2023 20th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE) | 979-8-3503-0676-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/CCE60043.2023.10332881

Bearing Vibrations Classification for Failure Detection with Machine Learning Tools

Luis Elias Salgado Solano
Electrical Engineering Department, CINVESTAV-IPN, Mexico City, Mexico
luis.salgado.s@cinvestav.mx

Oliverio Arellano Cárdenas
Electrical Engineering Department, CINVESTAV-IPN, Mexico City, Mexico
arellano@cinvestav.mx

Luis Martín Flores Nava
Electrical Engineering Department, CINVESTAV-IPN, Mexico City, Mexico
lmflores@cinvestav.mx

Felipe Gómez Castañeda
Electrical Engineering Department, CINVESTAV-IPN, Mexico City, Mexico
fgomez@cinvestav.mx

José Antonio Moreno Cadenas
Electrical Engineering Department, CINVESTAV-IPN, Mexico City, Mexico
jmoreno@cinvestav.mx

Abstract— **In this work, we have utilized Artificial Intelligence for the extraction and recognition of mechanical vibration information, which was obtained by electronic instruments, to identify whether a mechanical device is healthy or presents some kind of failure. We focused on building a Deep Autoencoder architecture (an unsupervised architecture), training the model, and extracting the internal structural information, also known as clustering. Lastly, the compressed information (as most Autoencoders have a reduced input representation in the internal structure) is classified into two classes using the supervised Extreme Learning Machine model.**

We utilized a dataset provided by Case Western Reserve University on its website. The signals were measured by sampling an accelerometer at 12kS/s, radially placed on a bearing. The measurements were taken for four cases, namely: the first case encompassed vibrational signals from a healthy bearing, and the other three cases involved bearings with intentionally induced failures in the inner race, outer race, and the ball, respectively.

We developed this intelligent system using Python within the Jupyter environment, which operates in conjunction with the TensorFlow and Keras frameworks. We also successfully trained the model.

Keywords— Deep Autoencoder, Extreme Learning Machine, bearing failures, Artificial Intelligence.

I. Introduction

It is estimated that each year, 10 billion bearings are manufactured globally. Out of these, 90% are installed in machinery or equipment where they remain throughout their lifespan. Around 9.5% are replaced as a preventive measure, while 0.5% are replaced due to corrective reasons [1].

The monitoring of the mechanical health of rotating components has been extensively investigated using various approaches. These approaches encompass experimental processes that can range from the straightforward utilization of accelerometers or ultrasonic sensors to more meticulous methods involving microscopic analysis [2].

Mechanical vibration signals have conventionally been examined through mathematical models and Gaussian analysis in various domains, including seismic signals and mechanical health monitoring. In recent times, emerging trends in computing have significantly enhanced the efficiency and adaptability of Artificial Intelligence algorithms, which can be applied to address various human requirements. Within the realms of signal processing and pattern recognition through Artificial Intelligence, information is approached from a cognitive standpoint, resulting in a computational cost that is lower than that of traditional signal processing methods.

There are two methods for extracting vibrational information from waveforms: amplitude demodulation and peak value detection. The latter method was pioneered by the Emerson Reliability Solutions© group. These approaches involve signal processing across various stages.

The aim of this study is to develop an efficient and cost-effective system utilizing a deep autoencoder architecture and a classifier. This system incorporates an Extreme Learning Machine model to ascertain the health condition of a rotating component based on its vibrational patterns. Furthermore, it is practical for implementation on hardware platforms like FPGAs or Raspberry systems. Such hardware could find utility across diverse environments, offering a diagnostic tool that circumvents the necessity for intricate analysis methodologies or advanced measurement instruments. This feasibility owes itself to machine learning techniques, which possess the capability to extract information and produce models suitable for deployment on a single-board computer.

II. Dimensionnality reduction

Dimensionality reduction is a process that reduces the quantity of random variables being examined, which holds significance in the analysis of extensive datasets. Some characteristics associated with this technique include:

- It prevents performance degradation when dealing with high-dimensional data, as efficiency and accuracy tend to deteriorate rapidly with increasing dimensions.
- It demands minimal computational resources.
- It aids in mitigating overfitting.

