Research Article

Aibi research, management and engineering journal. Volume 13, Number 1, Pag 11-19, January - April 2025, ISSN 2346-030X

Neural network quantification for solar radiation prediction: An approach for low power devices

Brenda Alejandra Villamizar-Medina[1], Angelo Joseph Soto-Vergel[2], Byron Medina-Delgado[3]
Darwin Orlando Cardozo-Sarmiento[4], Dinael Guevara-Ibarra[5], Oriana López-Bustamante[6]
[1,3,4,5,6]*Universidad Francisco de Paula Santander, Cúcuta - Colombia*
[2]*University at Buffalo, Buffalo - United States*

Received: june 12, 2024.
Accepted: november 29, 2024.
Publicado: january 01, 2025.

***Abstract*—** Accurate solar radiation prediction leverages various machine learning techniques, with artificial neural networks (ANN) being the most common and precise due to their ability to detect and learn relationships between meteorological variables and solar radiation. Traditionally, training and deploying these models require high-capacity computers. However, the proliferation of low-power smart devices, such as embedded systems and mobile devices, necessitates exploring methodologies for implementing ANN on systems with limited computational resources. This paper proposes a quantized neural network model for solar radiation prediction, considering the hardware limitations of low-power devices like the Raspberry Pi RP2040 microcontroller. The methodology involves five stages: hardware and software selection, neural network development and quantization, microcontroller implementation, model validation, and result analysis. Experimental design allows detailed performance evaluation of quantized neural networks, demonstrating that the TensorFlow Lite Quantized Aware model is suitable for solar radiation prediction. Metrics such as root mean square error (RMSE) of 44.24 and R^2 of 0.96 indicate that the selected quantized model differs from the original non-quantized model by less than 0.5% in RMSE and 0.04% in R^2. The study concludes that implementing quantized ANN models on microcontrollers is a technically and economically viable solution for solar radiation prediction. Quantization enables complex predictive models to run on low-cost, energy-efficient devices, thereby democratizing advanced prediction technologies for critical applications like solar energy generation.

Keywords: quantized neural network, solar radiation prediction, microcontroller.

*Corresponding author.
Email: angeloso@buffalo.edu (Angelo Joseph Soto Vergel).
Peer reviewing is a responsability of the Universidad de Santander.
How to cite this article: B. A. Villamizar-Medina, A. J. Soto-Vergel, B. Medina-Delgado, D. O. Cardozo-Sarmiento, D. Guevara-Ibarra and O. López-Bustamante, "Neural network quantification for solar radiation prediction: An approach for low power devices", *Aibi research, management and engineering journal*, vol. 13, no. 1, pp. 11-19 2025, doi: 10.15649/2346030X.4107

CC BY 4.0

I. INTRODUCTION

Solar radiation prediction is crucial in various applications, especially in energy generation, as it enhances the efficiency of photovoltaic solar power systems by enabling better production planning [1]. Machine learning techniques, particularly artificial neural networks (ANN), are commonly and effectively used for precise solar radiation prediction because they can detect and learn the relationships between meteorological variables and solar radiation. While ANN models are typically trained and deployed on high-capacity computers, the growing prevalence of low-power smart devices, such as embedded systems and mobile devices, necessitates research into methodologies for implementing ANN on systems with limited computational capabilities [2].

Quantization is a technique used to reduce hardware resource consumption in ANN implementation. This method decreases the precision of the ANN's weights and activations by converting the standard floating-point format to lower-bit formats. Reducing the bit count lowers computational intensity in terms of memory and energy consumption during processing. This technique facilitates the implementation of ANN on low-cost, low-power devices, particularly in autonomous solar energy monitoring and control systems [3], [4], [5].

Despite the hardware efficiency gains from quantization, challenges such as model precision loss must be addressed. Reducing the bit count for weight representation can degrade performance, especially in predictive tasks like solar radiation forecasting. Additionally, selecting the precision level for each network layer is crucial to ensure the model can still learn data patterns effectively [6], [7], [8].

To address these challenges, we propose a quantized ANN model for accurate solar radiation prediction, implemented on a microcontroller. This approach reduces the network's weights and activations, optimizing the limited computational resources of the microcontroller. By applying quantization, the model maintains satisfactory prediction accuracy while adapting to hardware constraints, demonstrating the feasibility of deploying advanced machine learning technologies on low-power platforms [9], [10].

