DOI: 10.53375/ijecer.2023.319

Implementing Deep Learning Models in Embedded Systems for Diagnosis Induction Machine

Kevin Barrera-Llanga*, Ángel Sapena-Bañó, Javier Martínez-Román, Ruben Puche-Panadero
Institute for Energy Engineering, Universitat Politècnica de València, Spain,
Email: kebarlla@upv.edu.es, asapena@die.upv.es, jmroman@die.upv.es, rupucpa@die.upv.es
**Corresponding author*

Abstract: **Diagnosis of induction machines based on deep learning models is becoming a trend in maintenance systems of modern industry. The implementation of such systems allows low-cost industrial monitoring due to the characteristics of hardware and software. However, combining deep learning models with current embedded systems is a difficult issue due to the computational cost to run data through neural networks. In particular, balancing the resources of graphics processing unit (GPU) with modern architecture models is a challenge for prototyping induction machine diagnostic systems. In this work, a novel method is proposed to implement trained deep learning models in low-cost embedded systems. We review algorithmic optimizations for running neural network models, balancing computational resources. In addition, we propose the use of a graphical user interface (GUI) as a tool for the end user of the embedded system. The approach proposed in this work can be of great help to future researchers who develop their prototype using deep learning models and low-cost embedded systems.**

Keywords: **Deep learning, Implementation of automatic models, Induction Machine, Embedded systems**

I. INTRODUCTION

Since the industrial revolution, induction machines have been widely used due to their durability, low implementation cost, easy maintenance, and efficient use of energy [1]. The operation of the machines is susceptible to the health of the induction motor (IM). Studies show that 90% of machine failures occur in the IM [2,3], and these sudden failures directly affect the production line. Therefore, fault diagnosis in induction machines is a relevant research topic that benefits maintenance costs in the production line.

The impact of AI (artificial intelligence) in fault diagnosis problems increasingly boosts to practical and sustainable implementation. In industry 4.0, automatic diagnostic systems are necessary for predictive maintenance. The models for the diagnosis are properly designed as a tool where the final decision making will continue to be centered on the human being [4] and its practical implementation is focused on the production line. The models are continuously trained in a learning process in which features are extracted until the model has an acceptable prediction [5]. This process demands a high computational cost in the graphics processing unit (GPU) to learn and predict large data sets. As the complexity and performance of the architectures increase in the latest generation models, the computational cost increases rapidly. Most contribution in the technical literature describe the design, development, and testing of AI models. However, the implementation of the trained model in a prototype for a practical use is a little discussed topic, which is addressed in this paper.

At the prototyping stage, a trained model should be implemented in a user friendly way. This implementation can be expensive since it is required to invest additional resources to have a better performance of the model [6]. From an industrial point of view, investing in expensive systems to implement deep learning models would raise the cost on the production line.

To achieve the required computational cost and to reduce equipment prices, the main hardware vendors design low-power embedded cards for deep learning [7]. Among these, Nvidia® proposes its Jetson developer card as a small, lightweight, and low power consumption embedded system compatible with AI architectures [8]. The use of embedded systems for the diagnosis of induction machines is a novel proposal that can be implemented easily using modern, state of the art devices.

The contribution of this work is the practical implementation of a previously trained deep learning model in an embedded system for the end user from a Jetson developer card. We approach the application of models with an efficient use of the GPU and an adequate distribution of computational resources. In addition, we propose a graphical user interface (GUI) system that serves as a monitoring tool and information manager for the diagnosis of induction machines.

The structure of this work is as follows: in Section II related jobs, background and an overview of the work are presented. In Section III we explain our method to implement the deep learning models in the embedded system. In Section IV we indicate the results obtained in the diagnosis of

This work was supported in part by the Spanish Ministerio de Ciencia, Innovación y Universidades (MCIU), in part by the Agencia Estatal de Investigación (AEI), and in part by the Fondo Europeo de Desarrollo Regional (FEDER) in the framework of the Proyectos I+D+i - Retos Investigación 2018, under Project Reference RTI2018-102175-B-I00 (MCIU/AEI/FEDER, UE)

induction machines. In Section V we present our conclusion and future steps of the investigation.