With the advent of artificial neural networks, an architecture known as the Autoencoder was introduced. This architecture can

This work was funded by CONAHCyT in the M.Sc. studies program for L. E. Salgado Solano and presented as proof of a Social Retribution Task.

effectively carry out dimensionality reduction tasks and has been swiftly embraced owing to its remarkable flexibility. [4].

A. *Basic Autoencoder*

The architecture of a basic autoencoder comprises an input layer, a hidden layer, and an output layer. This structure is depicted in Fig. 1.

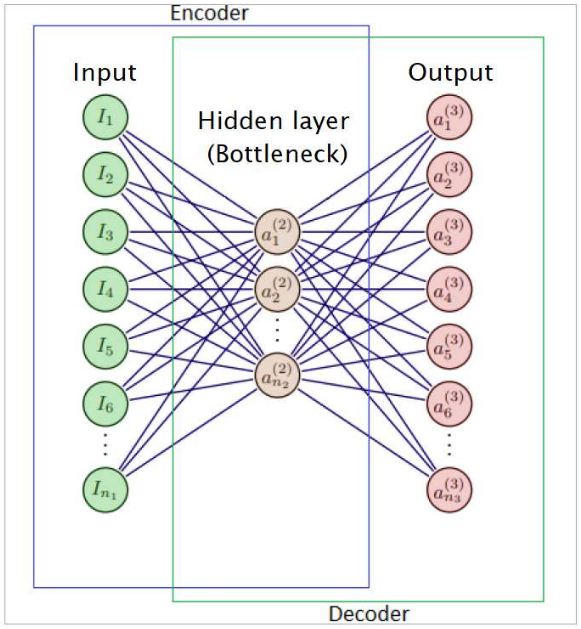

Fig. 1. Simple autoencoder.

The equation that describes this architecture is as follows:

$$\hat{x} = g\big(f(x)\big) \quad (1)$$

Where *f(x)* represents the vector function for feature extraction, also referred to as the hidden representation $\boldsymbol{h}$, and is described by:

$$\boldsymbol{h} = f(x) = s_f(\boldsymbol{W}^1\boldsymbol{x} + \boldsymbol{b}_h): \mathbb{R}^n \to \mathbb{R}^p, n > p \quad (2)$$

Where s_f ( ) signifies the activation function utilized in the hidden layer, $\boldsymbol{W}^1$ represents the synaptic weight matrix linking the input to the hidden layer, $\boldsymbol{x}$ denotes the input vector, $\boldsymbol{b}_h$ stands for the bias vector of the hidden layer, n denotes the input size, and p signifies the number of neurons in the hidden layer. $\hat{x}$ represents the input reconstruction produced by the decoding function $g(f(x))$, and their expressions are interconnected as shown in equation (3).

$$\hat{x} = g(h) = s_g\big(\boldsymbol{W}^2\boldsymbol{h} + \boldsymbol{b}_y\big): \mathbb{R}^p \to \mathbb{R}^n \quad (3)$$

Where s_g ( ) denotes the activation function applied in the output layer, $\boldsymbol{W}^2$ represents the synaptic weight matrix linking the hidden layer to the output layer, $\boldsymbol{h}$ signifies the hidden representation of $\boldsymbol{x}$, and $\boldsymbol{b}_y$ stands for the bias vector of the output layer.

B. *Deep Autoencoder*

A deep autoencoder consists of two or more hidden layers. This architecture offers the advantage of achieving enhanced output reconstruction and improved feature extraction. This is attributed to the hierarchical transfer of learning that transpires among the hidden layers. The overall structure is illustrated in Fig. 2.

