International Journal of Electrical and Computer Engineering (IJECE)
Vol. 13, No. 5, October 2023, pp. 5727~5736
ISSN: 2088-8708, DOI: 10.11591/ijece.v13i5.pp5727-5736

A multi-microcontroller-based hardware for deploying Tiny machine learning model

Van-Khanh Nguyen[1], Vy-Khang Tran[1], Hai Pham[1,2], Van-Muot Nguyen[1], Hoang-Dung Nguyen[1], Chi-Ngon Nguyen[1]
[1]Faculty of Automation Engineering, College of Engineering, Can Tho University, Can Tho, Vietnam
[2]Aerospace Engineering and Aviation Discipline, School of Engineering, Royal Melbourne Institute of Technology University, Melbourne, Australia

Article Info

Article history:

Received Jan 21, 2023
Revised Mar 20, 2023
Accepted Apr 7, 2023

Keywords:

Grayscale spectrogram image
Multi-tasking programming
Real-time operating system
Real-time recognition
Tiny machine learning

ABSTRACT

The tiny machine learning (TinyML) has been considered to apply on the edge devices where the resource-constrained micro-controller units (MCUs) were used. Finding a good platform to deploy the TinyML effectively is very crucial. The paper aims to propose a multiple micro-controller hardware platform for productively running the TinyML model. The proposed hardware consists of two dual-core MCUs. The first MCU is utilized for acquiring and processing input data, while the second one is responsible for executing the trained TinyML network. Two MCUs communicate with each other using the universal asynchronous receiver-transmitter (UART) protocol. The multi-tasking programming technique is mainly applied on the first MCU to optimize the pre-processing new data. A three-phase motors faults classification TinyML model was deployed on the proposed system to evaluate the effectiveness. The experimental results prove that our proposed hardware platform was improved 34.8% of the total inference time including pre-processing data of the proposed TinyML model in comparing with single micro-controller hardware platform.

Corresponding Author:

Chi-Ngon Nguyen
College of Engineering, Can Tho University
Campus II, 3/2 street, Ninh Kieu district, Can Tho city, Vietnam
Email: ncngon@ctu.edu.vn

1. INTRODUCTION

The demand for implementing machine learning model on the embedded computers and resource-constraint microcontrollers or Tiny machine learning (TinyML) [1]–[3] has been increased. Data analysis and decision making are necessarily applied to the edge devices according to the development of internet of thing (IoT) technology, therefore, the deployment of machine learning model on edges is very crucial to prevent the overload of cloud server systems. The Raspberry Pi or Jetson Nano embedded computing platforms have been used in deep learning networks (DLNs) applications [4]–[6]. These single-board computers have a powerful processor to successfully execute some DLNs tasks, nevertheless, power-consumption is their main issues that hardly apply to the edge devices using battery or renewable energy sources. Therefore, Tensorflow Lite [7] has been introduced to enable the trained DLNs to deploy effectively on tiny microcontrollers, such as nRF52840, ESP32. Based on this, Hussein *et al.* [8] proposed a customized multi-layer perception neural network which was built and trained on a personal computer and implemented on a limited microcontroller based on Tensorflow Lite. Similarly, a TinyML-oriented mosquito sound classification model has been proposed by Choi and Kim [9] operated on a 32-bit advanced reduced instruction set computer (RISC) machine (ARM) Cortex-M4F processor. TinyML has been also considered to deploy on some low power micro-controller units (MCUs) in the autonomous mini-vehicles or voice-recognition applications [10], [11]. However, these implementations have commonly unconcentrated on any techniques to enhance the performance of the proposed networks except Kwon and Park [11] have been proposed the field programmable gate array (FPGA)-based or hardware based solution.

The implementation of online training and making inference of the customized TinyML model on power-saving microcontrollers has been become research of interest. Some customized TinyML models such as TinyFedTL [12], machine learning-microcontroller unit (ML-MCU) [13], Train++ [14] or Globe2Train [15] have been proposed which can be operated directly on limited-resource microcontrollers. These platforms have been demonstrated the feasibility of building directly the TinyML models on the small MCUs. However, large scale models remain issues due to the lack of memory resources and processing speed of embedded machines. Further research should be conducted to find suitable solutions to increase the scale of these kind of TinyML models.

Real-time operating system (RTOS) is an operating system that is performed to the control of hardware, and must operate within specified time constraints [16], [17]. FreeRTOS [18] is one of the well-known RTOS kernels that is designed for many kind of MCUs. Due to the supporting of multi-tasking feature, RTOS is promising a good solution to improve the performance of artificial intelligence networks (AINs). In fact, Zim [19] applied the freeRTOS to parallel two parts of a AIN on two cores of ESP32 MCU. It is clearly to see that this work took advantage of the multitask on the multi-core MCUs to improve the inference time of the AINs. However, for the TinyML model building by the TensorFlow Lite, such solution is unable to apply due to the unknown structure of trained model. Furthermore, beside the difficulty of the implementing DLNs on MCUs, the pre-processing procedures of new data are also another issue that needs to be overcome. On top of that, a suitable multi-tasking mechanism corresponding with an effective hardware platform should be proposed to enhance executive time of the DLN based application.