II. BACKGROUND AND OVERVIEW

In modern industry, models are used as tools that help human beings in decision tasks. The internet of things (IoT) is the main link for sharing signals between machines and devices in an industrial environment. Using AI models in combination with IoT is the axis of Industry 4.0. In [9] the authors propose the use of deep learning models to reduce industrial energy consumption, by controling air conditioners in the line of production and monitoring the number of people, using an application programming interfaces (API) of Microsoft. The work is focused from the perspective of design and development of the model without delving into the implementation of the final application.

Real-time data analysis with deep learning models is a technique used for fault detection. In [10] it is proposed to monitor data from a thermomechanical pulp mill with neural network models to predict potential equipment failures, using a GPU dedicated high-powered Nvidia GeForce GTX1070 for model development and testing.

The practical implementation for the end user of deep learning models is proposed in the work [11], running a convolutional neural network (CNN) trained with the database of numbers (MNIST) on an Intel Movidius Myriad2 card. They use the Tensorflow API to design a lightweight single-convolution CNN. Implementing models of mobile architectures in an embedded system is proposed by [12] using use a Xilinx Zynq card to execute models with mobilnet and inception architectures for real-time object detection. A review of deep learning on the Jetson card is discussed in [8], where the author describes the features, applications, and limitations of the card to run various CNN architectures. The author concludes that this card can work with complex architecture models with satisfactory performance.

The diagnosis of faults in induction machines using deep learning is extensively analyzed in the review proposed by [13], which analyzes existing works for the detection of bearing faults in induction machines, focusing on the development and testing of models. In our research group we focus on the development of diagnostic techniques for induction machines. In the work [14] we propose a simulated method to calculate the phase inductances in rotating electrical machines under fault conditions. Analyzing motor currents with non-invasive methods is increasingly used in maintenance systems. In [15] a novel method is proposed to analyze the hidden harmonics in the fundamental that show asymmetry faults of the rotor. Machine learning systems are tools that can be applied to detect faults in induction machines. At work [16], we use expert systems optimized for the analysis of current spectra when the machine is in transient conditions.

The objective of this paper is to indicate our method for the implementation of an AI prototype for the diagnosis of induction machines. We run a deep learning model previously trained with the pytorch API under the python language in an embedded system based on Nvidia's Jetson card with the Ubuntu kernel. We explain the design of a user-friendly GUI for the purpose of monitoring and predictive analysis of the model. From the computational point of view, we analyze the performance of the card and the optimization of the resources that influence the automatic diagnosis.

III. IMPLEMENTATION OF THE DEEP LEARNING MODEL

A. *Architecture exploration of embedded Systems*

Nvidia's Jetson is an embedded card from the Single Board Computer series focused on deep learning tasks. The Jetson family consists of a series of cards that differ from each other due to their technical characteristics. We propose to describe the implementation from the Jetson Nano card, being the smallest card in the Jetson family. Therefore, the description of the implementation will be compatible for the entire range of Jetson cards.

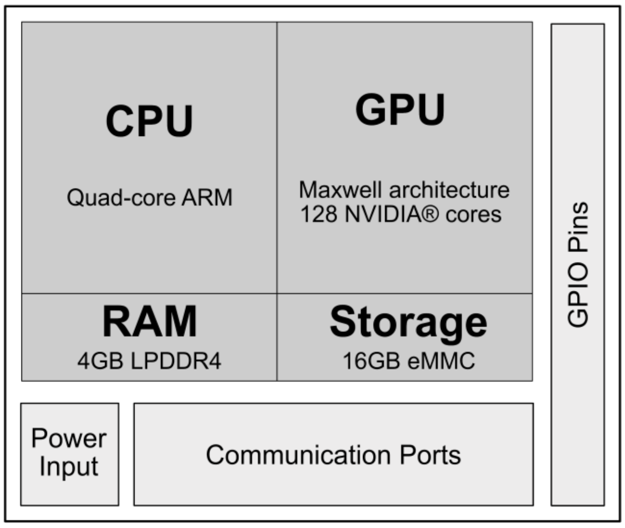


Fig. 1. Block diagram of the general architecture of the Jetson nano board.

The Jetson nano are embedded systems that have the memory, storage processor, input and output interface on the same board, see in Fig. 1. It has a hybrid CPU-GPU architecture, where the ARM CPU starts the Ubuntu 18.04 operating system, and the Nvidia Maxwell GPU processes the computational calculation of the Deep Learning models. The card supports CUDA (Compute Unified Device Architecture), allowing parallel processing with 128 cores to increase computing performance, CUDA compatibility is an important factor to consider in practical implementation. Most neural network applications are trained with GPUs that are coded with CUDA [17]. The card is supported by Nvidia, so it comes with a constantly updated developer software kit.

