Indonesian Journal of Electrical Engineering and Computer Science
Vol. 30, No. 1, April 2023, pp. 229~236
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v30.i1.pp229-236

Edge device for movement pattern classification using neural network algorithms

Ricardo Yauri[1], Rafael Espino[2]
[1]Faculty of Systems and Informatic Engineering, Universidad Nacional Mayor de San Marcos, Lima, Perú
[2]Department of Electronic Engineering, Faculty of Engineering, Universidad Tecnológica del Perú, Lima, Perú

Article Info

Article history:

Received Sep 12, 2022
Revised Nov 9, 2022
Accepted Nov 19, 2022

Keywords:

Edge computing
Edge device
Embedded intelligence
Internet of things
Neural network
Tiny machine learning

ABSTRACT

Portable electronic systems allow the analysis and monitoring of continuous time signals, such as human activity, integrating deep learning techniques with cloud computing, causing network traffic and high energy consumption. In addition, the use of algorithms based on neural networks are a very widespread solution in these applications, but they have a high computational cost, not suitable for edge devices. In this context, solutions are created that bring data analysis closer to the edge of the network, so in this paper models adapted to an edge device for the recognition of human activity are evaluated, considering characteristics such as inference time, memory, and precision. Two categories of models based on deep and convolutional neural networks are developed by implementing them in C language and comparing with the TensorFlow Lite platform. The results show that the implementations with libraries have a better accuracy result of 76% using principal component analysis inputs, obtaining an execution time of 9ms. Therefore, when evaluating the models, we must not only consider their accuracy but also the execution time and memory on the device.

Corresponding Author:

Ricardo Yauri
Faculty of Systems and Informatic Engineering, Universidad Nacional Mayor de San Marcos
Germán Amézaga 375, Lima, Perú
Email: ryaurir@unmsm.edu.pe

1. INTRODUCTION

The advancement of new technology has produced portable electronic systems with communication and processing capabilities, energy consumption, size and reduced cost which allows the analysis and continuous monitoring of human activity (HAR) [1], [2] but its operating approach is still in a cloud solutions scheme [3]. On the other hand, deep learning has achieved significant performance in applications such as temporal signal recognition and data analysis using wearable sensors, demonstrating its potential when deep neural networks are used [4]. This has allowed developing solutions that drive inference on edge devices (TinyML) [5] and analyzing characteristics such as processing response delay, memory usage, algorithm accuracy and data privacy Figure 1 [6], [7] in areas such as health monitoring [8], home or office [9].

Performing the integration of machine learning algorithms in edge devices requires the study of the characteristics mentioned above [1]. In this sense, the use of neural networks (NN) represents a very widespread solution, but it brings with it the problem of the computational load involved in training and inference activities [3], which makes it difficult to integrate it into edge devices (usually built with microcontrollers) [2], [10]. In the case of critical applications related to information privacy assurance, there is a loss of time in decision making when data analysis is performed in the cloud, generating delays and network traffic [11].

Due to the use of algorithms based on neural networks in edge devices, some works such as [1] investigate the efficiency of two models of neural networks such as the multilayer perceptron (MLP) and a

convolutional neural network (CNN), which are characterized in terms of CPU cycles, memory used, and power consumption of an Arm Cortex-M4 device. On the other hand, the research carried out in [2] focuses on proposing a high-speed CNN algorithm for activity data classification using a Raspberry Pi3 and STM32. In addition, in [12], [13] the implementation of classification models in smartphones for activity detection based on data extracted from these devices is carried out. In this research, frameworks from different manufacturers are evaluated, but a comparison is not made with manual implementations and in devices with lesser computational resources and low cost.

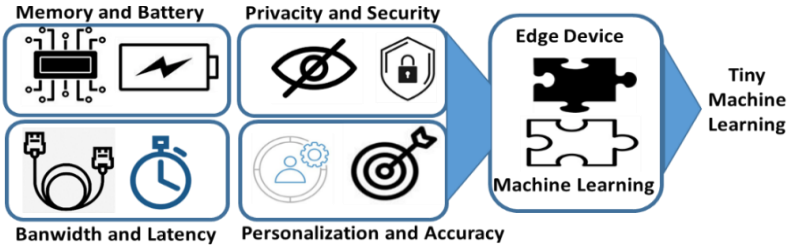

Figure 1. Considerations for implementing TinyML in embedded systems

In this context, solutions can be created to bring computational capacity closer to the edge of the network to avoid data transmission to the cloud for processing and alleviate critical problems related to latency [14], energy consumption [15], bandwidth and scalability [16]. Furthermore, the integration with artificial intelligence empowers machines with human-like intelligence and includes knowledge-based perception and decision-making capabilities [17]. Therefore, in this research we focus on evaluating algorithms based on neural networks adapted to an edge device in the context of a human activity recognition application [18].