The objective of this study is to propose a multiple MCU platform combining with multi-tasking programming based on the freeRTOS kernel to enhance the performance of the TinyML model. Our proposed hardware platform consists of two dual-core ESP32 MCUs for parallelly executing both pre-processing data and recognition tasks on their cores. The demonstrated TinyML model is a three-phase alternating current (AC) motor faults classification based on operation noise proposed by Nguyen *et al.* [20]. This DLN is modified to classify the faults of four motors instead of one as the original version, its input is a 64x64 pixel grayscale Mel-spectrogram image [21] of a one-second noise data segment. The trained model is implemented on Core 1 of the second ESP32 of the proposed hardware, while Core 0 is reserved. The pre-processing data procedures are multitasked by the first ESP32. All tasks are scheduled to run parallelly to optimize the total inference time of the motor's faults identification application.

2. DATA COLLECTION AND METHOD

2.1. Data description and analysis

The operated noise of three-phase AC motors corresponding to their common faults will be acquired. In this study, four datasets corresponding to four motor operation cases consist of normal operation and three common faults including phase shift, phase loss and bearing failure are logged. The target faults are generated deliberately, and their noises are acquired by a smartphone software at the sampling rate of 16 kHz. Figure 1 shows the experimental layout conducted by Tung *et al.* [22]. The activities are carried out on four different motors in accordance with parameters listed in Table 1.

Acquired datasets are processed to create data for implementing TinyML DLN. They are converted to grayscale Mel-spectrogram images for training and testing the proposed TinyML model before deploying the trained model on the proposed hardware. These images have been applied as the input of TinyML associated with the core algorithm can be found in [21] that can be modified for new project. The spectrogram image is created by three consecutive steps including denoising by wavelet filter using a soft threshold as described in (1) [23], splitting denoised data into one-second (i.e., a 16,000-sample segment) before converting it into a 64×64 pixel spectrogram image using the algorithm proposed in [21]. All created image sets will be classified randomly and labelled as representing in Table 2 to train and test the proposed TinyML model on the computer. During the training process on the computer side, the training data were split into two separate sets including 80% for training the model and 20% for validating the model.

$$D_s = \begin{cases} sgn(w)(|w| - T) & if\ |w| \geq T \\ 0 & if\ |w| < T \end{cases} \tag{1}$$

where, D_s is the wavelet coefficient after applying the soft threshold, *w* is the wavelet coefficient of the original signal, *T* is the soft threshold value, and *sgn* is the sign function.

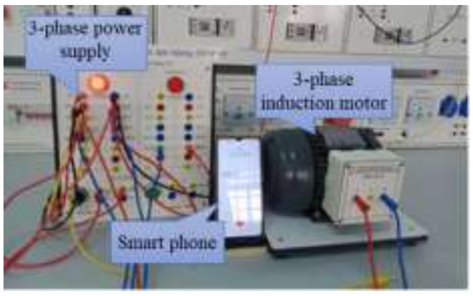

Figure 1. Motor's noise data logging experimental setup

Table 1. The parameters of using three-phase motors

Parameters	Motor 1	Motor 2	Motor 3	Motor 4
Max voltage (V)	220/380	220/380	220/380	220/380
Max current (A)	0.9	0.9	0.7	3
No-load current (A)	0.5	0.5	0.4	1.5
Speed (rpm)	1450	1390	1350	960
Rated power (kW)	0.37	0.37	0.18	1.5
Rated frequency (Hz)	50	50	50	50

Table 2. Data summary

Dataset	Training data	Testing data
Normal	10,500	2,250
Phase shift	10,500	2,250
Phase loss	10,500	2,250
Baring failure	10,500	2,250

2.2. Overview of process to build TinyML-based AC motor fault's classification

Figure 2 illustrates all procedures to build the motors' faults identification application. At first, the noise data of intentionally motor failures are acquired. After that, some pre-processing steps are applied to convert them to the grayscale Mel-spectrogram image sets. Then, these image sets are classified, labelled, and utilized to train and test the proposed TinyML model to classify them or identify the motor's faults.

Finally trained model will be converted to a TensorFlow Lite, quantized to reduce network capacity, and converted to a static array of C language before being integrated into a supported embedded platform. In this paper, the trained model was deployed on both our proposed hardware and nRF52840 MCU for validation. On both platforms, the proposed model will be tested with the live spectrogram image creating directly from the recorded noise data from a microphone.