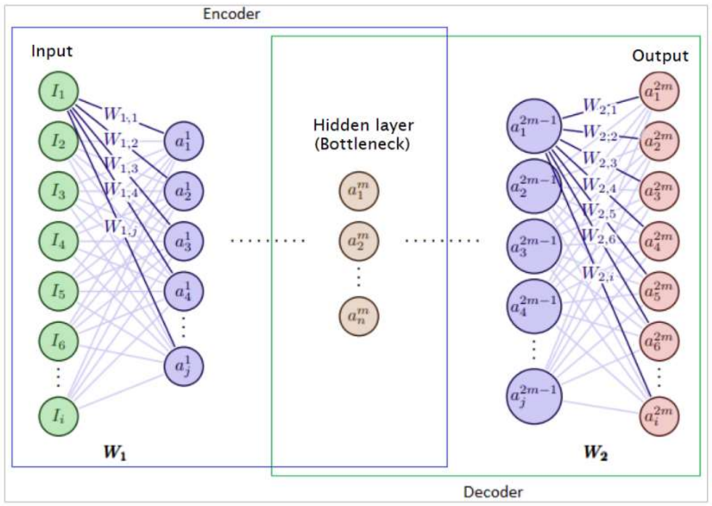

Fig. 2. Deep autoencoder.

A stacked autoencoder can be conceptualized as an array of sequentially linked autoencoders, where each layer serves as the hidden layer of a preceding autoencoder and concurrently functions as the input layer of a subsequent autoencoder.

Equations (4) to (7) formulate an autoencoder with n hidden layers, executing a dimensional transformation of inputs from $\mathbb{R}^{H_{in}} \to \mathbb{R}^{H_{out}}$, where $\mathbb{R}^{H_{in}}$ represents the input dimension and $\mathbb{R}^{H_{out}}$ signifies the output dimension. Equation (7) delineates the reconstruction process, where the input dimension aligns with the n-th layer and the output dimension matches that of the initial input data.

$$h_1 = \phi^1\big(\textstyle\sum_j w^1 x_j + b^1\big): \mathbb{R}^P \to \mathbb{R}^{H_1} \quad (4)$$

$$h_2 = \phi^2\big(\textstyle\sum_j w^2 h_{1,j} + b^2\big): \mathbb{R}^{H_1} \to \mathbb{R}^{H_2} \quad (5)$$

$$\vdots$$

$$h_n = \phi^n\big(\textstyle\sum_j w^n h_{n-1,j} + b^n\big): \mathbb{R}^{H_{n-1}} \to \mathbb{R}^{H_n} \quad (6)$$

$$y_i = \phi^{n+1}\big(\textstyle\sum_j w^{n+1} h_{n,j} + b^{n+1}\big): \mathbb{R}^{H_n} \to \mathbb{R}^P \quad (7)$$

Where $h_1 \ldots h_n$ is the representation of hidden layers, $\phi^1 \ldots \phi^n$, ϕ^{n+1} are the activation functions of layers, $w^1 \ldots w^n$, w^{n+1} are the synaptic weights of layers, $b^1 \ldots b^n$, b^{n+1} are the biases and y is the reconstructed output.

C. *Extreme Learning Machine model*

Extreme Learning Machine (ELM) is a machine learning algorithm designed for Single Layer Feedforward Neural Networks (SLFNs). It employs a random selection of hidden nodes and utilizes analytical methods to compute the output weights of the SLFNs through the solution of a linear system. This classification algorithm yields robust generalization performance while simultaneously maintaining an exceptionally rapid learning rate [5], [6].

Fig. 3 illustrates the ELM architecture, where it is evident that the input layer is linked to the intermediate layer via randomly generated weights, represented as w_i. Additionally, the intermediate layer connects to the output layer through weights denoted as β_i, which are determined using the Moore-Penrose pseudo-inverse matrix.

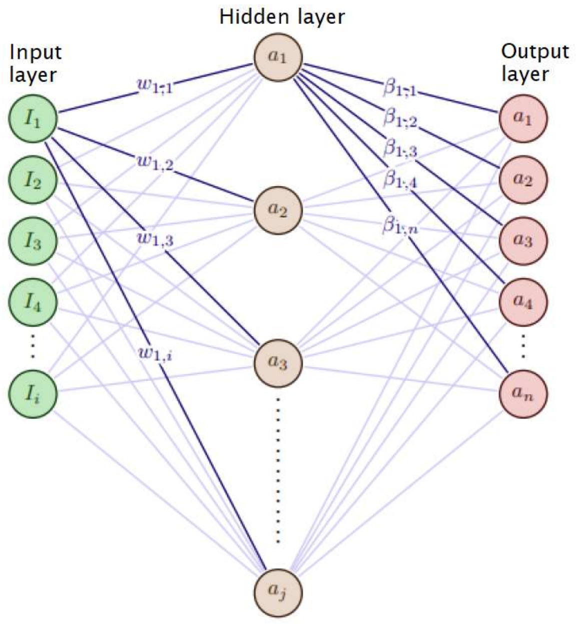

Fig. 3. ELM architecture, with *i* inputs, *j* hidden nodes, and *n* classes.