This paper details the development and evaluation of a quantized ANN model for solar radiation prediction using the Raspberry Pi RP2040 microcontroller. The study covers the design, implementation, and testing stages of the model, highlighting the network architecture adaptations to maximize computational efficiency without compromising prediction accuracy. We discuss the implementation challenges and the solutions adopted, providing a valuable framework for future research on implementing intelligent systems on constrained hardware platforms.

II. METHODOLOGY

The development process of a quantized Artificial Neural Network (ANN) model for solar radiation prediction encompasses five critical stages: hardware and software selection, neural network development and quantization, microcontroller implementation, model validation, and results analysis.

a. Selection of Hardware and Software

The hardware selection process involves a comprehensive analysis of various microcontrollers, evaluating their processing and memory capacities using the Binary Decision and Selection Method (BDSM) [11]. This method involves creating an attribute matrix, followed by an emphasis matrix, and ultimately a solution matrix to determine the optimal device for ANN implementation, prioritizing the attributes with the highest weight. Software selection involves reviewing various ANN programming and quantization libraries and tools (e.g., TensorFlow (TF) [12], PyTorch [13]), considering factors such as ease of use, flexibility, computational efficiency, and available documentation [14].

b. Development and Quantization of the Neural Network

The ANN is initially designed using Convolutional Neural Network (CNN) and Fully Connected Neural Network (FCNN) architectures with various layer configurations. Quantization of the ANN is then performed using two main approaches [15], [16]:

- Quantization-Aware Training (QAT): This method involves quantizing weights and activations during training. It is applied in three scenarios: (i) training from scratch with a randomly initialized network, (ii) retraining a pre-trained model by quantizing it and briefly retraining on the same dataset, and (iii) fine-tuning a pre-trained model on a different dataset [17].
- Post-Training Quantization (PTQ): This method requires only a small unlabeled calibration set, reducing the model size by 4x and accelerating inference by 2-3x [18]. PTQ is applied after model training, eliminating the need for retraining and access to the full training dataset.

c. Microcontroller Implementation

The quantized model is programmed into the Raspberry Pi RP2040 microcontroller. This involves coding for input data collection, ANN execution, and prediction processing [19], [20]. The model is loaded onto the microcontroller using Python, and the necessary code is developed for input data collection. The microcontroller executes the ANN model by processing input data through its layers with quantized weights and generates predictions, which are then displayed.

d. Model Validation

Model validation assesses the model's ability to generalize patterns from training data to new data. For the solar radiation prediction model implemented on the Raspberry Pi RP2040 microcontroller, validation involves comparing model predictions with actual solar radiation data.

Independent solar radiation datasets, not used during model training, are collected and used as inputs for the implemented model, which generates corresponding predictions. Performance metrics such as the R^2 coefficient and Root Mean Squared Error (RMSE) are used to quantify model accuracy. The R^2 coefficient measures the goodness of fit, with values close to 1 indicating a perfect fit. RMSE measures prediction error dispersion, with lower values indicating better accuracy [21].

e. Results Analysis

A thorough comparison of the quantized model's performance with similar models is conducted, identifying areas for improvement and optimization opportunities. Potential weaknesses in the current implementation are addressed with proposed solutions, including adjustments in quantization algorithms, code optimizations for model execution on the microcontroller, or changes in the model architecture to better suit hardware constraints.

By addressing these stages comprehensively, the proposed quantized ANN model demonstrates the feasibility and effectiveness of deploying advanced machine learning techniques on low-power, cost-effective devices for solar radiation prediction.

III. RESULTS

a. Selection of Hardware and Software

The microcontroller selection involved comparing four devices: PIC16F877, Raspberry Pi (RPi) RP2040, Arduino Uno, and Arduino Nano 33 BLE. Table 1 details the most relevant technical specifications for each device.

Table 1: Microcontroller analysis parameters.

Microcontroller	Energy Consumption (mA)	Processing Speed (MHz)	Flash Memory (Kb)	Price (Thousands of COP)
PIC16F877	25	20	14	69
RPi RP2040	12	133	16,000	16
Arduino Uno	50	16	32	33
Arduino Nano 33 BLE	15	64	1,000	300

Source: Own Elaboration.

A matrix of attributes (Table 2) was created, assigning a corresponding variable to each attribute: A1 for energy consumption, A2 for processing speed, A3 for storage, and A4 for price. Emphasis coefficient matrices were also developed for each attribute (Tables 3 to 6).

Table 2: Microcontroller Attribute Matrix.

	A1	A2	A3	A4	Value	Weight
A1	x	1	0	1	2	0.333
A2	0	x	1	1	2	0.333
A3	1	0	x	0	1	0.167
A4	0	0	1	x	1	0.167
Total					6	1

Source**:** Own Elaboration.