The use of card memory varies depending on the architecture of the deep learning model to be used [18]. Our choice has been the use of a card with the largest amount of RAM memory. To have an implementation margin, we used the 4GB RAM configuration. The power supply for the embedded system according to its manufacturer is 5V and 4-6A. The prediction process makes the GPU work continuously increasing the internal temperature of the embedded system, the prototype will be connected to the power supply all the time. Therefore, we consider using a 5V-4A power supply and an additional ventilation system to mitigate the temperature. The speed of the automatic diagnosis will not be affected due to the optimization explained in section III-C.

B. Deep learning model configuration

In this work we indicate the practical implementation of the deep learning model for the diagnosis of IM. Our model is a convolutional neural network (CNN), trained with images of frequency spectra of the supply current of an induction motor to predict four types of failures. The implementation scheme can be seen in Fig. 2.

In Fig. 2-B, we indicate the process that is carried out in the embedded system. We start by storing the current signals of the IM, these values are obtained by a clamp meter for 100 seconds with a sampling frequency of 20KHz. The clamp signal passes through an analog-to-digital converter to the GPIO ports of the Jetson Nano card [19].

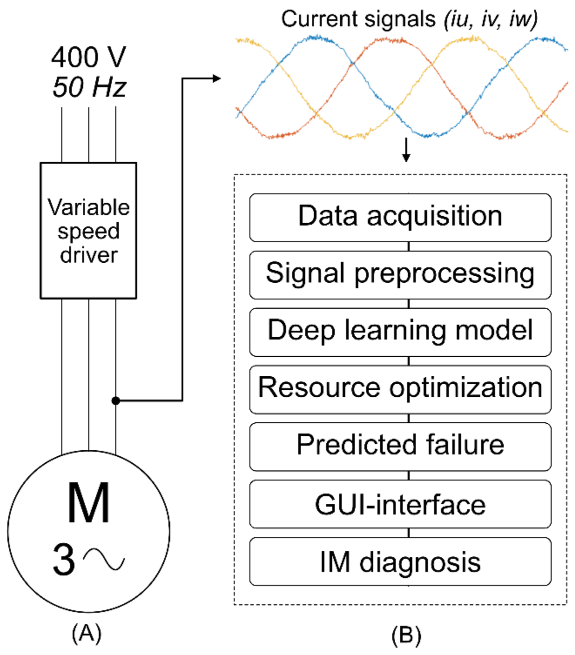


Fig. 2. Implementation diagram of the deep learning model in the embedded system. (A) Obtaining current signals. (B) Internal process in the embedded system for the diagnosis of induction machines.

On the first initialization of the embedded system, we proceed to update the Python package manager (PIP) that is installed by default. For our implementation, we used python version 3.6.9 due to stability and compatibility with the Jetson Nano card. We describe the packages and libraries installed in the embedded system for the implementation of the deep learning model in Table I.

Additionally, our research group has a database of tests on induction machines stored under Matlab coding. The compatibility of the prototype with the database is done through the SciPy library, in addition the library is used for mathematical algorithms, scientific and technical computing.

TABLE I. LIBRARIES USED FOR THE IMPLEMENTATION OF THE DEEP LEARNING MODEL IN THE EMBEDDED SYSTEM.

Library	Description	Version
Jetson stats	Monitoring and control of Jetson	3.1.3
Pytorch	Machine and deep learning development	1.8.0
Numpy	Numerical calculation	1.18.4
Pandas	Large volume data analysis	1.1.5
Streamlit	GUI creation from Python	1.8.1
Plotly	Data visualization in GUI	5.7.0
SciPy	Scientific and technical computing	1.5.4

C. Deep learning model configuration

We show the step diagram of the implementation in the embedded system in Fig. 3. We design two external functions using the Python language that connect to the GUI interface for the diagnosis of the induction machine. In the first function, the samples of the digital signal are entered. The stored signal has a length of 2x106 for each phase of the current. We split the signal every 10 seconds to decrease the length of the vector and to do a parallel calculation. We use the fast Fourier transform on the vector and store the images of the spectra in the internal memory of the embedded card, and then we use the SciPy and numpy libraries to perform the mathematical calculation.