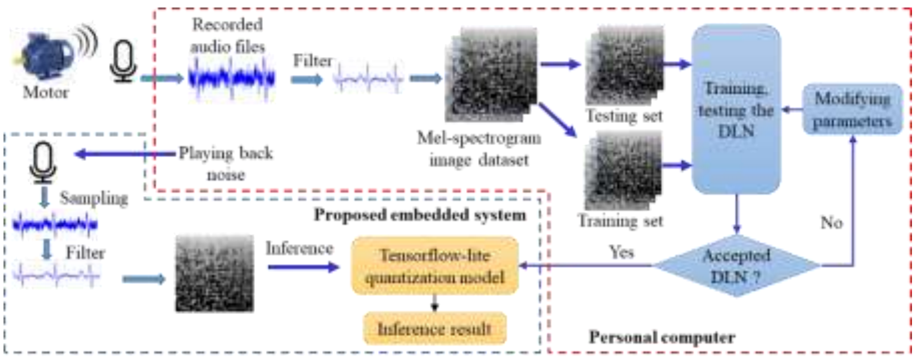

Figure 2. System architecture for faults identification system

2.3. Overview of the proposed TinyML model

In this study, the TinyML model is based on a customized DLN structure. Supporting that three common faults of the 3-phase motors are considered to identify including the normal operation (four cases of interest), regarding DLN has four labels $\mathcal{C} = \{c_1, c_2, c_3, c_4\}$ represent for a set motor operation noise (i.e., normal operation, phase shift, phase loss, and baring failure) is proposed to classify them. This DLN will be trained by a training set $\mathcal{X} = \{x_i, Y_i \mid 1 \leq i \leq m\}$, where $Y_i \subset \mathcal{C}$, to learn a multi-lable classifier to predict lables of new noise samples. As mentioned above, x_i is a set of two-dimensional (2D) grayscale Mel-spectrogram images of the operation noise. The proposed DLN with parameter Θ can be described as (2):

$$\mathcal{F}(\mathcal{X} \mid \Theta) = f_n(\ldots f_3(f_2(f_1(\mathcal{X} \mid \theta_1) \mid \theta_2) \mid \theta_3) \ldots \mid \theta_n) \tag{2}$$

where $f_j(\mathcal{X} \mid \theta_j)$ $(j \in n)$ represents the layer j^{th} of the network with total number of layer n. The proposed DLN structure in this study is illustrated in Figure 3. This is a 2D DLN has 64×64 input size, four outputs correspond to the four predicted cases of motor operation states including normal, phase shift, phase loss, and bearing failure. Feature extraction stage comprises of four 2D convolutional layers following by a 2D max polling for each layer. The convolutional layer can be formulated in the (3) as shown:

$$f_i(\mathcal{X}_i \mid \theta_i) = h(W * \mathcal{X}_i + b) \tag{3}$$

where, W is the set of kernels or filters, *h(.)* and b stand for activation function and bias value, respectively. The convolutional layers and the first fully connected (Dense) layer apply the redirected linear unit (ReLU) function which is formulated as (4).

$$h(x) = \max(0, x) \tag{4}$$

The last dense layer utilizes the softmax activation function for the multiclass classification with a posterior probability output in which the equation can be described in the (5):

$$f(z) = \frac{e^{z_i}}{\sum_j^C e^{z_j}} \tag{5}$$

where z_i is the inferred scores of each class in $\mathcal{C}$ by the DLN. The loss function uses the form of cross-entropy loss is formulated as (6):

$$Loss_{cross\_entropy} = -\log\left(\frac{e^{z_p}}{\sum_j^C e^{z_j}}\right) \tag{6}$$

where z_p is the score of the positive class. To prevent overfitting of the DLN, a Dropout layer is added right after the Flatten layer with a default dropout percentage of 50%. The proposed DLN has totally 163,588 training parameters after quantizing.

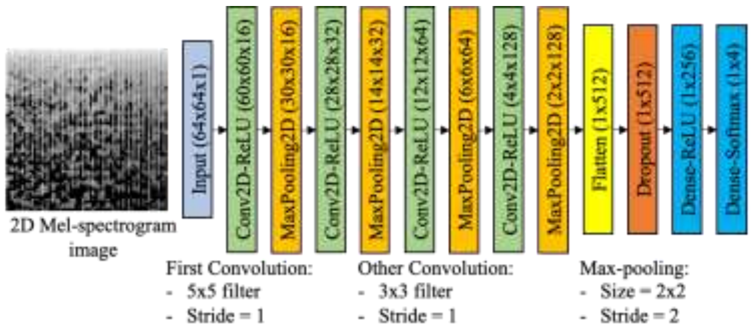

Figure 3. The architecture of proposed DLN

After training, the trained DLN was deployed on ESP32 and nRF52840 MCUs. The measured inference time of ESP32 and nRF52840, excluding the pre-processing time of new input data, is around 1.07 and 0.90 seconds, respectively. The algorithm of online single-task recognition program on these MCUs is demonstrated in Figure 4. It is clear to see that all tasks are executed sequentially, however the total inference time, including the pre-processing new data proceed, is very large and should be improved.