The main contributions of the study are: i) an experimental work is shown to analyze the behavior of different machine learning models in an edge device with and without the use of a quantization framework, in the context of human activity recognition [19]-[21] using an own dataset; ii) the edge device is evaluated by analyzing inference time, memory usage and classification accuracy for two types of neural network-based learning models. In addition, the results of the paper allow to obtain indications to contribute to future research with machine learning solutions integrated in edge devices.

Section 2 describes the neural networks to be evaluated, section 3 describes the TensorFlow Lite framework. Sections 4 and 5 present the edge device, discussion and results, respectively. Finally, section 6 shows the conclusions.

2. CONVOLUTIONAL AND DEEP NEURAL NETWORKS

Results obtained by the neural networks are calculated using the values of the weights and the biases in a process called forward propagation as shown in Figure 2. This procedure has a much lower computational cost than that required during the training stage, therefore it can be implemented manually or using specialized libraries in hardware devices with reduced characteristics. This consists of four fundamental elements: input neurons with scaling, operations on the neurons of the hidden layers together with the weights and ways, and the output layer. Within each neuron, the weights are multiplied by each of the inputs, adding a bias, as shown in (1). Each neuron adds these weights to the input signal and then applies an activation function to calculate the output signal (2). The hidden layer sends the resulting signal to all units in the next layer (output units).

$$v_j = \sum_{i=1}^{n} w_{i,j} * x_j + b_j \quad (1)$$

$$Y_j = f(v_j) \quad (2)$$

For a classifier based on neural networks, such as CNN and/or deep multilayer networks (DNN) [22], $y = f * (x)$ assigns an input "x" to a category "y". A feedforward network defines a mapping $y = f(x; \theta)$ and learns the value of the parameters θ that result in the best function approximation. CNN are biologically inspired networks used in computer vision for image classification and pattern detection. It usually consists of a convolutional layer followed by a max-pooling layer [23]. In the case of the development of this paper, only the implementation of the calculation stage of the outputs in the edge device will be used. The types of networks evaluated in this research are:

– NNM: neural network using dimensionally reduced inputs (PCA) without TensorFlow Lite (TFlite).

- NNT: neural network using dimensionally reduced inputs (PCA) with TensorFlow lite.
- DNNSP: deep neural networks with TensorFlow Lite without dimension reduction.
- CNN1: convolutional networks with TensorFlow Lite without dimension reduction.
- CNN2: structure like the previous one but with the MaxPool technique.

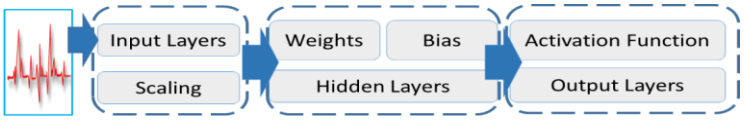

Figure 2. Feedforward neural network stages

3. TENSORFLOW LITE AND DEPLOYMENT ON AN EDGE DEVICE

TensorFlow is used to train machine learning models on computational devices with high hardware resources, while TensorFlow Lite is applied to the evaluation and deployment of models that support the required optimizations on a microcontroller edge device. TensorFlow Lite is platform dependent, turning a large and robust model into something light and small. It also allows you to convert and optimize a model considering three main goals:

- Reduce model size and RAM usage.
- Reduce the number of calculations required for each prediction, latency, and battery usage.
- Adapt the model to the specific limitations of the device.

Edge devices implement machine learning processes for pattern detection in input data [24]. Figure 3 shows a schematic of the application of edge intelligence in combination with machine learning for scenarios based on smart internet of things (IoT) sensors which allows alleviating critical problems of latency, energy consumption, bandwidth, and scalability. This technique is known as TinyML and provides a unique solution by summarizing and analyzing data at the edge, providing intelligent summary statistics that take into account patterns, anomalies, and analysis [25], [26].

