Low-power EEGNet-based Brain-Computer Interface implemented on an Arduino Nano 33 Sense

Daniel Enériz
Group of Power Electronics and Microelectronics (GPEM)
Aragon Institute of Eng. Research (I3A)
University of Zaragoza
Zaragoza, Spain
eneriz@unizar.es

Nicolás Medrano
Group of Power Electronics and Microelectronics (GPEM)
Aragon Institute of Eng. Research (I3A)
University of Zaragoza
Zaragoza, Spain
nmedrano@unizar.es

Belén Calvo
Group of Power Electronics and Microelectronics (GPEM)
Aragon Institute of Eng. Research (I3A)
University of Zaragoza
Zaragoza, Spain
becalvo@unizar.es

Diego Antolín
Group of Power Electronics and Microelectronics (GPEM)
Aragon Institute of Eng. Research (I3A)
University of Zaragoza
Zaragoza, Spain
dantolin@unizar.es

2023 38th Conference on Design of Circuits and Integrated Systems (DCIS) | 979-8-3503-0385-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/DCIS58620.2023.10335978

***Abstract*—The use of Convolutional Neural Networks (CNNs) to process Electroencephalograph (EEG) signals has been introduced in recent years with great success in the field of Brain-Computer Interfaces (BCI). Nevertheless, in order to advance towards a CNN-based BCI prototype, they must be efficiently mapped into low-power and low-cost hardware, enabling a real-time, portable and Internet-independent brain-computer communication. This work presents the implementation of an EEGNet-based model into an ARM Cortex M4F microcontroller, available on the Arduino Nano 33 Sense. Starting from models trained over the Physionet Motor Movement/Imagery dataset, 8-bit integer post-training quantization has been considered to reduce computing complexity, with a mean downgrade of 2.64±0.77% in accuracy. Moreover, their computational impact and memory footprint have been characterized by measuring the associated operations and Random-Access Memory (RAM) usage. Finally, a selected model has been implemented on the ARM Cortex M4F, with a latency of 137 ms and an energy per inference of 2.55 mJ, a 40% lower than other EEGNet implementation on the same microcontroller.**

Keywords—EEGNet, Brain-Computer Interfaces, Convolutional Neural Networks, Edge computing, Arduino

I. Introduction

Brain-Computer Interfaces (BCIs) offer a straightforward communication between the human brain and machines by reading brain signals, classifying them, and triggering an actuator or an action. BCIs have medical applications in communication for disabled people, moods and emotions identification, and sleep stages analysis, as well as other fields like gaming, home automation, and human enhancement [1]–[6]. One of the most popular BCI paradigms is Motor Imagery (MI) activity identification [7], which involves recognizing intended movements that are not physically executed, but whose chemoelectrical activity can be used to trigger different actions. Electroencephalographs (EEGs) are signals related to this chemoelectrical activity, typically used since are non-invasive and easy-to-acquire through electrodes placed in the subject's external scalp [8].

This work has been supported through the grants PID2019-106570RB-I00 (AEI/10.13039/501100011033), PID2022-138785OB-I00 (MCIN/AEI/10.13039/501100011033/FEDER, UE) and the Government of Aragon PhD grant BOA20201210014.

Traditionally, EEG signals have been processed using machine learning techniques, such as Common Spatial Pattern (CSP), Independent Component Analysis (ICA), Support Vector Machines (SVM), Linear Discriminant Analysis (LDA), k-Nearest-Neighbor (kNN) or Decision Trees (DT) [7], [9]–[12]. However, recent works have shown that Convolutional Neural Networks (CNNs) can achieve similar or even higher performance than traditionally used EEG processing algorithms [13]–[18]. CNNs are designed to process large amounts of raw data with minimal human intervention, making them particularly useful for processing EEG signals. Among them, the EEGNet is a remarkable example, as it is a compact CNN capable of learning abstract features over a range of different EEG-based BCI paradigms, while achieving high performance, similar to other preceding reference algorithms [19].

But while CNNs show up many advantages, one potential drawback is that they are computationally intensive and memory hungry. As a result, they are typically executed on cloud or high-performance platforms. However, for BCI applications, a real-time, portable, Internet-independent, low-cost, and low-power implementation is necessary. Therefore, implementing a CNN on small hardware satisfying all these features is of great interest. Besides, such an implementation would also preserve user privacy, as data would be acquired and processed in the same device.