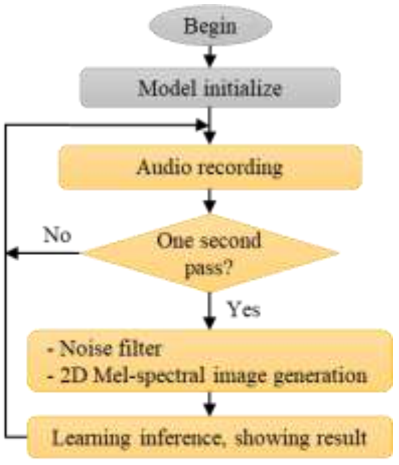

Figure 4. Sequential program algorithm

2.4. Proposed multiple MCU embedded platform

Due to the inference time of the trained DLN is nearly unable to optimize, our solution is to apply the FreeRTOS kernel to multitask the pre-processing new data on dual-core MCU. Simultaneously, other MCU will be predicting the previous processed data. It means that the best total inference time equal the DLN’s inference time plus with the data communication time. Consequently, Figure 5 depicts the block diagram of our proposed hardware. In fact, ESP32-1 is responsible for multi-tasking all pre-processing tasks and transmitting the result or spectrogram image to ESP32-2, while ESP32-2 is simultaneously identifying the previous image and showing the prediction on light-emitting diodes (LEDs). The motor operation noise is recorded by an IMNP441 omnidirectional micro-electro-mechanical systems (MEMS) microphone manufactured by InvenSen Inc [24].

However, the fault detection is tested with two scheduling schemes of ESP32-1 including single task and multi-task to clarify the advantage of our proposed platform. For the ESP32-2, it is mainly waiting for the spectral image from the ESP32-1 to recognize, therefore the multi-tasking is unnecessary. The timing diagram of executed tasks on two ESP32s is shown in Figure 6. For Case a, ESP32-1 only runs single-core, or sequential tasks, but it can take the advantage of the direct memory access (DMA) controller to buffer one second sound data. In the first recognized cycle, the recognition time is longer, since ESP32-2 does not have spectral image to recognize. However, from the second period onwards, the time between two printing results (t_a) is shorter as ESP32-2 and ESP32-1 execute simultaneously. In this case, it is possible to utilize two cores on ESP32-1 to perform the whole application concurrently, unfortunately, ESP32 MCU have insufficient memory for the most cases.

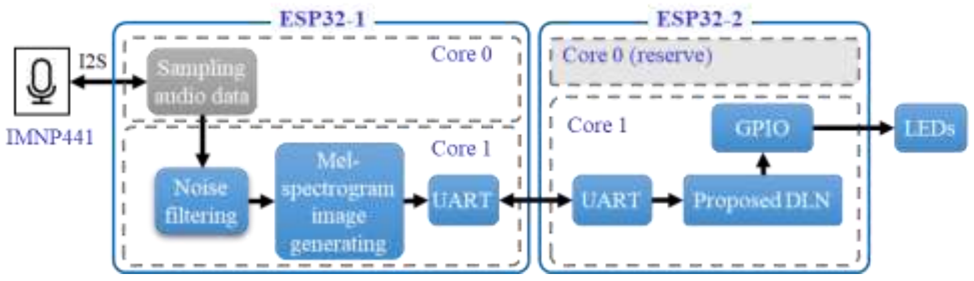

Figure 5. Hardware block diagram of the proposed embedded hardware

For Case b, ESP32-1 executes parallelly two tasks on Core 0 and 1. One task is responsible for reading data periodically from the buffer of the DMA controller, while the other task carries out denoising new audio data, generating and transmitting Mel-spectrograms to the ESP32-2 for model inference. For the first identification cycle, the inference result should be shown out at the same time with Case a. However, from the second cycle onwards, the time between two printing recognition outcomes (t_b) is equal to the total of transmitted data and inference time.