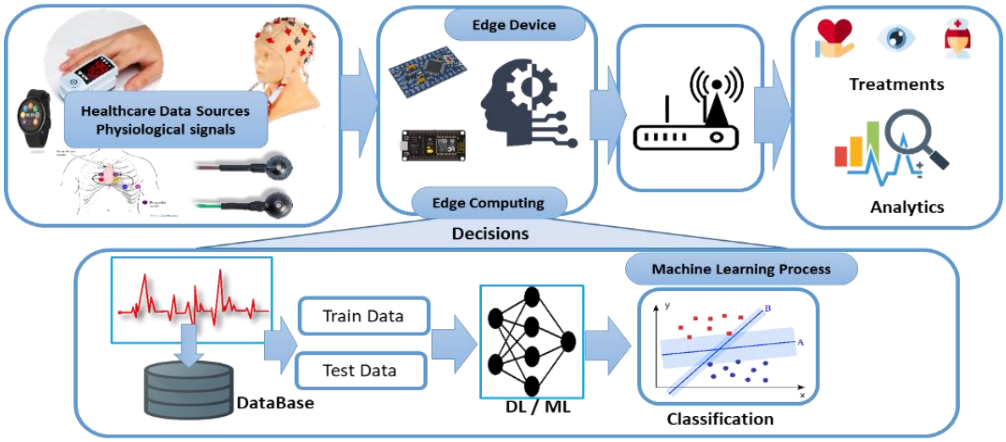

Figure 1. Edge device usage scenario

4. DESCRIPTION AND CHARACTERISTICS OF THE EDGE DEVICE

Figure 4 shows the processes implemented in the edge device for the detection of the class of movement of the person, preparation and cleaning of data. Data captures related to the running (co), walking (ca) and static (es) classes are made and based on this procedure, an own dataset is created to carry out the training of the models. An inertial sensor is used capturing data from the three axes using a sampling time of 10 milliseconds for 10 seconds and they are recorded in text files corresponding to each type of movement creating the dataset to train the models. Edge devices use the communication block for the transmission of online classification results and a data monitoring stage is implemented to display the results of the classification process. The edge device can be integrated into a system for detecting an event, sending an alert to database servers and/or applications so that interested people can be informed. This allows access to information at any time of the day as shown in Figure 5 [26], [27].

4.1. ESP32 IoT device and training data generation

An edge device based on the ESP32 module is used, which is manufactured by the company Ai Thinker based on a system on chip (SoC) with 520 KB of internal memory and 4 MB external [28]. The module contains serial transmit and receive pins and general-purpose input-output (GPIO) pins as shown in Figure 6. The edge computing IoT device is composed of four subsystems powered by a 2,200 mAh battery, an inertial sensor with accelerometer, and the Wi-Fi communication module.

4.2. Multilayer neural network with PCA

A stage based on reduction component analysis (PCA) is used as input to this model, using three variations of the multilayer neural network model with four neurons in the hidden layer and outputs corresponding to each of the three classes. To perform the training, the categorical output is transformed generating dummy variables. Considering the numerical data of classes (0, 1,2), the outputs are generated using three variables for each label: Walking (1 0 0), Running (0 1 0) and Static (0 0 1). The "hidden layer" and "output" processes are implemented through loops to automate the computation of each neuron's output as shown in Figure 7. A section of the program with the flowchart that performs the hidden layer Figure 7(a) shows the neural networks (NNM) model and output layer Figure 7(b) output computation using iteration statements.