Table 3: Emphasis Coefficient Matrix for Attribute A1: Energy Consumption.

	PIC16F877	RPi RP2040	Arduino Uno	Arduino Nano 33 BLE	Value	Weight
PIC16F877	x	0	1	0	1	0.167
RPi RP2040	1	x	1	1	3	0.5
Arduino Uno	0	0	x	0	0	0
Arduino Nano 33 BLE	1	0	1	x	2	0.333
TOTAL					6	1

Source**:** Own Elaboration.

Table 4: Emphasis Coefficient Matrix for Attribute A2: Processing Speed.

	PIC16F877	RPi RP2040	Arduino Uno	Arduino Nano 33 BLE	Value	Weight
PIC16F877	x	0	1	0	1	0.167
RPi RP2040	1	X	1	1	3	0.5
Arduino Uno	0	0	x	0	0	0
Arduino Nano 33 BLE	1	0	1	x	2	0.333
Total					**6**	**1**

Source**:** Own Elaboration.

Table 5: Emphasis Coefficient Matrix for Attribute A3: Flash Memory Storage.

	PIC16F877	RPi RP2040	Arduino Uno	Arduino Nano 33 BLE	Value	Weight
PIC16F877	x	0	0	0	0	0
RPi RP2040	1	x	1	1	3	0.5
Arduino Uno	1	0	x	0	1	0.167
Arduino Nano 33 BLE	1	0	1	x	2	0.333
Total					6	1

Source**:** Own Elaboration.

Table 6: Emphasis Coefficient Matrix for Attribute A4: Price.

	PIC16F877	RPi RP2040	Arduino Uno	Arduino Nano 33 BLE	Value	Weight
PIC16F877	x	0	0	1	1	0.167
RPi RP2040	1	x	1	1	3	0.5
Arduino Uno	0	1	x	1	2	0.333
Arduino Nano 33 BLE	0	0	0	x	0	0
Total					6	1

Source: Own Elaboration.

Based on the weights from Tables 1 to 6, the solution matrix was developed, as shown in Equation 1. Variables were assigned to each microcontroller: M1 for PIC16F877, M2 for RPi RP2040, M3 for Arduino UNO, and M4 for Arduino Nano 33 BLE.

$$\begin{bmatrix} M1 \\ M2 \\ M3 \\ M4 \end{bmatrix} = \begin{bmatrix} 0.167 & 0.5 & 0 & 0.33 \\ 0.167 & 0.5 & 0 & 0.33 \\ 0 & 0.5 & 0.167 & 0.33 \\ 0.167 & 0.5 & 0.333 & 0 \end{bmatrix} * \begin{bmatrix} 0.333 \\ 0.333 \\ 0.167 \\ 0.167 \end{bmatrix} \quad (1)$$

The decision and binary selection method resulted in Equation 2, selecting the RPi RP2040 microcontroller with a weight of 0.5, the highest value.

$$\begin{bmatrix} M1 \\ M2 \\ M3 \\ M4 \end{bmatrix} = \begin{bmatrix} 0.139 \\ 0.5 \\ 0.083 \\ 0.277 \end{bmatrix} \quad (2)$$

The RPi RP2040 demonstrated superior characteristics compared to the other evaluated devices, with significantly lower energy consumption, higher processing speed, and storage, while also being economically affordable in the national market.

In software selection, specific features such as the ease of use of open-source libraries, existing documentation, and the developer community were considered. A literature review and practical experience led to the use of the TF library. Additionally, Google Colaboratory was employed for code execution and neural network model implementation.

b. Development and Quantization of the Neural Network

Three ANN models were executed experimentally, varying the number of convolutional and fully connected layers to compare total parameters and metric values: Mean Squared Error (MSE), Root Mean Squared Error (RMSE), model size (SIZE), and R^2, as shown in Table 7.

Table 7: Comparison of Neural Network Model Metrics.

Model Number	Convolutional Layer	Fully Connected Layer	No. of Parameters	MSE	RMSE	Size (Bytes)	R^2
1	1	0	6,657	2,377.63	48.76	222,627	0.96
2	1	1	2,389	1,939.78	44.04	161,016	0.97
3	2	1	4,853	2,492.36	49.92	228,983	0.96

Source: Own Elaboration.

Considering Table 7, model architecture 2, consisting of one convolutional layer and one fully connected layer, was chosen for its lower error values and size, with R^2 closest to 1. The factors and levels of the experimental design were then identified (Table 8), creating a total of 180 combinations to measure metrics: accuracy degradation, inference time, energy consumption, and model memory size.