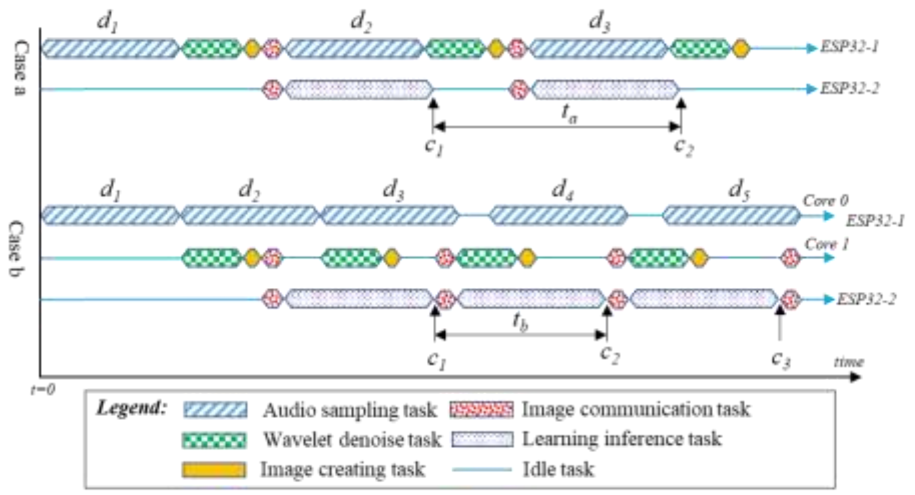

Figure 6. Tasks scheduling scheme

2.5. System evaluation

To evaluate the TinyML DLN on the proposed hardware platform, 400 segments of live audio records were tested for each fault (i.e., phase shift, phase loss, and baring failure) and normal operation case. These are total 16,000 noise segments were tested to evaluate the DLN performance. The average inference time of all cases is used to evaluate the total inference time. Besides, the TinyML DLN is also evaluated by using different performance metrics, e.g., accuracy, precision, recall, and F1 value. They are explained in (7) to (10), respectively.

$$Accuracy = \frac{TP+TN}{Total\ samples} \tag{7}$$

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

$$F1 = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{10}$$

where, true positive (TP) and true negative (TN) are precisely predicted as the positive class and negative class, correspondingly, whereas false positive (FP) and false negative (FN) is improperly projected with the positive class and negative class, respectively. In this study, macro averaged value of accuracy, precision, recall, and F1 are used to evaluate the proposed DLN based on model's inference results with real-time data.

3. RESULTS AND DISCUSSION

3.1. The efficiency of the proposed platform

Figure 7 shows the experimental layout for real-time testing the trained DLN on our proposed hardware. Since there are no engines to test, the recorded noise data was played through a high-definition speaker to imitate sufficiently the real engine operation noise. The microphone is placed in front of the speakers to record the sound and identify the type of noise or the motor fault. Four LEDs are used to indicate the prediction fault. Furthermore, data samples such as the recorded noise, filtered noise, Mel-spectrogram image, and identification outputs were also printed to the Serial Monitor window for collecting and analyzing afterward. The executed time of a particular task is measured by the deviation time of the system timer at the begin and end of the task.

Table 3 indicates the executed time of the tasks on the experimental system. One-second audio data is sampled by IMNP441 at the sampling rate of 16 kHz and sent back as a data stream on the Inter-IC sound (I2S) bus. For evaluating the performance, the parameters of wavelet filter applied on both computer and embedded side are similar, such as using the *sym4* wavelet function, the soft threshold determined by the *sureshrink* method [25]. In addition, the wavelet filter library developed by Hussain [26] has been modified to be compatible for running on the embedded system.

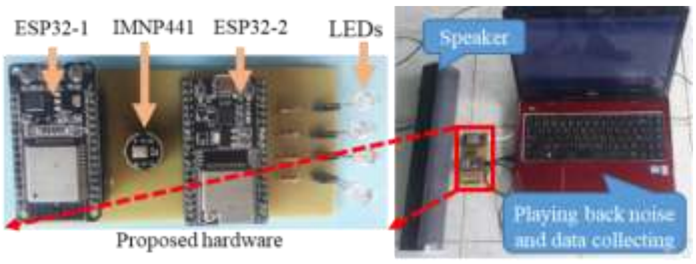

Figure 7. Real-time testing setup

Table 3. The executing time (seconds) of implemented tasks on the proposed hardware

Task	ESP32	nRF52840
Audio recording	1.00	1.00
Wavelet denoise	0.44	2.52
Spectrogram image generating	0.13	0.51
UART communication	0.15	-
DLN learning inference	1.07	0.90
Total inference time on single core MCU	-	4.93
Total inference time of Case a (t_a)	1.87	-
Total inference time of Case b (t_b)	1.22	-

Table 3 also shows the total inference time of the TinyML DLN on our proposed hardware and single core MCU. The average time between two consecutive identifications is 1.87 seconds associated with the ESP32-1 running in single core, compare to 1.22 seconds for multi-core running that enhanced 34.8%. Besides, the program is also modified to test on a single-core MCU nRF52840, the inference time of DLN is 0.90 seconds compared to 1.07 seconds from ESP32, nevertheless, the total inference time including pre-processing new data is up to 4.93 seconds.

Consequently, our proposed hardware combining with the multi-tasking programming technique is remarkably improved the speed of the AC motor faults classifier. However, Figure 6 also shows that the cores of ESP32-1 and ESP32-2 have not fully executed, on which the application scalability of the system is obviously remaining. The idle time of these cores could be exploited to conduct the additional tasks such as communicating with the cloud server or other edge devices on the IoT system. Besides, the averaged consumption current during inference time is only 165.9 mA and can be lower if the low power modes of ESP32 MCUs were applied. So, it is feasible to apply DLNs to analyze and make decisions directly at IoT devices.