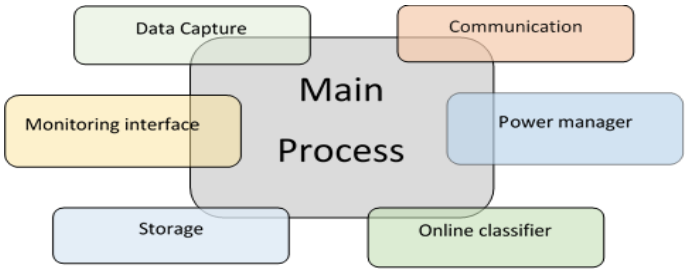

Figure 4. Interaction of firmware processes on the edge device

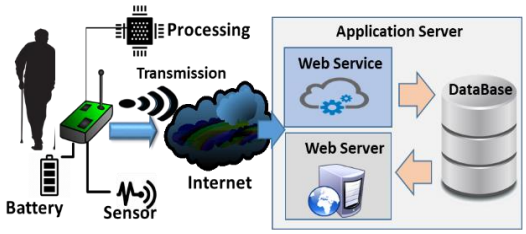

Figure 5. Usage environment for edge device

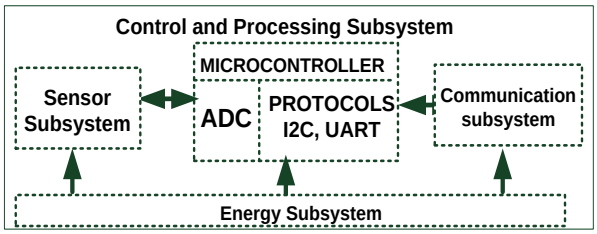

Figure 6. Sensor node hardware

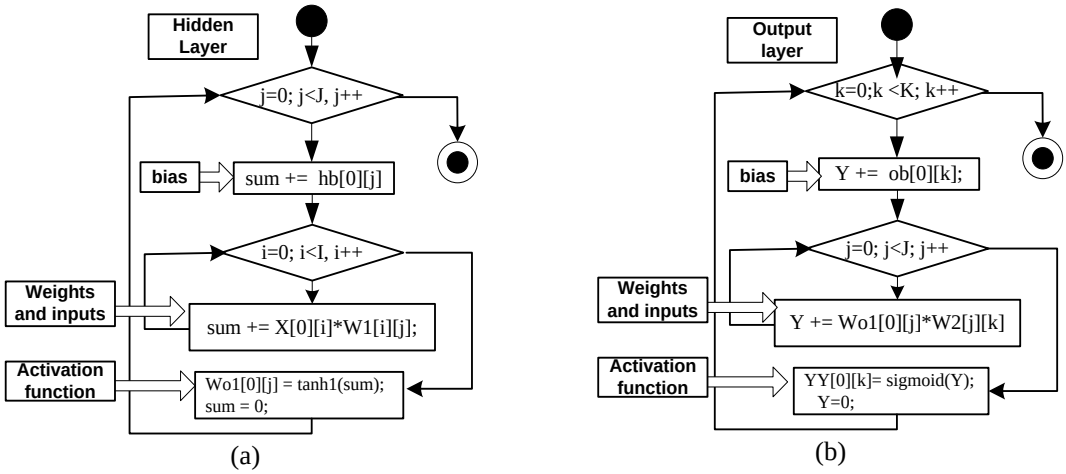

Figure 7. NNM model flowchart for (a) hidden layer process and (b) output layer process

4.3. Multilayer neural network with TensorFlow Lite and principal component analysis

Unlike the previous neural network implementation, this model (NNT) is created using TensorFlow Lite by obtaining a C language file with the model information, which is imported from the edge device along with the necessary firmware libraries as shown in Figure 8. Neural network prediction and configuration functions are encapsulated within TensorFlow Lite libraries to be imported from the edge device programming IDE. In addition, the original file is in binary format and is converted to hexadecimal using the "xxd" function for integration into the edge device program.

```
# Save the file as a C source file
!xxd -i exercice_model.tflite > exercice_model.cc
```

```
# Print the source file
#!cat exercice_model.cc
f = open("exercice_model.cc", "r")
print(f.read())
f.close()
```

```
unsigned char exercice_model_tflite[] = {
  0x1c, 0x00, 0x00, 0x00, 0x54, 0x46, 0x4c, 0x33, 0x14, 0x00, 0x20, 0x00,
  0x04, 0x00, 0x08, 0x00, 0x0c, 0x00, 0x10, 0x00, 0x14, 0x00, 0x00, 0x00,
  0x18, 0x00, 0x1c, 0x00, 0x14, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00,
```

Figure 8. Model file generated by TensorFlow Lite

5. RESULTS AND DISCUSSION

5.1. Evaluation of neural network models with principal component analysis

Evaluations on the accuracy of the models with data obtained from a principal component analysis process are evaluated in the embedded system for different neural network algorithm configurations as shown in Figure 9.

a) Manually generated neural networks (NNM). The accuracy results without implementing the neural networks with TensorFlow Lite are greater than 85%. In the case of the test results, these do not exceed 75% for the three types of neural networks, as shown in Figure 9(a). The execution time of processes can reach up to 41 ms when the models are implemented in the edge device, considering that the critical time is used by the PCA process.

b) Automatically generated neural networks with TFLite (NNT). Figure 9(b) shows the accuracy when making the prediction of the neural network generated with TFLite using inputs resulting from the PCA model with 3, 7 and 16 principal components with the ESP32 module (edge device). The training results are above 95%. The model using 16 components outperforms the others during its implementation on the edge device.