Table 8: Neural Network Experiment Matrix.

Factors	Levels	Metrics
Types of Quantization	Without Quantization FP32	• Accuracy degradation. • Inference time. • Mode size. • Energy consumption.
	Post-training Quantization Weights and INT8 Activation	
	Post-training Quantization Weights INT8 and INT16 Activation	
	Quantization-aware Training Weights and INT8 Activation	
No. of neurons	1024, 512, 256, 128, 64, 32, 16, 8, 4, and 2	
Kernel size	8x1, 4x1, and 2x1	
Number of filters	64, 32, 16, 8, 4, and 2	

Source: Own Elaboration.

c. Implementation in the Microcontroller

The model's execution code was designed using Python, following this algorithm:

- Libraries for TF and its Keras interface for deep learning, along with mathematical, data manipulation, and graphics libraries, were imported.
- The data was read from an Excel file, preprocessing it by removing unnecessary columns, transposing, and adjusting it for a suitable format.
- A sequence function for training in groups of 10 data points was defined, splitting into feature sequences (all columns except the last) and label sequences (the last column).
- Training and testing subsets were created, extracting 80% for training and 20% for testing.

For constructing the CNN model, convolution layers (Conv2D) were imported, specifying the number of filters, kernel size, activation function ('relu'), and input shape. Data was reformatted for the first convolution layer, and additional layers included max pooling (MaxPooling2D), flattening (Flatten), and a densely connected layer (Dense) with a single output unit and linear activation. Each layer was sequentially connected.

The model was trained with the following hyperparameters: number of epochs, batch size, validation data for performance evaluation, and early stopping criteria. The 'History' variable stored training process information and epoch metrics for performance visualization.

Quantization techniques were applied post-training:

- Post-training Quantization - Weights and Activation Functions to INT8: The TF model was converted to a TF Lite model without quantization to maintain original precision parameters. The representative dataset was selected for model quantization with 8-bit integers, and the new quantized model was saved.
- Post-training Quantization - Weights INT8 and Activation Functions INT16: This technique followed the previous process, modifying to 8-bit weights and 16-bit activation functions.
- Quantization-aware Training - Weights and Activation Functions to INT8: The original model was quantized using TF Model Optimization, compiled, and trained for 500 epochs with early stopping criteria. The trained model was converted to TF Lite, applying 8-bit integer quantization and specifying TF Lite's integrated operations, selecting the representative dataset for precise quantization.

This approach ensured the model ran efficiently on low-power devices while maintaining acceptable prediction accuracy.

d. Model Validation

The stored information in 'History' is used to visualize loss curves during training and to graph them as shown in Figure 1. If losses in both the training and validation groups decrease, it indicates good model learning. However, if there's a variation where loss decreases in one group and increases in the other, it signifies overfitting.

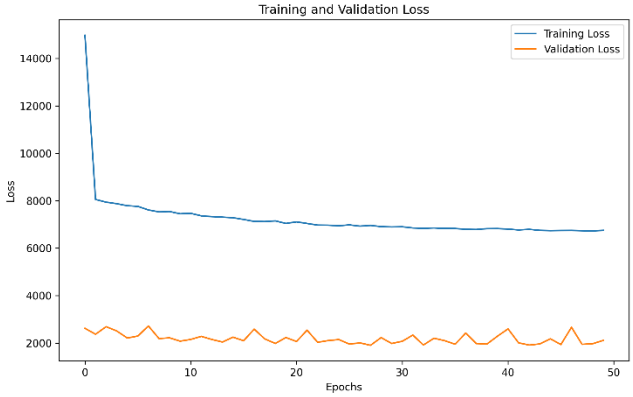

Figure 1: Loss Curve During Training and Validation.
Source: Own Elaboration.

Subsequently, the real data versus the predicted data are compared, the alignment of the predictions is evaluated, and error metrics between the data groups are calculated to quantitatively measure the model's performance in predicting values in the test set. For the radiation prediction models, as shown in Figure 2, a number 'n' of time steps is established, in this case with n = 50.