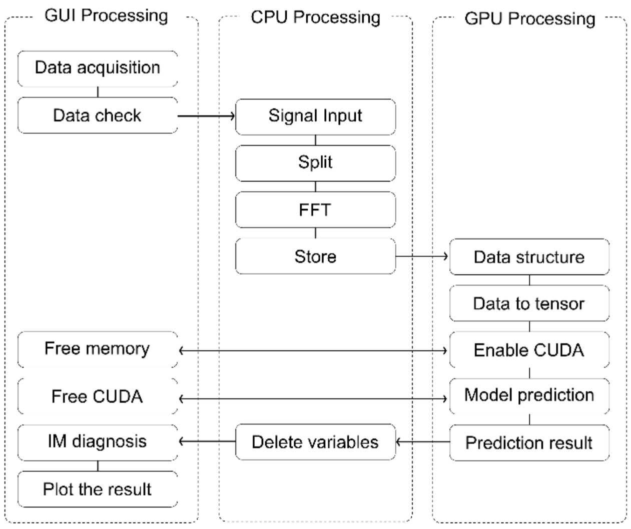


Fig. 3. Step diagram of the implementation in the embedded system. In the GUI the diagnosis of the induction machine is displayed, and we monitor the embedded system. Signal processing and model prediction is performed jointly by the CPU and the GPU.

The next step is to introduce the images into the deep learning model. Generally, the basic architecture of the CNN is made up of convolutional layers, color filters, fully connected layers, and probability functions [20]. At the input of a CNN an image enters and at the output we obtain a vector with the prediction probabilities [21]. The length of the vector will depend on the training labels [22]. In our case the length of the vector is four. CNN-based research provides the complexity and performance of architectures, which is why they bear a distinctive name [23], some of the most popular in the literature are EfficientNet [24], ResNet [25], VGG16-19 [26] and Inception [27]. Our model of induction machine diagnostic system is based on VGG19 with a performance of 98%. The second external function creates a data structure in the form of a list with the environment variable of the address of each spectrum image stored in internal memory. We use the Pandas library to create this data frame. We store the value of the temperature and initial memory of the embedded system using the Jetson stats package and load the trained CNN model with the load_learn function of the Pytorch library.

We convert the stored images into tensors, that is, into matrices with sets of numbers to introduce them into the model. The function learn.predict returns the vector of four positions with the probability corresponding to each class. We pass the tensor corresponding to the image of the current spectrum through learn.predict, in this process the CNN makes a prediction using the trained parameters. The computational calculation increases, to decrease the load on the GPU, we use CUDA's parallel computation with the torch.cuda.available function.

