The CNN model was subsequently converted and optimized for deployment on an ESP32 microcontroller using TensorFlow Lite and C++. Although a slight decrease in performance was observed (an average accuracy of 94.2%) compared to Python-based tests, this discrepancy can be attributed to both the computational limitations of the embedded environment and the mechanical wear of the bearing support caused by extensive data acquisition sessions.

This study provides a solid foundation for future research in mechanical fault detection using AI. The use of a low-cost platform such as the ESP32, combined with TinyML tools and vibration-based diagnostics, highlights the strong educational potential of this work. Beyond demonstrating a practical application of embedded AI, the system can be easily replicated in classrooms, laboratories, and other training environments due to its affordability and straightforward implementation. Students can assemble the test bench, collect their own data, and train models, gaining hands-on experience. This makes the proposed setup an accessible and effective tool for strengthening STEM competencies.

Although the system demonstrated strong performance in the experimental setup, a significant limitation of this study is the low sampling frequency (50 Hz) used during data acquisition. While this rate was sufficient to capture the vibration patterns associated with the specific fault conditions of the used DC motor, it remains considerably below the frequencies typically required for industrial bearing fault diagnosis (kHz range). Future work should therefore include the use of higher-sampling-rate sensors and more robust acquisition hardware, as well as redesigning the test bench with heat-resistant materials such as ABS or PETG to prevent structural deformation during prolonged operation. Additionally, adopting a more powerful embedded platform could enable the use of more complex neural network architectures, potentially improving performance in real-world industrial applications.

References

[1] F. Xu, N. Ding, N. Li, L. Liu, N. Hou, N. Xu, W. Guo, L. Tian, H. Xu, C. L. Wu, X. Wu, and X. Chen, "A review of bearing failure modes, mechanisms, and causes," *Eng. Fail. Anal.*, vol. 152, Art. no. 107518, Oct. 2023, doi: 10.1016/j.engfailanal.2023.107518.

[2] L. S. Sawaqed and A. M. Alrayes, "Bearing fault diagnostic using machine learning algorithms," *Prog.*

Artif. Intell., vol. 9, no. 4, pp. 341–350, Oct. 2020, doi: 10.1007/s13748-020-00217-z.

[3] S. Nandi, H.A. Toliyat, and X. Li, "Condition Monitoring and Fault Diagnosis of Electrical Motors—A Review," *IEEE Trans. Energy Convers.*, vol. 20, no. 4, pp. 719–729, Dec. 2005, doi: 10.1109/TEC.2005.847955.

[4] J. Vargas-Machuca, F. García, and A. M. Coronado, "Detailed Comparison of Methods for Classifying Bearing Failures Using Noisy Measurements," *J. Failure Anal. Prev.*, vol. 20, no. 3, pp. 744–754, May 2020, doi: 10.1007/s11668-020-00872-3.

[5] S. Zhang, S. Zhang, B. Wang, and T. G. Habetler, "Deep Learning Algorithms for Bearing Fault Diagnostics—A Comprehensive Review," *IEEE Access*, vol. 8, pp. 29857–29881, 2020, doi: 10.1109/access.2020.2972859.

[6] W. Olarte, M. Botero, and B. A. Cañon, "Análisis de Vibraciones: Una Herramienta Clave en el Mantenimiento Predictivo," *Scientia Et Technica*, vol. 16, no. 45, pp. 219–222, 2010. [Online]. Available: https://www.redalyc.org/articulo.oa?id=84917249040.

[7] W. Qian, S. Li, and X. Jiang, "Deep transfer network for rotating machine fault analysis," *Pattern Recognit.*, vol. 96, Art. no. 106993, Dec. 2019, doi: 10.1016/j.patcog.2019.106993.

[8] M. H. M. Ghazali, and W. Rahiman, "Vibration Analysis for Machine Monitoring and Diagnosis: A Systematic Review," *Shock Vib.*, vol. 2021, no. 1, Art. no. 9469318, 2021, doi: 10.1155/2021/9469318.