3.2. Evaluation of real-time fault classification

The proposed DLN in Figure 3 has been trained on the computer using the training data in Table 2. The implementation of the DLN and training were done on the Google Colab. Figure 8 shows the accuracy and loss of training process. The training accuracy is up to 97.7 % after 300 epochs, the total training time is 3 hours and 35 minutes on the Colab. The testing accuracy is 95.6 % using the testing data in Table 2.

The accuracy of the trained TinyML DLN on the proposed embedded system was also evaluating using the same experimental layout illustrated in the Figure 7. The computer randomly plays about 400-second noise data of each data set via speaker for the real-time classification. Each sound type was identified 400 times. The experimental result shows that the macro accuracy, precision, recall, and F1 values are 83.7%, 85.2%, 83.7%, and 84.4%, respectively. This accuracy is significantly proved in comparing with the similar DLN model and pre-processing procedures but applied on only one motor in [20]. Figure 9 presents the confusion matrix of real-time inference of all cases.

3.3. Discussion

The work has successfully proposed an embedded hardware platform to apply the multi-tasking programming technique for improving the speed of the TinyML-oriented application. Furthermore, some pre-processing data procedures, such as wavelet denoising and grayscale Mel-spectrogram imaging, are successfully implemented and optimized execution period on the proposed hardware. The preliminary results promise many future applications, especially in integrating the TinyML to IoT edge devices. In addition, this study has also initially solved the energy problem for IoT devices integrated artificial intelligence that mainly operate with the renewable energy sources. Indeed, tiny MCUs like ESP32 or nRF52840 consume very low power and support power-saving modes, while many embedded computers deploying popular DLN platforms are high energy consumption that is impractical for renewable energy sources.

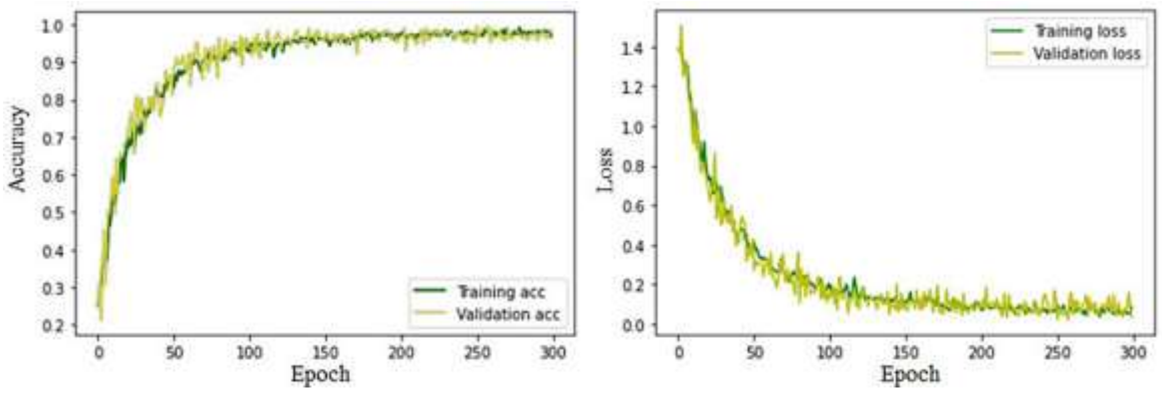

Figure 8. Training and validation loss of the proposed DLN

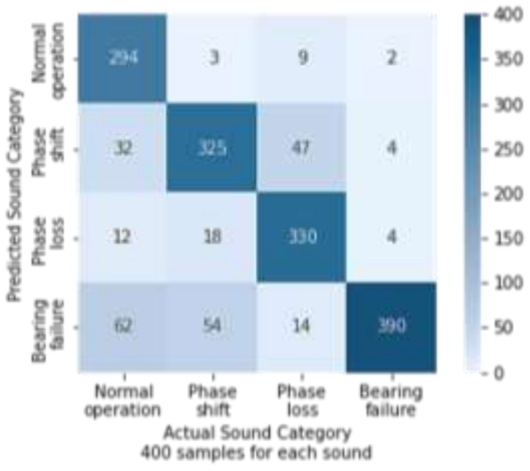

Figure 9. Confusion matrix of real-time classification

4. CONCLUSION

Multiple MCU-based hardware platforms have been successfully proposed to take advantage of the multi-tasking programming techniques to speed up the TinyML-oriented application. The total inference time of a TinyML network on our platform is decreased by 34.8% in comparison with a single processor platform. Besides, the proposed DLN has been modified to classify well the faults of four three-phase motors with the best accuracy of real-time inference (detect the live audio samples recorded directly by MCU) is 83.7%. In future work, our hardware platform can be utilized to directly implement artificial intelligence to IoT devices without computation tasks on the computer or cloud server.