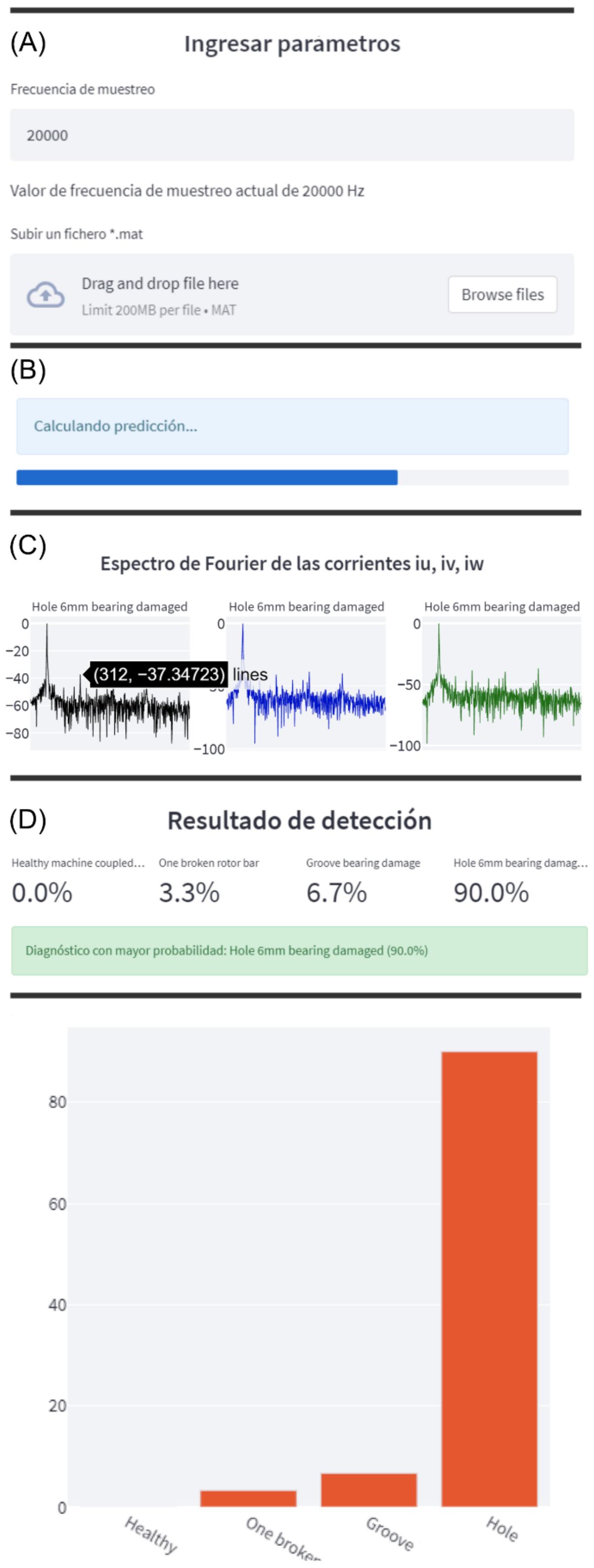
At the end of the first learn.predict we store the prediction vector, temperature and current memory. If the memory usage gradually increases, we call the garbage collector function [28] to release variables that are no longer going to be used, such as residual computations and the input tensor [29]. On the GPU we free the CUDA cache with the torch.cuda.empty_cache function. The temperature variable is sent through the GPIO ports to control the external ventilation of the system. This process is repeated with all the images of the spectrum, and at the end we group a second dataframe that contains the Fourier spectrum, its respective prediction, and the diagnosis with the highest probability.

To simplify the communication between the end user and the diagnostic system, we have developed a visual GUI environment using the Streamlit library. The graphical process of operation of the GUI is indicated in Fig. 4, the current data enters the GUI (see in the Fig. 4-A. The GUI calls the first function for signal preprocessing, visually represented by a progress bar (see in the Fig. 4-B). The GUI calls the second external function to perform diagnostics on the induction machine.

We use the values of the second dataframe to make interactive graphs with the plotly library. At the end of the diagnosis, in the GUI we can visually analyze the Fourier spectra with their diagnosis per signal (see in the Fig. 4-C). Additionally, we obtain a global plot of all prediction probabilities in the current signal (see in the Fig. 4-D).

IV. EVALUATION

To evaluate the implementation of the model in the embedded system, we propose to monitor the computational resources during the signal analysis of a 100 second. The prototype splits the signal every 10 seconds, with 10 data sets for each phase and a total of 30 data sets for the three-phase current signal. In the experiment, we monitor the amount of RAM memory, the availability of CPU, GPU, temperature, and power used by the embedded system when we use the deep learning model. The experiment is carried out in two parts: in the first part, we monitor the optimization of resources in the process of data acquisition and pre-processing of the current signal. In the second part, we monitor the optimization of the resources used by the deep learning model in the diagnosis of the induction machine.


(A) Ingresar parametros

Frecuencia de muestreo

20000

Valor de frecuencia de muestreo actual de 20000 Hz

Subir un fichero *.mat

Drag and drop file here
Limit 200MB per file • MAT
Browse files

(B)

Calculando predicción...

(C) Espectro de Fourier de las corrientes iu, iv, iw

(D) Resultado de detección

Healthy machine coupled...	One broken rotor bar	Groove bearing damage	Hole 6mm bearing damag...
0.0%	3.3%	6.7%	90.0%

Diagnóstico con mayor probabilidad: Hole 6mm bearing damaged (90.0%)

Fig. 4. GUI interface of the induction machine diagnostic system. (A) Acquisition of current signals, (B) preprocessing and prediction calculation, (C) Visual inspection of frequency spectra and (D) Global diagnosis of the current signal.

Initializing the operating system along with the GUI interface on the Jetson nano, we have initial values of 1.1 GB RAM occupied, 16% CPU, 0% GPU, 40 oC internal CPU and 38.5oC on the GPU.

In Fig. 5 we indicate the comparative graphs of the implementation in the prototype. In Fig. 5-A we visualize the operating frequency of the CPU and GPU of the experiment in the embedded system. The operating frequency of the CPU in the signal pre-processing stage and the prediction of the deep learning model have a similar behavior. On the GPU the variation is noticeable between the two stages.