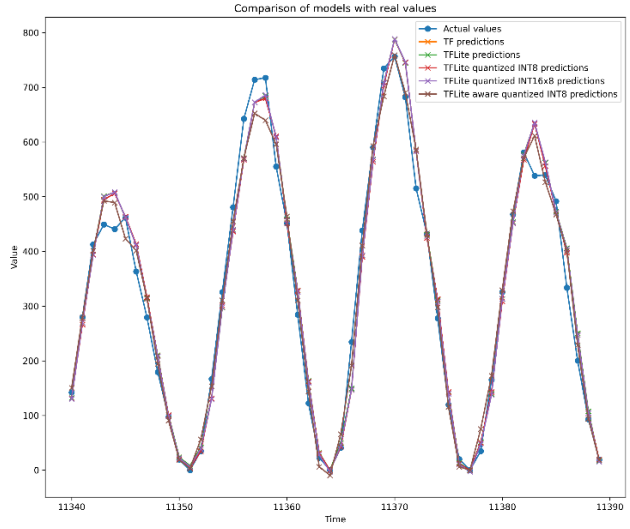

Figure 2: Comparison of Real Data vs Predicted Data for the Last n Time Steps.
Source: Own Elaboration.

After comparing the prediction of the quantized models with the original models, two functions are defined in the code: 'predict_tflite' and 'evaluate_tflite', to perform inferences and evaluations on a quantized TF Lite (TFLite) model using test data and true labels. The 'predict_tflite' function takes a quantized TFLite model and test data as input, prepares the test data, initializes the interpreter, and performs inferences with the TFLite model, returning the resulting predictions.

On the other hand, the 'evaluate_tflite' function takes a quantized TFLite model, test data, and true labels, uses the results obtained from the 'predict_tflite' function, and calculates the loss between the predictions and the true labels using metrics like MSE, RMSE, SIZE, R2, and inference time, employing the original model's loss function, and returning the resulting loss value.

The original model evaluates the loss on the test data using the 'evaluate' method. Then, the 'evaluate_tflite' function calculates the loss of the TF Lite models, including the original and quantized models. This allows comparison of the performance of different models in terms of their ability to fit the test data, where a lower loss value indicates a better model fit.

Finally, the file sizes of each TFLite model are calculated to compare the values obtained with the original TF model size and evaluate their storage efficiency.

e. Results Analysis

In validating the neural network model for solar radiation prediction, the corresponding experiments of the possible combinations of factors and levels from the proposed experimental design are executed. The data obtained in each experiment are organized as shown in Table 9, where the 'Inference Time' metric values are aligned with the model to which they belong, and the degradation proportion compared to the original TF model is calculated in percentage. In the following, the three types of quantization applied are referred to as Q1: TF Lite Quantized INT8, Q2: TF Lite Quantized INT16x8, and Q3: TF Lite Quantized Aware.

Similarly, the loss values obtained for each metric with the evaluated models and their size are organized in Table 10 and Table 11 respectively.

Table 9: Inference Time - Experiment No. 1.

Model	Inference Time	Proportion
TF	0.121 ms	0.00 %
TF Lite	0.221 ms	-82.87 %
Q1	0.101 ms	16.20 %
Q2	0.137 ms	-13.95 %
Q3	0.064 ms	46.59 %

Source**:** Own Elaboration.

Table 10: Loss Comparison in Each Metric - Experiment No. 1.

Model	MSE	MSE Degradation	RMSE	RMSE Degradation	MAE	MAE Degradation	R^2	R^2 Degradation
TensorFlow	2,231.56	0.00 %	47.23	0.00 %	34.58	0.00 %	0.96	0.00 %
TF Lite	2,231.56	0.00 %	47.23	0.00 %	34.58	0.00 %	0.96	0.00 %
Q1	2,348.66	-5.24 %	48.46	-2.59 %	35.28	-2.03 %	0.95	0.20 %
Q2	3,587.16	-60.74 %	59.89	-26.78 %	36.45	-5.42 %	0.93	2.41 %
Q3	3,026.68	-35.63 %	55.01	-16.46 %	40.53	-17.20 %	0.94	1.41 %

Source: Own Elaboration.

Table 11: Model Size Comparison - Experiment No. 1.

Model	Size (bytes)	Reduction (bytes)	Proportion
TensorFlow	1,736,899	0	0.00 %
TensorFlow Lite	532,916	1,203,983	69.31 %
Q1	136,984	1,599,915	92.11 %
Q2	138,016	1,598,883	92.05 %
Q3	141,688	1,595,211	91.84 %

Source: Own Elaboration.

To determine the best model for implementing solar radiation prediction, the data obtained in each experiment are collected and organized in a spreadsheet. This allows for a comprehensive comparison of all quantized TFL models with the original TFL model, evaluating the Root Mean Square Error (RMSE), which refers to prediction errors, and the R2 coefficient, indicating inference accuracy.