The equation for ELM architecture is defined as **H β = Y,** where **H** is the output of the hidden layer, **β** is the unknown weights matrix, and **Y** is the targets matrix:

$$\mathbf{H} = \begin{bmatrix} g(w_1^T x_1 + b_1) \cdots g(w_d^T x_1 + b_d) \\ \vdots \\ g(w_1^T x_N + b_1) \cdots g(w_d^T x_N + b_d) \end{bmatrix} \in \mathbb{R}^{Nxd} \quad (8)$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_N^T \end{bmatrix} = \begin{bmatrix} \beta_{11} \cdots \beta_{1m} \\ \vdots \\ \beta_{N1} \cdots \beta_{Nm} \end{bmatrix} \in \mathbb{R}^{Nxm} \quad (9)$$

$$\mathbf{Y} = \begin{bmatrix} y_1^T \\ \vdots \\ y_N^T \end{bmatrix} = \begin{bmatrix} y_{11} \cdots y_{1m} \\ \vdots \\ y_{N1} \cdots y_{Nm} \end{bmatrix} \in \mathbb{R}^{Nxm} \quad (10)$$

The solution for this system is defined as:

$$\boldsymbol{\beta} = \mathbf{H}^{\dagger}\,\mathbf{Y} \quad (11)$$

where $\mathbf{H}^{\dagger}$ is the generalized Moore-Penrose inverse.

III. Proposed Methodology

A. Dataset

To conduct our analysis, we employed a dataset accessible on the website of Case Western Reserve University [7]. This dataset encompasses a collection of files containing numerical data pertaining to bearing vibrations, recorded during both normal operating conditions and deliberately induced failure scenarios. Each file contains vibrational data corresponding to a distinct experimental condition. The files are categorized based on the specific region where the failure was induced, in addition to the diameter of the failure. The failures were deliberately introduced through mechanized electrical discharge at a single point, with diameters of 7 mil (0.1778 mm), 14 mil (0.3556 mm), and 21 mil (0.5334 mm).

The signals were acquired by sampling data from an accelerometer, positioned radially within a bearing, at a frequency of 12,000 samples per second. Our dataset comprised four distinct cases, encompassing a vibrational signal obtained from a healthy bearing, as well as signals stemming from bearings with deliberately induced failures in the inner race, outer race, and ball, respectively.

B. Calculation of vector size

To determine the appropriate length of each training vector input for the model, two criteria need to be considered [8]. Firstly, the input should have the smallest possible dimension to conserve model resources. Secondly, it should include the maximum amount of information to allow the network to extract characteristic biases while minimizing information loss. With these criteria in mind, a tool was employed to visualize the frequency spectrum of the signals in the database. This aimed to identify the frequency ranges that contain the most significant information. To accomplish this, a Python program was employed. It extracted 1000 samples from four vectors within the database and then applied the discrete Fourier transform. The resulting graph is depicted in Fig. 4.

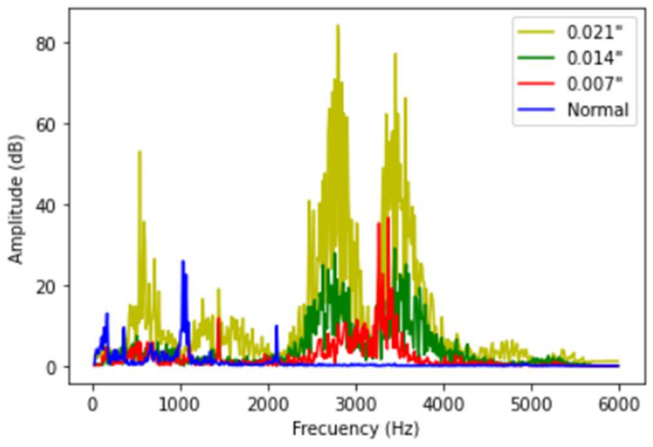

Fig. 4. Frequency spectra of Normal, 0.007", 0.014", and 0.021" databases.