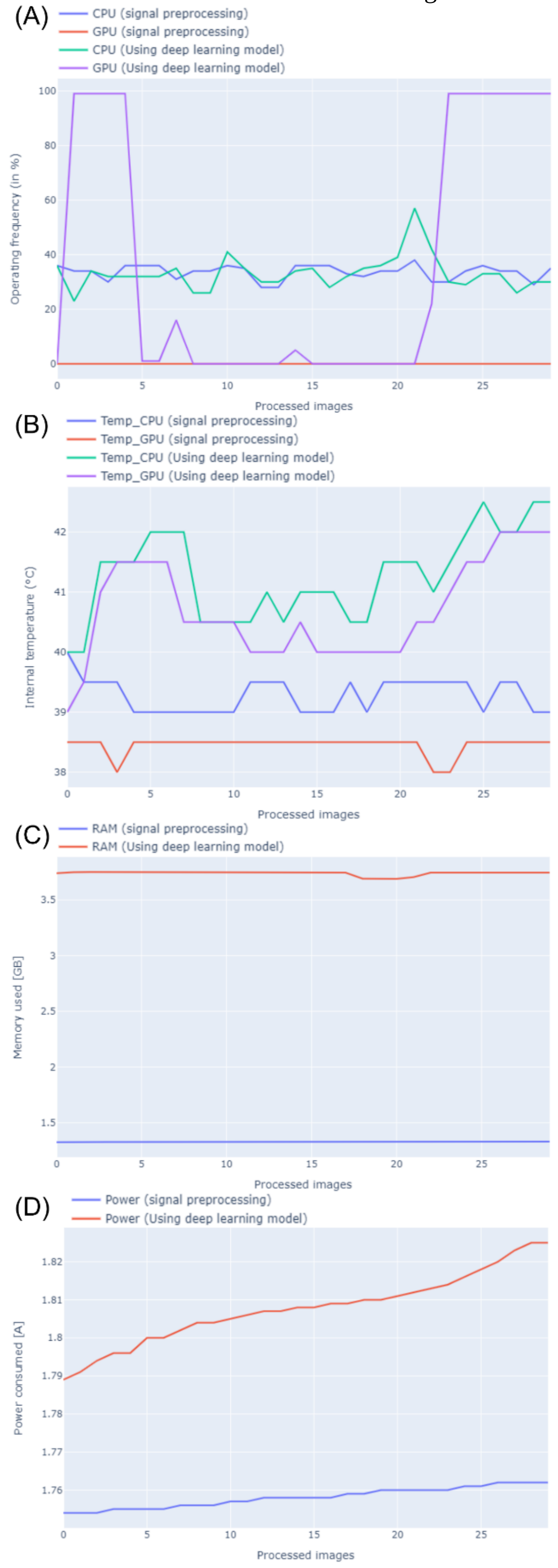


Fig. 5. Computational performance of the embedded system in the prediction of 30 Fourier spectra of the current of an induction motor. (A) Percentage of CPU and GPU operating frequency. (B) Internal temperature of the CPU or GPU. (C) RAM used in the diagnostic assay. (D) Power used by the embedded system in the diagnosis.

The GPU isn't used when performing signal pre-processing. When the deep learning model processes the images, the frequency on the GPU increased to 100%, at this point the optimization explained in the previous section acts to free up memory by reducing the GPU frequency values. In the GUI we use the GPU to display the diagnostic result, so the frequency trend increases at the end, when all predictions are finished the values return to their initial state.

In Fig. 5-B we monitor the internal temperature of the CPU and GPU during the test. At the signal pre-processing stage, both the CPU and GPU are kept in a normal range. In the process of prediction by the model, the temperature of both starts to increase, however, the temperature of the GPU is lower than that of the CPU, because the optimization of the prototype is applied to the GPU. In Fig. 5-C we observe the behavior of the RAM used during the test. The memory is kept at the initial value of the test, that is, it's used only by the operating system and the GUI interface. The calculations performed by the embedded system increase when we use the deep learning model, evidencing that the embedded system works with more complex deep learning architectures such as VGG19. The energy use seen in Fig. 5-D is reasonable when it increases the computational calculation and temperature. However, the current consumption doesn't exceed 2A, so it can be used in an industrial environment as a low-power device.