Three-dimensional graphs show the relationship between the number of neurons, kernel size, the number of filters, and the evaluated metric for each quantized neural network model. Thus, Figure 3 presents the RMSE metric data, showing that the Q3 model has more points with low RMSE values. Figure 4 presents the R2 metric data, showing that the total points of Q3 present a high R2. Furthermore, Figures 5 and 6 show box plots of the models for the RMSE and R2 metrics, where the quantized Q3 model performs similarly to the non-quantized models, while the data for Q1 and Q2 differ.

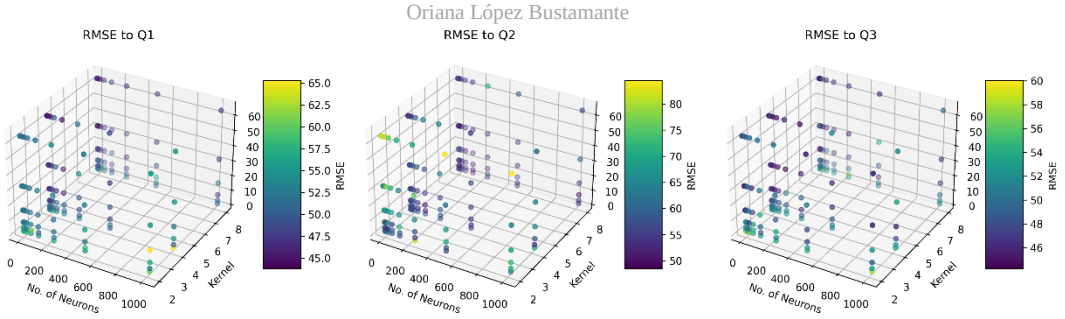

Figure 3: 3D Graph Comparing RMSE Metric.
Source: Own Elaboration.

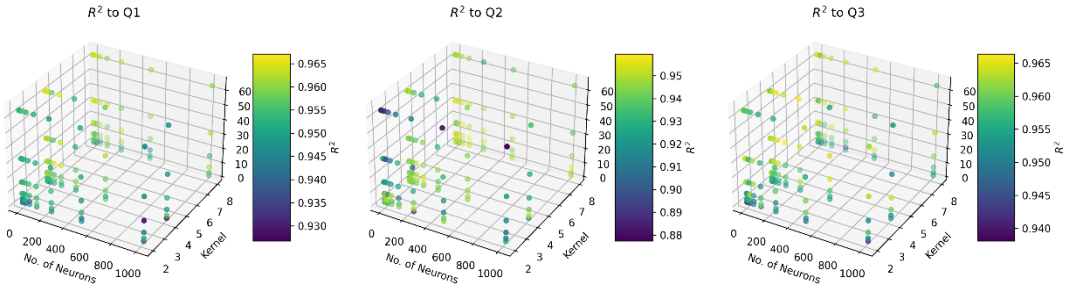

Figure 4: 3D Graph Comparing R2 Metric.
Source: Own Elaboration.

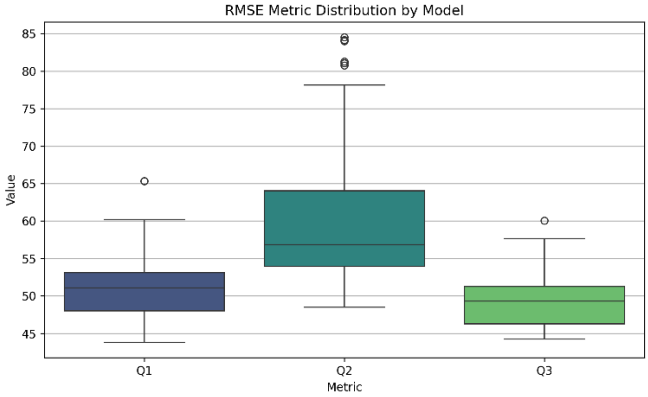

Figure 5: RMSE Metric Distribution by Model.
Source: Own Elaboration.

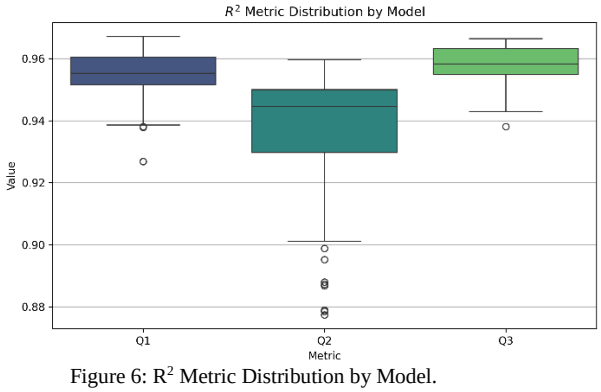

Figure 6: R^2 Metric Distribution by Model.
Source: Own Elaboration.