ACKNOWLEDGEMENTS

The authors would like to thank Mr. Van-Tung Hoang, lecturer at Bac Lieu Vocational College, who carried out the data acquisition and shared the audio data set to help us complete this study.

REFERENCES

[1] 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, May 2022, doi: 10.3390/mi13060851.

[2] N. Schizas, A. Karras, C. Karras, and S. Sioutas, "TinyML for ultra-low power AI and large scale IoT deployments: A systematic review," *Future Internet*, vol. 14, no. 12, Dec. 2022, doi: 10.3390/fi14120363.

[3] P. P. Ray, "A review on TinyML: State-of-the-art and prospects," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 4, pp. 1595–1623, Apr. 2022, doi: 10.1016/j.jksuci.2021.11.019.

[4] R. Mohandas, M. Bhattacharya, M. Penica, K. Van Camp, and M. J. Hayes, "On the use of deep learning enabled face mask detection for access/egress control using TensorFlow Lite based edge deployment on a Raspberry Pi," in *2021 32nd Irish Signals and Systems Conference (ISSC)*, Jun. 2021, pp. 1–6, doi: 10.1109/ISSC52156.2021.9467841.

[5] P. Sertic, A. Alahmar, T. Akilan, M. Javorac, and Y. Gupta, “Intelligent real-time face-mask detection system with hardware acceleration for COVID-19 mitigation,” *Healthcare*, vol. 10, no. 5, May 2022, doi: 10.3390/healthcare10050873.

[6] S. Ryu and S.-C. Kim, “Embedded identification of surface based on multirate sensor fusion with deep neural network,” *IEEE Embedded Systems Letters*, vol. 13, no. 2, pp. 49–52, Jun. 2021, doi: 10.1109/LES.2020.2996758.

[7] TensorFlow, “Deploy machine learning models on mobile and edge devices.” TensorFlow, https://www.tensorflow.org/lite (accessed Dec. 15, 2022).

[8] M. Hussein, Y. S. Mohammed, A. I. Galal, E. Abd-Elrahman, and M. Zorkany, “Smart cognitive IoT devices using multi-layer perception neural network on limited microcontroller,” *Sensors*, vol. 22, no. 14, Jul. 2022, doi: 10.3390/s22145106.

[9] I. Choi and H. Kim, “Reducing energy consumption and health hazards of electric liquid mosquito repellents through TinyML,” *Sensors*, vol. 22, no. 17, Aug. 2022, doi: 10.3390/s22176421.

[10] M. de Prado, M. Rusci, A. Capotondi, R. Donze, L. Benini, and N. Pazos, “Robustifying the deployment of tinyML models for autonomous mini-vehicles,” *Sensors*, vol. 21, no. 4, Feb. 2021, doi: 10.3390/s21041339.

[11] J. Kwon and D. Park, “Hardware/software co-design for TinyML voice-recognition application on resource frugal edge devices,” *Applied Sciences*, vol. 11, no. 22, Nov. 2021, doi: 10.3390/app112211073.

[12] K. Kopparapu and E. Lin, “TinyFedTL: Federated transfer learning on tiny devices,” *Prepr. arXiv.2110.01107*, Oct. 2021.

[13] B. Sudharsan, J. G. Breslin, and M. I. Ali, “ML-MCU: A framework to train ML classifiers on MCU-based IoT edge devices,” *IEEE Internet of Things Journal*, vol. 9, no. 16, pp. 15007–15017, Aug. 2022, doi: 10.1109/JIOT.2021.3098166.

[14] B. Sudharsan, P. Yadav, J. G. Breslin, and M. Intizar Ali, “Train++: An incremental ML model training algorithm to create self-learning IoT devices,” in *2021 IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/IOP/SCI)*, Oct. 2021, pp. 97–106, doi: 10.1109/SWC50871.2021.00023.

[15] B. Sudharsan, J. G. Breslin, and M. Intizar Ali, “Globe2Train: A framework for distributed ML model training using IoT devices across the globe,” in *2021 IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/IOP/SCI)*, Oct. 2021, pp. 107–114, doi: 10.1109/SWC50871.2021.00024.

[16] E. Lamie, *Real-time embedded multithreading using ThreadX*. CRC Press, 2019.

[17] R. V Aroca and G. Caurin, “A real time operating systems (RTOS) comparison,” *WSO-Workshop de Sistemas Operacionais*, no. 12, pp. 2441–2452, 2009.

[18] R. Barry, “Mastering the freertos real time kernel,” *Real Time Engineers Ltd*, 2016.

[19] M. Z. H. Zim, “TinyML: Analysis of Xtensa LX6 microprocessor for Neural Network Applications by ESP32 SoC,” *Prepr. arXiv.2106.10652*, Jun. 2021, doi: 10.13140/RG.2.2.28602.11204.