V. CONCLUSIONS AND FUTURE WORKS

In this paper, we have proposed an approach for the practical implementation of a fault diagnosis model of an induction machine using an embedded system. The efficiency obtained by our method allows the use of robust neural network architectures on the Jetson Nano board. The advantage of the presented method is to be a non-invasive method for automatic diagnosis using low-cost systems for monitoring and predictive maintenance. The results of our experiment indicate that, by using the proposed optimization, we reduce the computational load on the GPU of the embedded system. In future works, we will implement the diagnosis of induction machines using IoT and models with greater robustness according to failure analysis in IM.

REFERENCES

[1] M. Q. Tran, M. Elsisi, K. Mahmoud, M. K. Liu, M. Lehtonen and M. M. F. Darwish, “Experimental Setup for Online Fault Diagnosis of Induction Machines via Promising IoT and Machine Learning: Towards Industry 4.0 Empowerment,” *IEEE Access*, vol. 9, pp. 115429–115441, 2021. DOI: 10.1109/ACCESS.2021.3105297.

[2] P. Gangsar and R. Tiwari, “Signal based condition monitoring techniques for fault detection and diagnosis of induction motors: A state-of-the-art review,” *Mechanical Systems and Signal Processing*, vol. 144, p. 106908, Oct. 2020. DOI: 10.1016/J.YMSSP.2020.106908.

[3] R. Jigyasu, A. Sharma, L. Mathew and S. Chatterji, “A Review of Condition Monitoring and Fault Diagnosis Methods for Induction Motor,” *Proceedings of the 2nd International Conference on Intelligent Computing and Control Systems, ICICCS 2018*, pp. 1713–1721, 2019. DOI: 10.1109/ICCONS.2018.8662833.

[4] V. Terziyan and O. Vitko, “Explainable AI for Industry 4.0: Semantic Representation of Deep Learning Models,” *Procedia Computer Science*, vol. 200, pp. 216–226, 2022. DOI: 10.1016/J.PROCS.2022.01.220.

[5] A. Shrestha and A. Mahmood, “Review of deep learning algorithms and architectures,” *IEEE Access*, vol. 7, pp. 53040–53065, 2019. DOI: 10.1109/ACCESS.2019.2912200.

[6] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, “The Computational Limits of Deep Learning,” *ArXiv*, 2020. DOI: 10.48550/arxiv.2007.05558.

[7] T. Kloda, M. Solieri, R. Mancuso, N. Capodieci, P. Valente, and M. Bertogna, “Deterministic memory hierarchy and virtualization for

modern multi-core embedded systems," *Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS*, vol. 2019-April, pp. 1–14, 2019. DOI: 10.1109/RTAS.2019.00009.

[8] S. Mittal, "A Survey on optimized implementation of deep learning models on the NVIDIA Jetson platform," *Journal of Systems Architecture*, vol. 97, pp. 428–442, 2019. DOI: 10.1016/J.SYSARC.2019.01.011.

[9] M. Elsisi, M. Q. Tran, K. Mahmoud, M. Lehtonen, and M. M. F. Darwish, "Deep Learning-Based Industry 4.0 and Internet of Things towards Effective Energy Management for Smart Buildings," *Sensors*, vol. 21, no. 4, p. 1038, 2021. DOI: 10.3390/S21041038.

[10] M. Elhefnawy, A. Ragab, and M. S. Ouali, "Fault classification in the process industry using polygon generation and deep learning," *Journal of Intelligent Manufacturing*, no. 0123456789, 2021. DOI: 10.1007/s10845-021-01742-x.

[11] A. Kyriakos, E. A. Papatheofanous, B. Charalampos, E. Petrongonas, D. Soudris, and D. Reisis, "Design and Performance Comparison of CNN Accelerators Based on the Intel Movidius Myriad2 SoC and FPGA Embedded Prototype," *Proceedings - 2019 3rd International Conference on Control, Artificial Intelligence, Robotics and Optimization, ICCAIRO*, pp. 142–147, 2019. DOI: 10.1109/ICCAIRO47923.2019.00030.

[12] A. Sharma, V. Singh and A. Rani, "Implementation of CNN on Zynq based FPGA for Real-time Object Detection," *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1-7, 2019. DOI: 10.1109/ICCCNT45670.2019.8944792.