The times used by the embedded system for the execution of the classification processes are shown in Figure 10. Where in the worst case it can reach 10ms in transformation time and class prediction (for 16 components of the PCA process). Specifically, the PCA data transformation process is the one that uses the most processing time compared to the prediction time in NNT model, as shown in Figure 11.

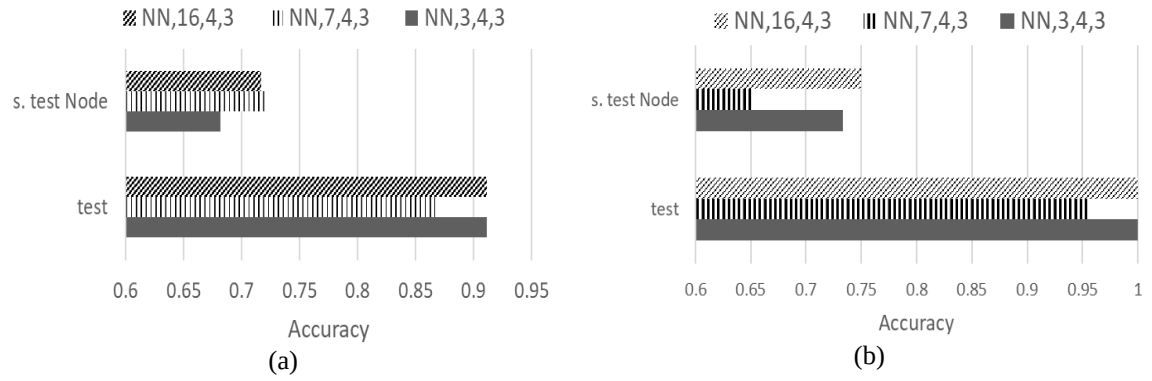

Figure 9. Evaluation of accuracy of (a) the model NNM and (b) NNT generated by the edge device

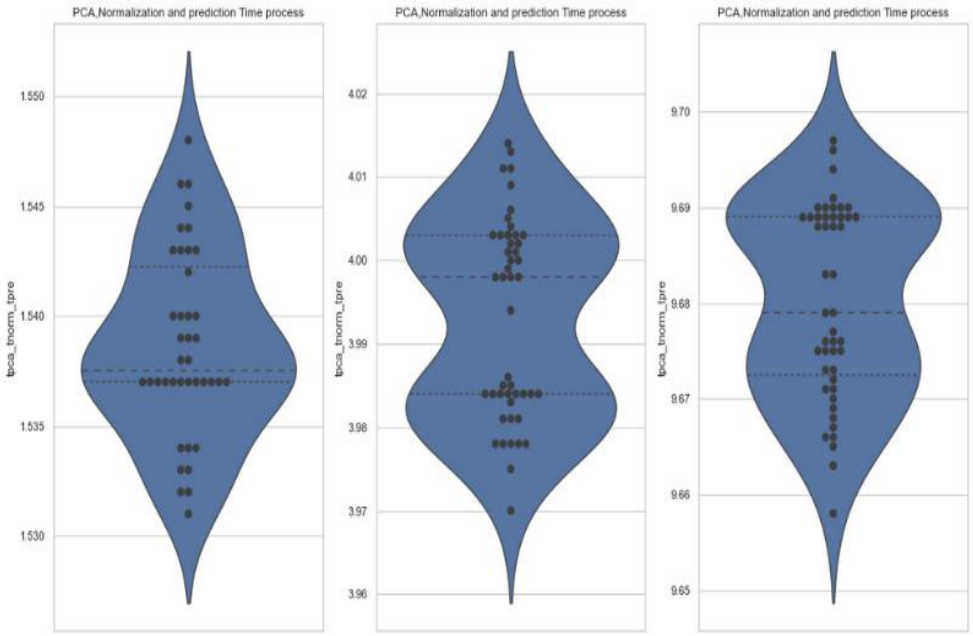

Figure 10. NNT model classification times and component analysis (PCA) in the edge device