[20] V. K. Nguyen, V. K. Tran, M. K. Nguyen, V. T. E. Thach, T. L. H. Pham, and C. N. Nguyen, “Realtime non-invasive fault diagnosis of three-phase induction motor,” *Journal of Technical Education Science*, no. 72B, pp. 1–11, Oct. 2022, doi: 10.54644/jte.72B.2022.1231.

[21] P. Warden and D. Situnayak, *Tinyml: Machine learning with tensorflow lite on arduino and ultra-low-power microcontroller*. O’Reilly Media, 2019.

[22] H. Van Tung, N. Van Khanh, and N. C. Ngon, “Proposal of noninvasive failure diagnosis of electrical motor using googlenet,” *Journal of Technical Education Science*, no. 66, pp. 83–93, Oct. 2021, doi: 10.54644/jte.66.2021.1070.

[23] A. A. Jaber and R. Bicker, “Real-time wavelet analysis of a vibration signal based on arduino-UNO and LabVIEW,” *International Journal of Materials Science and Engineering*, pp. 66–70, 2015, doi: 10.12720/ijmse.3.1.66-70.

[24] A. Devices, “Omnidirectional microphone with bottom port and I2S digital output,” *ADMP421, Data Sheet*, 2012.

[25] D. L. Donoho and I. M. Johnstone, “Adapting to unknown smoothness via wavelet shrinkage,” *Journal of the American Statistical Association*, vol. 90, no. 432, pp. 1200–1224, Dec. 1995, doi: 10.1080/01621459.1995.10476626.

[26] R. Hussain, “wavelib,” Accessed: Sep. 12, 2022. [Online]. Available: https://github.com/rafat/wavelib.

BIOGRAPHIES OF AUTHORS

Van-Khanh Nguyen received his master’s degree from Ho Chi Minh University of Technology, Vietnam in 2014 and Doctor of Engineering degree from Tokyo University of Marine Science and Technology, Japan in 2020. Since 2007, he has been a lecturer at Faculty of Automation Technology, College of Engineering, Can Tho University. Currently, he is a head of PLC Technology and Industrial IoT Lab. His research interests concentrate on embedded systems, AIoT- and IoT-based applications in environmental and agricultural control. He can be contacted at email: vankhanh@ctu.edu.vn.

Vy-Khang Tran is a B.S. degree student in Automation and Control Engineering of the Faculty of Automation Technology, College of Engineering, Can Tho University, Vietnam. He will graduate his B.S. degree at the end of December 2022. He can be contacted at email: tranvykhang1906@gmail.com.

Hai Pham received his master’s degree from University of South Australia (UniSA) in 2010. Since 2012, he has been a lecturer at Faculty of Automation Technology, College of Engineering, Can Tho University. His research interests focus on Bistatic LIDAR system for gas measurement in environmental and agricultural applications. He can be contacted at email: ptlhai@ctu.edu.vn.

Van-Muot Nguyen graduated a Bachelor degree in Electronics engineering in 1998 from Can Tho University, and he received a Master degree of Automation in 2009 at the University of Transport-Ho Chi Minh city, Vietnam. In 2020, he was awarded a Ph.D degree in Control engineering from the University of Rostock, Germany. He is currently a chief of Sensors and Measurements Lab and also a lecturer in College of Engineering, Can Tho University. Some of his interested subjects and research are related to automation control theory, advanced control theory, neural network control, fuzzy control, and medical device control. His official email to be contacted is nvmuot@ctu.edu.vn.

Hoang-Dung Nguyen is a senior lecturer and head of the Department of Automation Technology, Can Tho University. Additionally, he is director of the Innovation Center of Can Tho University. He earned his master degree on automation engineering from Ho Chi Minh University of Technology and obtained his Ph.D. on Cogno-mechatronics engineering from Pusan National University, Republic of Korea. His research is mainly focused on artificial intelligence, brain engineering, 3D brains imaging, entrepreneurship, control engineering. He can be contacted at email: hoangdung@ctu.edu.vn.

Chi-Ngon Nguyen received B.S. and M.S. degrees in Electronic Engineering from Can Tho University and the National University, Ho Chi Minh City University of Technology, Vietnam, in 1996 and 2001, respectively. The degree of Ph.D. in Control Engineering was awarded by the University of Rostock, Germany, in 2007. Since 1996, he has worked at the Can Tho University. He is an associate professor in automation at Faculty of Automation Technology, and former dean of the College of Engineering at the Can Tho University. Currently, he is a Vice Chairman of the Board of Trustee of Can Tho University. His research interests are intelligent control, medical control, pattern recognition, classifications, speech recognition, computer vision and agricultural automation. He can be contacted at email: ncngon@ctu.edu.vn.