[13] D. Neupane and J. Seok, "Bearing fault detection and diagnosis using case western reserve university dataset with deep learning approaches: A review," *IEEE Access*, vol. 8, pp. 93155–93178, 2020. DOI: 10.1109/ACCESS.2020.2990528.

[14] A. Sapena-Bano, J. Martinez-Roman, R. Puche-Panadero, M. Pineda-Sanchez, J. Perez-Cruz, and M. Riera-Guasp, "Induction machine model with space harmonics for fault diagnosis based on the convolution theorem," *International Journal of Electrical Power & Energy Systems*, vol. 100, pp. 463–481, 2018, DOI: 10.1016/J.IJEPES.2018.03.001.

[15] R. Puche-Panadero, J. Martinez-Roman, A. Sapena-Bano, and J. Burriel-Valencia, "Diagnosis of Rotor Asymmetries Faults in Induction Machines Using the Rectified Stator Current*," IEEE Transactions on Energy Conversion*, vol. 35, no. 1, pp. 213–221, 2020. DOI: 10.1109/TEC.2019.2951008.

[16] J. Burriel-Valencia et al., "Automatic Fault Diagnostic System for Induction Motors under Transient Regime Optimized with Expert Systems," *Electronics*, vol. 8, no. 1, p. 6, 2018. DOI: 10.3390/ELECTRONICS8010006.

[17] A. A. Suzen, B. Duman, and B. Sen, "Benchmark Analysis of Jetson TX2, Jetson Nano and Raspberry PI using Deep-CNN," *2nd International Congress on Human-Computer Interaction, Optimization and Robotic Applications*, 2020. DOI: 10.1109/HORA49412.2020.9152915.

[18] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures," *IEEE Access*, vol. 6, pp. 64270–64277, 2018. DOI: 10.1109/ACCESS.2018.2877890.

[19] M. T. Pham, J. M. Kim, and C. H. Kim, "Deep learning-based bearing fault diagnosis method for embedded systems," *Sensors*, vol. 20, no. 23, pp. 1–15, 2020. DOI: 10.3390/s20236886.

[20] L. Chen, S. Li, Q. Bai, J. Yang, S. Jiang, and Y. Miao, "Review of Image Classification Algorithms Based on Convolutional Neural Networks," *Remote Sensing*, vol. 13, no. 22, p. 4712, 2021. DOI: 10.3390/RS13224712.

[21] A. Burger, C. Qian, G. Schiele, and D. Helms, "An Embedded CNN Implementation for On-Device ECG Analysis," *IEEE International Conference on Pervasive Computing and Communications Workshops*, 2020. DOI: 10.1109/PerComWorkshops48775.2020.9156260.

[22] Y. Zhijie, W. Lei, L. Li, L. Shiming, G. Shasha, and W. Shuquan, "Bactran: A hardware batch normalization implementation for CNN training engine," *IEEE Embedded Systems Letters*, vol. 13, no. 1, pp. 29–32, 2021. DOI: 10.1109/LES.2020.2975055.

[23] C. Alippi, S. Disabato, and M. Roveri, "Moving Convolutional Neural Networks to Embedded Systems: The AlexNet and VGG-16 Case," *17th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pp. 212–223, 2018. DOI: 10.1109/IPSN.2018.00049.

[24] M. Tan and Q. v. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," *36th International Conference on Machine Learning*, vol. 2019-June, pp. 10691–10700, 2019, DOI: 10.48550/arxiv.1905.11946.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, pp. 770–778, 2015. DOI: 10.48550/arxiv.1512.03385.

[26] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *3rd International Conference on Learning Representations*, 2014. DOI: 10.48550/arxiv.1409.1556.

[27] C. Szegedy et al., "Going deeper with convolutions," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1-9, 2015. DOI: 10.1109/CVPR.2015.7298594.

[28] A.L.Sayeth Saabith, MMM.Fareez and T.Vinothraj, "Python Current Trend Applications- An Overview", *International Journal of Advance Engineering and Research Development*, vol. 6, no. 10, pp. 6–12, 2019.

[29] W. Xie, C. Zhang, Y. Zhang, C. Hu, H. Jiang, and Z. Wang, "An Energy-Efficient FPGA-Based Embedded System for CNN Application," *IEEE International Conference on Electron Devices and Solid State Circuits*, pp. 2–3, 2018. DOI: 10.1109/EDSSC.2018.8487057.