Graphically, it is observed that the quantization technique during training (Q3) shows a smaller range of variation compared to the original TF and TFL models. Additionally, considering that a lower RMSE value and a value close to one for the R2 coefficient indicate a better model fit, it is established that the appropriate quantized ANN model for microcontroller implementation is specified in Table 12.

Table 12: Specifications - Selected Quantized ANN Model

Experiment No.	No. of Neurons in Dense Layer	Kernel	No. of Filters	No. of Parameters	RMSE	R^2
169	2	4x1	64	1,093	44.24	0.96

Source: Own Elaboration.

With the identified ANN model and using the 'keras_spiking' library, an energy consumption estimate is performed on different processing units as shown in Table 13.

Table 13: Energy Consumption Estimate - Selected Quantized ANN Model.

Processing Unit	Total Energy per Inference (Joule)
CPU	$22.36x10^{-3}$
GPU	$410.44x10^{-3}$
ARM	$8.60x10^{-3}$

Source**:** Own Elaboration.

Finally, Table 14 presents the degradation of the evaluated metrics, and Table 15 presents the inference time and model size in memory.

Table 14: Metrics Degradation - Selected Quantized ANN Model.

Model	RMSE	RMSE Degradation	MAE	MAE Degradation	R^2	R^2 Degradation
TF	44.43	0.00 %	32.00	0.00 %	0.96	0.00 %
Q3	44.24	0.42 %	31.01	3.09 %	0.96	0.03 %

Source: Own Elaboration.

Table 15: Inference Time and Model Size - Selected Quantized ANN Model.

Model	Inference Time	Proportion	Model Size (bytes)	Proportion
TF	0.12 ms	0.00 %	140,544	0.00 %
Q3	0.03 ms	77.56 %	6,016	95.71 %

Source: Own Elaboration.

IV. CONCLUSION

The implementation of quantized ANN models in microcontrollers proves to be a technically and economically viable solution for solar radiation prediction. Thanks to quantization, it is possible to run complex predictive models on low-cost, low-power devices, thus democratizing advanced prediction technologies in critical applications such as solar energy generation and precision agriculture.

The results highlight that through the applied experimental design, the performance of the quantized neural networks is evaluated in detail, observing that the "Tensor Flow Lite Quantized Aware" ANN model is the suitable model for implementing solar radiation prediction, with metrics such as RMSE of 44.24 and R2 of 0.96, indicating that the selected quantized model differs from the original non-quantized model by less than 0.5 % and 0.04 % respectively in these metrics.

The Tensor Flow Lite Quantized Aware ANN model presents a 95 % reduction in model size and minimal energy consumption, demonstrating that quantization achieves significant reductions in memory storage and energy consumption of ANN models without compromising performance and prediction efficiency.

V. REFERENCES

[1] P.-E. Novac, G. Boukli Hacene, A. Pegatoquet, B. Miramond, and V. Gripon, “Quantization and Deployment of Deep Neural Networks on Microcontrollers,” Sensors, vol. 21, no. 9, p. 2984, Apr. 2021, doi: 10.3390/s21092984.

[2] A. Aljanad, N. M. L. Tan, V. G. Agelidis, and H. Shareef, “Neural Network Approach for Global Solar Irradiance Prediction at Extremely Short-Time-Intervals Using Particle Swarm Optimization Algorithm,” Energies (Basel), vol. 14, no. 4, p. 1213, Feb. 2021, doi: 10.3390/en14041213.

[3] W. I. Hameed et al., “Prediction of Solar Irradiance Based on Artificial Neural Networks,” Inventions, vol. 4, no. 3, p. 45, Aug. 2019, doi: 10.3390/inventions4030045.

[4] Q. Huang and S. Wei, “Improved quantile convolutional neural network with two-stage training for daily-ahead probabilistic forecasting of photovoltaic power,” Energy Convers Manag, vol. 220, p. 113085, Sep. 2020, doi: 10.1016/j.enconman.2020.113085.

[5] M. M. Lotfinejad, R. Hafezi, M. Khanali, S. S. Hosseini, M. Mehrpooya, and S. Shamshirband, “A Comparative Assessment of Predicting Daily Solar Radiation Using Bat Neural Network (BNN), Generalized Regression Neural Network (GRNN), and Neuro-Fuzzy (NF) System: A Case Study,” Energies (Basel), vol. 11, no. 5, p. 1188, May 2018, doi: 10.3390/en11051188.