Drawing from the insights conveyed by Fig. 4, it was ascertained that a vector comprising 100 samples corresponds to a period of 83.3μs when subjected to a sampling frequency of 12kHz. Consequently, this yields a vector frequency of 120Hz. Such a configuration adequately encompasses high-frequency information existing beyond the 1000Hz mark.

C. Methodology

The entire procedure undertaken by our proposed system is illustrated in Fig. 5. This diagram encapsulates the full methodology, encompassing the training of the Case Western Reserve University database, signal conditioning as detailed in Table I, dimensionality reduction through a deep autoencoder, and the construction of the classifier using the Extreme Learning Machine algorithm (ELM). Each of these stages was simulated using the Python language within the TensorFlow platform and leveraged the Keras libraries for machine learning purposes.

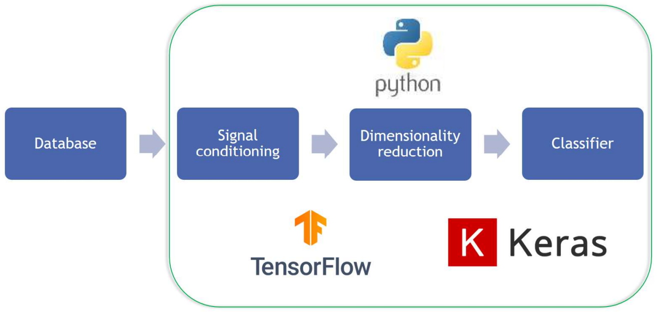

Fig. 5. Proposed methodology.

The dataset comprises single-row vectors, each containing over 100 x 10^3 samples. To facilitate the training process, it was imperative to disintegrate these vectors into arrays of 100 sample vectors, as demonstrated in Table I, for the scenario of 0.007 inches.

TABLE I. TRAINING VECTORS AND SAMPLES

File (0.007")	Samples number	Vectors number
Normal	243938	2439
Failure in inner race	121265	1212
Failure in outer race	121991	1219
Failure in ball	122571	1225

The deep autoencoder utilized for dimensionality reduction comprises the subsequent layers: an input layer with 100 neurons and a linear activation function, followed by hidden layers consisting of 31, 10, and 3 neurons, respectively, utilizing the rectified linear activation function (ReLU). Finally, the output layer with 100 neurons employs the hyperbolic tangent as its activation function. This configuration of the autoencoder is depicted in Fig. 6.

In order to establish a comprehensive measurement methodology, the encoder section was seamlessly integrated into the classifier, enabling predictions to be made in a single step. Figure 7 illustrates the proposed integrated methodology. This approach involves the encoder's role in compressing the information and generating a three-characteristic representation of the input. Subsequently, the ELM architecture leverages the output information from the three encoder neurons to predict their corresponding classes.

IV. EXPERIMENTS AND RESULTS

The model underwent training using files containing normal samples as well as failures of sizes 0.007", 0.014", and 0.021". In the initial phase, the encoder was trained to perform signal reconstruction. As illustrated in Fig. 8, the reconstruction for the 0.007" case might not be completely precise, although the results are reasonably coherent. The reconstruction exhibits a level of differentiation between the datasets, and yet it is viable to achieve effective clustering of the data. The core concept is centered around reducing the vectors from a dimension of 100 down to just 3 parameters. These parameters can be graphically represented in a three-dimensional plot, as portrayed in Fig. 9. They subsequently serve as inputs for the classifier.