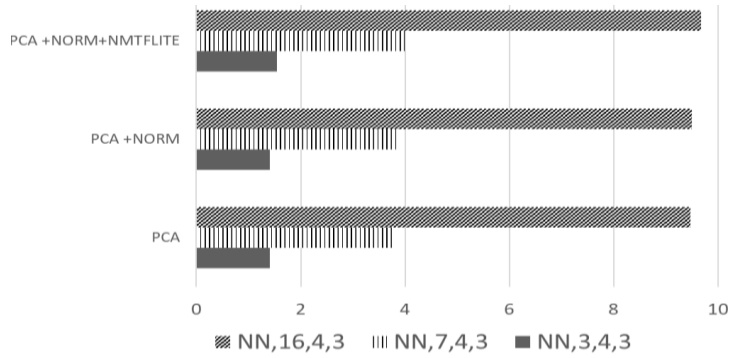

Figure 11. Process execution times on the edge device

5.2. Deep and convolutional neural networks without PCA

In this case, the results of the implementation of models CNN1, CNN2 and DNNSP considering the 300 samples of the inertial sensor readings are shown. In the case of the DNNSP, a layer with 300 scaled inputs, 20 neurons in the hidden layer and three outputs are considered. For the CNN1 network, the Maxpool process (to reduce the size of the generated model) and the 2D convolution process are used, because the TensorFlow Lite libraries are not compatible with one-dimensional data evaluations. For the CNN1 and CNN2 models, the absence and presence of the "MaxPool" process are considered Figure 12. Figure 13 shows that the DNNSP model is slightly better than CNN1 considering that the size of the generated file has a much lower weight in the CNN2 model when "Maxpool" is used Table 1. The prediction in the model generated by hardware is superior for the CNN2 model compared to the other models.

Layer (type)	Output Shape	Param #
conv2d_20 (Conv2D)	(None, 1, 300, 8)	136
max_pooling2d_18 (MaxPooling	(None, 1, 6, 8)	0
flatten_11 (Flatten)	(None, 48)	0
dense_12 (Dense)	(None, 3)	147

Figure 12. CNN1 neural network generated on the edge device

Table 1. Parameters of neural network models without PCA

	Size NNTF Kbytes	% Program memory	% Dynamic Memory
CNN1	102 040	32	12
CNN2	20 091	31	12
DNNSP	160 966	33	14

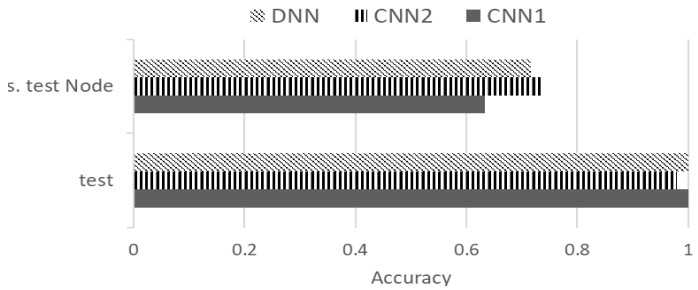

Figure 13. Accuracy values obtained edge device

6. CONCLUSION

The deployment of classification algorithms using neural networks directly on the edge device is evaluated by comparing their accuracy and execution times in hardware. Two categories of neural networks have been developed. The first one consists of algorithms with (NNM and NNT) and without (CNN1, CNN2 DNNSP) PCA type inputs. In addition, within each category, its implementation must be conducted considering its construction directly in C language from the definition of the feedforward process and using the functionalities of TensorFlow Lite libraries. The experimental results show that the implementations with TFLite (NNT) have a better result in the precision obtained with PCA inputs with 76% and an execution time of 9ms, so its use is recommended in the case presented in this paper.

In the case of the implementation of convolutional and deep networks with TFLite, the best results are obtained by the convolutional network model (CNN2) with an accuracy of 97%. As a conclusion, when evaluating an algorithm, we should not only consider its accuracy, but also the execution time (latency) and memory usage on the edge device. As future research, integration with other types of health monitoring sensors can be evaluated, obtaining power consumption and lifetime of edge devices.

ACKNOWLEDGEMENTS

This paper was developed in the doctoral studies at the Faculty Systems and Informatic Engineering, in the laboratories of the Faculty of Electronic and Electrical Engineering of the Universidad Nacional Mayor de San Marcos.