[6] S. Islam, J. Deng, S. Zhou, C. Pan, C. Ding, and M. Xie, “Enabling Fast Deep Learning on Tiny Energy-Harvesting IoT Devices,” in 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE, Mar. 2022, pp. 921–926. doi: 10.23919/DATE54114.2022.9774756.

[7] X. Wei, H. Chen, W. Liu, and Y. Xie, “Mixed-Precision Quantization for CNN-Based Remote Sensing Scene Classification,” IEEE Geoscience and Remote Sensing Letters, vol. 18, no. 10, pp. 1721–1725, Oct. 2021, doi: 10.1109/LGRS.2020.3007575.

[8] I. N. Kosovic, T. Mastelic, and D. Ivankovic, “Using Artificial Intelligence on environmental data from Internet of Things for estimating solar radiation: Comprehensive analysis,” J Clean Prod, vol. 266, p. 121489, Sep. 2020, doi: 10.1016/j.jclepro.2020.121489.

[9] Y. Nahshan et al., “Loss aware post-training quantization,” Mach Learn, vol. 110, no. 11–12, pp. 3245–3262, Dec. 2021, doi: 10.1007/s10994-021-06053-z.

[10] I. L. Orasan, C. Seiculescu, and C. D. Caleanu, “Benchmarking TensorFlow Lite Quantization Algorithms for Deep Neural Networks,” in 2022 IEEE 16th International Symposium on Applied Computational Intelligence and Informatics (SACI), IEEE, May 2022, pp. 000221–000226. doi: 10.1109/SACI55618.2022.9919465.

[11] J. E. Gutierrez-Lopera, J. A. Toloza-Rangel, Á. J. Soto-Vergel, O. A. López-Bustamante, and D. Guevara-Ibarra, “Sistema integrado de monitoreo inalámbrico de variables agroambientales en un cultivo de tomate para la generación de mapas de intensidad,” Revista UIS Ingenierías, vol. 20, no. 2, Feb. 2021, doi: 10.18273/revuin.v20n2-2021014.

[12] R. David et al., "TensorFlow Lite Micro: Embedded Machine Learning for TinyML Systems," in Proceedings of Machine Learning and Systems, A. Smola, A. Dimakis, and I. Stoica, Eds., 2021, pp. 800–811. [Online]. Available: https://proceedings.mlsys.org/paper_files/paper/2021/file/6c44dc73014d66ba49b28d483a8f8b0d-Paper.pdf.

[13] A. Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in Advances in Neural Information Processing Systems, H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

[14] R.-Y. Ju, T.-Y. Lin, J.-H. Jian, and J.-S. Chiang, "Efficient convolutional neural networks on Raspberry Pi for image classification," J Real Time Image Process, vol. 20, no. 2, p. 21, Apr. 2023, doi: 10.1007/s11554-023-01271-1.

[15] S. S. Saha, S. S. Sandha, and M. Srivastava, "Machine Learning for Microcontroller-Class Hardware: A Review," IEEE Sens J, vol. 22, no. 22, pp. 21362–21390, Nov. 2022, doi: 10.1109/JSEN.2022.3210773.

[16] M. Rusci, M. Fariselli, A. Capotondi, and L. Benini, "Leveraging Automated Mixed-Low-Precision Quantization for Tiny Edge Microcontrollers," 2020, pp. 296–308. doi: 10.1007/978-3-030-66770-2_22.

[17] I. Hubara, Y. Nahshan, Y. Hanani, R. Banner, and D. Soudry, "Accurate Post Training Quantization With Small Calibration Sets," in Proceedings of the 38th International Conference on Machine Learning, M. Meila and T. Zhang, Eds., in Proceedings of Machine Learning Research, vol. 139. PMLR, Sep. 2021, pp. 4466–4475. [Online]. Available: https://proceedings.mlr.press/v139/hubara21a.html.

[18] S. Ghamari et al., "Quantization-Guided Training for Compact TinyML Models," Mar. 2021.

[19] Y. Abadade, A. Temouden, H. Bamoumen, N. Benamar, Y. Chtouki, and A. S. Hafid, "A Comprehensive Survey on TinyML," IEEE Access, vol. 11, pp. 96892–96922, 2023, doi: 10.1109/ACCESS.2023.3294111.

[20] P. Warden and D. Situnayake, Tinyml: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers. O'Reilly Media, 2019.

[21] T. O. Hodson, "Root-mean-square error (RMSE) or mean absolute error (MAE): when to use them or not," Geosci Model Dev, vol. 15, no. 14, pp. 5481–5487, Jul. 2022, doi: 10.5194/gmd-15-5481-2022.