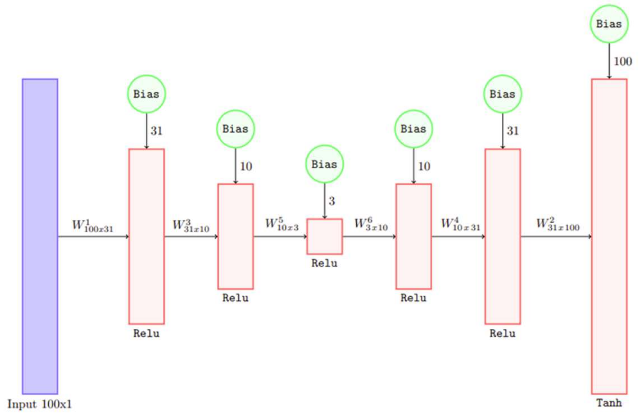

Fig. 6. Deep autoencoder model.

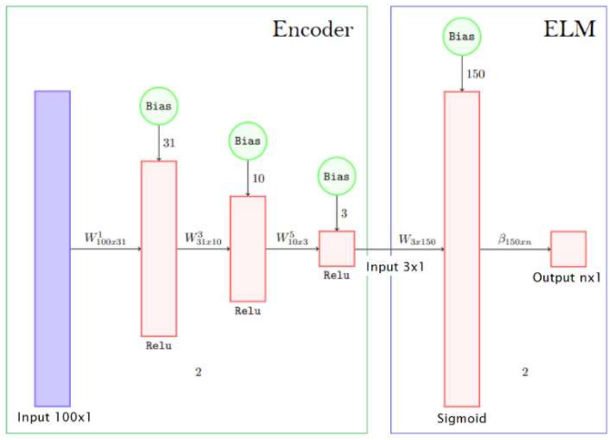

Fig. 7. Integral architecture, including the encoder and ELM stages.

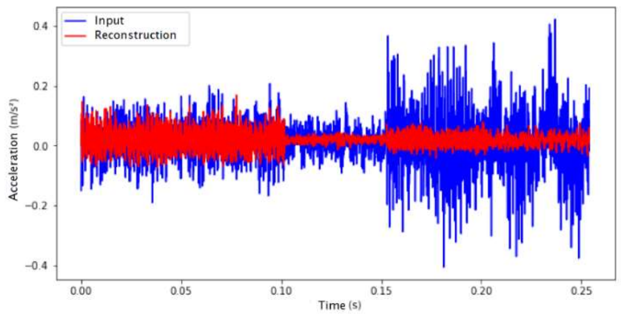

Fig. 8. Original signal (blue) and reconstruction (red) for 0.007" failures.

The ELM architecture was trained using the set of 3-parameter vectors obtained from the encoder. To assess the performance of the integrated architecture, figures 10, 11, and 12 depict the confusion matrices generated by the ELM architecture for both the healthy and failure classes of sizes 0.007", 0.014", and 0.021", respectively.

Based on the information provided from the confusion matrices, it's evident that the accuracy ranges from 80% to 99%. These results are notably improved compared to the performance for four classes. Importantly, in the context of detecting failures in bearings, the specific type of failure might not be as critical as simply determining the presence of a failure.

In this regard, your system appears to be performing effectively for this purpose.

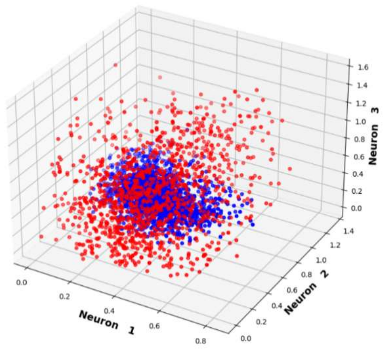

Fig. 9. Graph of the sets discerned by the encoder for 0.007" failures.

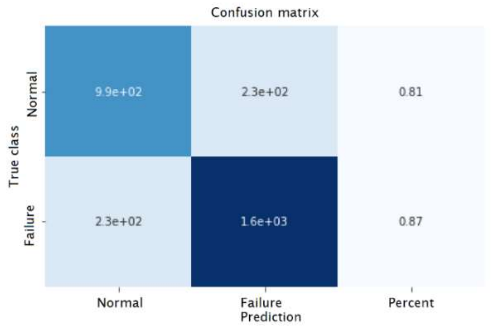

Fig. 10. Confusion matrix for two classes and failure of 0.007".