REFERENCES

[1] E. Lattanzi, M. Donati, and V. Freschi, "Exploring artificial neural networks efficiency in tiny wearable devices for human activity recognition," *Sensors*, vol. 22, no. 7, Apr. 2022, doi: 10.3390/S22072637.

[2] J. Chang, M. Kang, and D. Park, "Low-power on-chip implementation of enhanced SVM algorithm for sensors fusion-based activity classification in lightweighted edge devices," *Electronics*, vol. 11, no. 1, Jan. 2022, doi: 10.3390/ELECTRONICS11010139.

[3] N. Rashid, B. U. Demirel, and M. Abdullah Al Faruque, "AHAR: Adaptive CNN for energy-efficient human activity recognition in low-power edge devices," *>IEEE Internet of Things Journal*, vol. 9, no. 15, pp. 13041–13051, Aug. 2022, doi: 10.1109/JIOT.2022.3140465.

[4] E. S. Jeon, A. Som, A. Shukla, K. Hasanaj, M. P. Buman, and P. Turaga, "Role of data augmentation strategies in knowledge distillation for wearable sensor data," *>IEEE Internet of Things Journal*, vol. 9, no. 14, pp. 12848–12860, Jul. 2022, doi: 10.1109/JIOT.2021.3139038.

[5] M. E. Ghmary, Y. Hmimz, T. Chanyour, and M. O. C. Malki, "Time and resource constrained offloading with multi-task in a mobile edge computing node," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 4, pp. 3757–3766, 2020, doi: 10.11591/IJECE.V10I4.PP3757-3766.

[6] N. N. alajlan and D. M. Ibrahim, "TinyML: Enabling of inference deep learning models on ultra-low-power iot edge devices for AI Applications," *Micromachines*, vol. 13, no. 6, p. 851, May 2022, doi: 10.3390/MI13060851.

[7] R. Yauri, J. Lezama, and M. Rios, "Evaluation of a wireless low-energy mote with fuzzy algorithms and neural networks for remote environmental monitoring," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 23, no. 2, pp. 717–724, 2021, doi: 10.11591/IJEECS.V23.I2.PP717-724.

[8] Y. A. Qadri, A. Nauman, Y. Bin Zikria, A. V. Vasilakos, and S. W. Kim, "The future of healthcare internet of things: A survey of emerging technologies," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1121–1167, 2020, doi: 10.1109/COMST.2020.2973314.