[9] D.-T. Hoang and H.-J. Kang, "Rolling element bearing fault diagnosis using convolutional neural network and vibration image," *Cogn. Syst. Res.*, vol. 53, pp. 42–50, Mar. 2018, doi: 10.1016/j.cogsys.2018.03.002.

[10] D. H. Carvalho, D. P. Viana, A. A. Lima, M. F. Pinto, L. Tarrataca, F. L. Silva, R. H. Ramirez, T. Moura, U. A. Barbosa, and D. Barreto, "Diagnostic and severity analysis of combined failures composed by imbalance and misalignment in rotating machines," *Int. J. Adv. Manuf. Technol.*, vol. 114, pp. 3077–3092, Apr. 2021, doi: 10.1007/s00170-021-06873-2.

[11] A. Khorram, M. Khalooei, and M. Rezghi, "End-to-end CNN + LSTM deep learning approach for bearing fault diagnosis," *Appl. Intell.*, vol. 51, pp. 736–751, Aug. 2020, doi: 10.1007/s10489-020-01859-1.

[12] V. K, A. Rajakannu, M. Kamarudden, R. KP, and S. R. A V, "Intelligent Fault Diagnosis of Rotating Machinery Using Deep Learning Algorithms: A Comparative Analysis of MLP, CNN, RNN, and LSTM," *Int. J. Electr. Electron. Eng.*, vol. 11, no. 9, pp. 294–315, Sep. 2024, doi: 10.14445/23488379/ijeee-v11i9p127.

[13] S. Gao, L. Xu, Y. Zhang, and Z. Pei, "Rolling bearing fault diagnosis based on SSA optimized self-adaptive DBN," *ISA Trans.*, vol. 128, pp. 485–502, Sep. 2022, doi: 10.1016/j.isatra.2021.11.024.

[14] M. A. Jamil and S. Khanam, "Fault Classification of Rolling Element Bearing in Machine Learning Domain," *Int. J. Acoust. Vib.*, vol. 27, no. 2, pp. 77–90, Jun. 2022, doi: 10.20855/ijav.2022.27.21829.

[15] A. F. Cotrino, J. A. López, J. C. Blandón, and A. Toro Lazo, "Low-cost prototype for bearing failure detection using Tiny ML through vibration analysis," *HardwareX*, vol. 22, pp. e00658–e00658, May 2025, doi: 10.1016/j.ohx.2025.e00658.

[16] C. G. Northcutt, A. Athalye, and J. Mueller, "Pervasive label errors in test sets destabilize machine learning benchmarks," in *Proc. 35th Conf. Neural Inf. Process. Syst. (NeurIPS)*, Virtual Conf., Dec. 2021, pp. 1–15.

[17] S. Mohammed, L. Budach, M. Feuerpfeil, N. Ihde, A. Nathansen, N. Noack, H. Patzlaff, F. Naumann, and H. Harmouch, "The effects of data quality on machine learning performance on tabular data," *Inf. Syst.*, vol. 132, Art. no. 102549, Mar. 2025, doi: 10.1016/j.is.2025.102549.

[18] R. Liu, B. Yang, E. Zio, and X. Chen, "Artificial intelligence for fault diagnosis of rotating machinery: A review," *Mech. Syst. Signal Process.*, vol. 108, pp. 33–47, Aug. 2018, doi: 10.1016/j.ymssp.2018.02.016.

[19] Y. Chen, B. Zheng, Z. Zhang, Q. Wang, C. Shen, and Q. Zhang, "Deep Learning on Mobile and Embedded Devices," *ACM Comput. Surv.*, vol. 53, no. 4, pp. 1–37, Aug. 2020, doi: 10.1145/3398209.

[20] Edge Impulse, "Edge Impulse," [Online]. Available: https://edgeimpulse.com.

[21] N. M. Foumani, L. Miller, C. W. Tan, G. I. Webb, G. Forestier, and M. Salehi, "Deep Learning for Time Series Classification and Extrinsic Regression: A Current Survey," *ACM Comput. Surv.*, vol. 56, no. 9, Art. no. 217, pp. 1–45, Apr. 2024, doi: 10.1145/3649448.