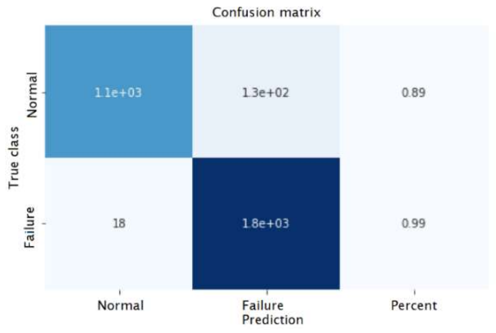

Fig. 11. Confusion matrix for two classes and failure of 0.014".

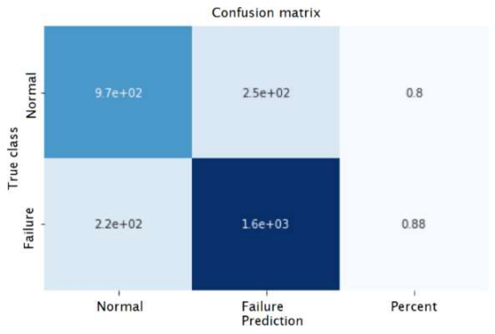

Fig. 12. Confusion matrix for two classes and failure of 0.021".

V. DISCUSSION

Concerning the accuracy of the autoencoder namely, the reconstruction error, which is not good, it is enough in the dimensionality reduction task as a source of data for the ELM classifier. Fig. 9 shows in a visual manner the separation between good and bad bearings, leading to acceptable classification values as presented in Figs. 10-12. The reduction of dimension to 3 was chosen because it allows visualizing the clustering of input data, otherwise, this analysis becomes cumbersome.

VI. CONCLUSIONS

In summary, the employment of autoencoders can significantly decrease the dimensionality of vector parameters within the data, thereby simplifying problems that involve a large number of features. This simplification in turn enables the implementation of the system on platforms like FPGAs or Raspberry systems.

Machine learning techniques are highly appropriate for processing extensive amounts of information, making them valuable tools for extracting insights from Big Data. The fusion of a decoder stage and an ELM classifier yielded a unified measurement architecture. This integrated architecture was successfully simulated and exhibited satisfactory classification outcomes.

REFERENCES

[1] Daño de rodamientos y análisis de fallas. Grupo SKF 2017. PUB BU/I3 17186 ES · Febrero 2017

[2] J. Halme y P. Andersson, "Rolling contact fatigue and wear fundamentals for rolling bearing diagnostics - State of the art", Proc. Inst. Mech. Eng. Part J J. Eng. Tribol. vol. 224, n. 4, pp. 377-393, 2010.

[3] Emerson. AMS 2140 Machinery Health Analyzer. Technical report +1 865 675 2400. 835 Innovation Drive Knoxville, TN 37932 USA: Emerson Reliability Solutions, dic. de 2017.

[4] Quentin Fournier y Daniel Aloise. "Empirical Comparison between Autoencoders and Traditional Dimensionality Reduction Methods". 2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE). IEEE, June 2019. doi: 10.1109/aike.2019.00044. url: https://doi.org/10.1109/aike.2019.00044..

[5] Xian-Da Zhang. A Matrix Algebra Approach to Artificial Intelligence. Springer Singapore, 2020. doi 10.1007/978-981-15-2770-8. url: https://doi.org/10.1007/978-981-15-2770-8.

[6] Zhou, Hongming, "Extreme learning machine for classification and regression", Doctoral thesis, Nanyang Technological University, Singapore. 2014. Download URL: https://hdl.handle.net/10356/61529, https://doi.org/10.32657/10356/61529.

[7] Case Western Reserve University. Bearing Data Center. Available at https://engineering.case.edu/bearingdatacenter/12k-drive-end-bearing-fault-data.

[8] Aston Zhang et al. Dive into Deep Learning. Cambridge University Press. ISBN-10:1009389432, ISBN-13: 978-1009389433, arXiv:2106.11342, 2021.