[9] Z. Sharif, L. T. Jung, M. Ayaz, M. Yahya, and D. Khan, "Smart home automation by internet-of-things edge computing platform," *International Journal of Advanced Computer Science and Applications(IJACSA)*, vol. 13, no. 4, pp. 474–484, 2022, doi: 10.14569/IJACSA.2022.0130455.
[10] P. E. Novac, G. B. Hacene, A. Pegatoquet, B. Miramond, and V. Gripon, "Quantization and deployment of deep neural networks on microcontrollers," *Sensors*, vol. 21, no. 9, May 2021, doi: 10.3390/S21092984.
[11] A. M. Ghosh and K. Grolinger, "Edge-cloud computing for internet of things data analytics: embedding intelligence in the edge with deep learning," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 3, pp. 2191–2200, Mar. 2021, doi: 10.1109/TII.2020.3008711.
[12] A. Albeshri, "SVSL: A human activity recognition method using soft-voting and self-learning," *Algorithms*, vol. 14, no. 8, Aug. 2021, doi: 10.3390/A14080245.
[13] M. K, A. Ramesh, R. G, S. Prem, R. A A, and D. M. P. Gopinath, "1D convolution approach to human activity recognition using sensor data and comparison with machine learning algorithms," *International Journal of Cognitive Computing in Engineering*, vol. 2, pp. 130–143, Jun. 2021, doi: 10.1016/J.IJCCE.2021.09.001.
[14] P. Mahadevappa and R. K. Murugesan, "A data quarantine model to secure data in edge computing," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 12, no. 3, pp. 3309–3319, 2022, doi: 10.11591/IJECE.V12I3.PP3309-3319.
[15] R. Yauri, S. Rubiños, J. Grados, and M. Chauca, "Characterization of a low consumption wireless sensor node for the intensive transmission of physiological signals," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 14, no. 2, p. 957, May 2019, doi: 10.11591/ijeecs.v14.i2.pp957-965.
[16] V. Hayyolalam, M. Aloqaily, O. Ozkasap, and M. Guizani, "Edge intelligence for empowering iot-based healthcare systems," *IEEE Wireless Communications*, vol. 28, no. 3, pp. 6–14, Mar. 2021, doi: 10.1109/MWC.001.2000345.
[17] A. Bakhtiarnia, N. Milošević, Q. Zhang, D. Bajović, and A. Iosifidis, "Dynamic split computing for efficient deep edge intelligence," *arXiv preprint arXiv:2205.11269*, vol. 23, May 2022.
[18] Z. Shen, N. Howard, and J. Nunez-Yanez, "Big–little adaptive neural networks on low-power near-subthreshold processors," *Journal of Low Power Electronics and Applications*, vol. 12, no. 2, Jun. 2022, doi: 10.3390/JLPEA12020028.
[19] D. M. R. Martín and D. Rodríguez-Martín, "Contribución al análisis del movimiento humano aplicado a la identificación de posturas y bloqueos de la marcha en pacientes con Parkinson," Universitat Politècnica de Catalunya, 2014. Accessed: Nov. 09, 2021. [Online]. Available: https://dialnet.unirioja.es/servlet/tesis?codigo=97427&info=resumen&idioma=SPA
[20] P. K. D. Pramanik, S. Pal, and P. Choudhury, "Beyond automation: The cognitive IoT. artificial intelligence brings sense to the internet of things," in *Lecture Notes on Data Engineering and Communications Technologies*, vol. 14, Springer, Cham, 2018, pp. 1–37, doi: 10.1007/978-3-319-70688-71.
[21] R. Shrivastava and M. Pandey, "Adaptive window based fall detection using anomaly identification in fog computing scenario," *Multiagent Grid Syst.*, vol. 17, no. 1, pp. 15–37, 2021, doi: 10.3233/MGS-210341.
[22] W. Ertel, *Introduction to Artificial Intelligence*. Cham: Springer International Publishing, 2018. doi: 10.1007/978-3-319-58487-4.
[23] P. Fabian, "Scikit-learn: machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[24] M. T. Yazici, S. Basurra, and M. M. Gaber, "Edge machine learning: enabling smart internet of things applications," *Big Data Cogn. Comput.*, vol. 2, no. 3, pp. 1–17, Sep. 2018, doi: 10.3390/bdcc2030026.
[25] S. Salerno, "GitHub - eloquentarduino/tinymlgen: Generate C code for microcontrollers from Tensorflow models," 2021. https://github.com/eloquentarduino/tinymlgen (accessed Nov. 10, 2021).
[26] S. Raschka and V. Mirjalili, *Python machine learning : machine learning and deep learning with python, scikit-learn, and tensorflow 2*, 1st ed. Birmingham: Packt Publishing Ltd, 2019.
[27] R. Jenatton, G. Obozinski, and F. Bach, "Structured sparse principal component analysis," in *Journal of Machine Learning Research*, Mar. 2010, vol. 9, pp. 366–373.
[28] N. Cameron, *Electronics Projects with the ESP8266 and ESP32*. Apress, 2021. doi: 10.1007/978-1-4842-6336-5.

BIOGRAPHIES OF AUTHORS

Ricardo Yauri received the B.Eng. degree in electronic engineering from Universidad Nacional Mayor de San Marcos, Perú and the M.S. in biomedical engineering and Ph.D. student in Systems Engineering. Currently, he is an Associate Professor at the Department of Electronical Engineering at Universidad Nacional Mayor de San Marcos. He is professor at Universidad Tecnológica del Perú and Universidad Privada del Norte. He has participated as a teacher in courses oriented to the Internet of things and applications in home automation and the Cisco academy for IoT. He was researcher at INICTEL-UNI in the Embedded Systems and Internet of Things research group. His research interests include implementation of low consumption IoT devices, inference techniques, machine learning algorithms and computational intelligence. He can be contacted at email: ryaurir@unmsm.edu.pe.

Rafael Espino He is a graduate of the Faculty of Electronics of the National Universidad Nacional de Ingeniería, Perú. He is Specialist in the development of electronic systems with solid knowledge in photovoltaic systems and experience in implementing embedded firmware in electronic systems. He is currently a professor at the Universidad Tecnológica del Perú and at the Universidad de Ingeniería y Tecnología. He was researcher at INICTEL-UNI in the Embedded Systems research group. With aptitudes for teaching, research, and project management. He can be contacted at rafalloelo@gmail.com.