[22] J. Chen and X. Ran, "Deep Learning with Edge Computing: A Review," *Proc. IEEE*, vol. 107, no. 8, pp. 1655–1674, Jul. 2019, doi: 10.1109/jproc.2019.2921977.

[23] W. Roth, G. Schindler, B. Klein, R. Peharz, S. Tschiatschek, H. Fröning, F. Pernkopf, and Z. Ghahramani, "Resource-Efficient Neural Networks for Embedded Systems," *J. Mach. Learn. Res.*, vol. 25, no. 1, Art. no. 50, pp. 2506-2556, Jan. 2024, doi: 10.5555/3722577.3722627.

[24] A. A. Lakis, "Rotating machinery fault diagnosis using time-frequency methods," in *Proc. 7th WSEAS Int. Conf. Elect. Pow. Syst., High Volt., Elect. Mach.*, Venice, Italy, Nov. 2007, pp. 139–144.

[25] Dynamox, "Análisis espectral: cómo aplicarlo para analizar las vibraciones en activos industriales," 2024. [Online]. Available: https://dynamox.net/es/blog/analisis-espectral-como-aplicarlo-para-analizar-las-vibraciones-en-activos-industriales.

[26] Edge Impulse (s.f) "Using the Edge Impulse Python SDK with TensorFlow and Keras," [Online]. Available: https://docs.edgeimpulse.com/docs/tutorials/ml-and-data-engineering/ei-python-sdk/python-sdk-with-tf-keras.

A. F. Cotrino Herrera received the B.S. degree in mechatronics engineering from Universidad Autónoma de Occidente, Colombia, in 2025. He is currently pursuing the M.S. degree in electrical engineering at Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), Brazil, where he also works as a part-time researcher at the Laboratório de Inteligência Computacional Aplicada (ICA). His current research interests include artificial intelligence, robotics, embedded systems and decision-support methods.

J. A. López Sotelo has over 25 years of experience in teaching and developing Artificial Intelligence-related projects. He is currently affiliated with Universidad Autónoma de Occidente in Cali, Colombia. He has published several articles, book chapters, and books focusing on Artificial Neural Networks, Deep Learning, and other Artificial Intelligence techniques. He has also been an invited speaker at national and international conferences to discuss the technical and social aspects of AI. His research interests include Artificial Neural Networks and Deep Learning, Artificial Intelligence on edge devices (Edge AI), the teaching of Artificial Intelligence, and the impact this technology may have on society. Prof. López is a professional senior member of the IEEE, where he belongs to the Colombian chapter of the Computational Intelligence Society.

A. Toro Lazo received the systems and telecommunications engineer degree from Universidad Católica de Pereira, Colombia. He obtained the M.S. degree in software project management and development from Universidad Autónoma de Manizales, Colombia, and the Ph.D. degree in big data management from the University of Salerno, Italy. He is currently a Full-time Assistant Professor in the Faculty of Basic Sciences and Engineering at Universidad Católica de Pereira, Colombia, and the Leader of the research group Entre Ciencia e Ingeniería at the same university. He is the author of over 30 indexed journal articles. His main research areas include software engineering, software quality assurance (SQA), automated testing, big data, and artificial intelligence and data analytics applied to industry 4.0 technologies (industrial internet of things – IIoT and cyber-physical systems – CPS). Prof. Toro is a member of international academic committees, including CICCSI (Argentina) and the PMI Italian chapter.

J. C. Blandón Andrade received the title of systems engineer, and subsequently the title of Specialist in university teaching. He obtained the M.S. degree in engineering with an emphasis on systems and computing from Universidad Javeriana, Cali, Colombia, and the Ph.D. degree in systems and informatics engineering from Universidad Nacional de Colombia, Medellín. He is currently a Full-time Associate Professor at Universidad Católica de Pereira, Colombia. He has developed tools for classifying citizen communications and managing information in productive sectors, integrating artificial intelligence methods with real needs. He has extensive experience in natural language processing (NLP), computational ontologies, and text automation in Spanish. His research interests include artificial intelligence applied to natural language processing (NLP), software engineering, and pedagogy in engineering.