In this sense, as it is compact and end-to-end, the EEGNet is well-suited for deployment on embedded devices. Reviewing the literature, an EEGNet-based model was implemented in ARM Cortex M4F and M7 Microcontroller Units (MCU) in [20], achieving latencies of 101 ms and 44 ms, and energy consumptions per inference of 4.28 mJ and 18.1 mJ, respectively. In [21], the same research group introduced an 8-bit EEGNet-based model implementation on a Parallel Ultra-Low Power (PULP) System on-Chip (SoC) using custom RISC-V extensions, reducing the latency up to 5.82 ms with 0.627 mJ energy consumption. There are also Field-Programmable Gate Arrays (FPGA) implementations, where the latency varies from tens of milliseconds in a Xilinx Zynq 7020 SoC [22] up to tens of microseconds in a Xilinx Virtex-7 [23]. Finally, there is also an NVIDIA Jetson TX2

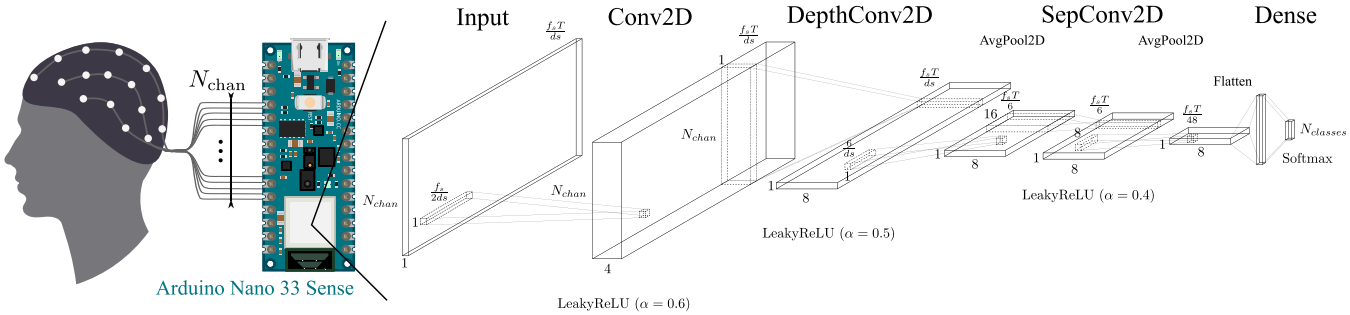

Fig. 1. Arduino-based Brain-Computer Interface scheme. The EEGNet-based model is implemented on the ARM Cortex M4F MCU available on the Arduino Nano 33 Sense. The shapes of the feature maps and kernels are marked, showing their dependence on the input data parameters.

TABLE I. MODEL ARCHITECTURE DESCRIPTION

Layer	# filters	Padding	Kernel	# params.	Activation	Output shape
Input	-	-	-	-	-	(N_{chan}, fs·T/*ds*, 1)
Conv2D	4	same	(1, *fs*/2*ds*)	2*fs*/*ds*	LeakyReLU (α=0.6)	(N_{chan}, *fs*·*T*/*ds*, 4)
DepthConv2D	2·4	valid	(N_{chan}, 1)	8N_{chan}	LeakyReLU (α=0.5)	(1, *fs*·*T*/*ds*, 8)
AvgPool2D	-	valid	(1, 6/*ds*)	-	-	(1, *fs*·*T*/6, 8)
SepConv2D	8	same	(1, 16)	192	LeakyReLU (α=0.4)	(1, *fs*·*T*/6, 8)
AvgPool2D	-	valid	(1, 8)	-	-	(1, *fs*·*T*/48, 8)
Flatten	-	-	-	-	-	*fs*·T/6
Dense	$N_{classes}$	-	-	$N_{classes}$ *fs*·*T*/6	Softmax	$N_{classes}$

EEGNet-based model implementation that reports a latency of 48.7 ms.

This work presents an 8-bit implementation of a EEGNet-based model over the ARM Cortex M4F MCU using the Arduino Nano 33 Sense and the TensorFlow Lite framework, which, compared to other implementation on the same MCU, exhibits a similar latency in clock cycles and a decrease of 40% in energy consumption per inference. The dataset used in this work is presented in Section II. Section III introduces the model and the training procedure. Section IV includes the implementation strategy and the obtained results. The power consumption characterization method is available in Section V. Finally, some conclusions are drawn in Section VI.

II. DATASET

The dataset used in this work to train and evaluate the MI BCI models is the Physionet Motor Movement/Imagery dataset [24]. It contains 64-channel EEG recordings of 105 subjects sampled at 160 Hz during four different motor imagery tasks: opening and closing the left fist (L), opening and closing the right fist (R), opening and closing both fists (B), and opening and closing both feet (F). Additionally, there are baseline data that allows the formation of a resting class (0). In this work we only consider the $N_{classes} = 4$ scheme, that uses the L/R/0/F MI tasks, as done in [17], [20], [22].

Each EEG recording is related to the chemoelectrical activity during a 4 s stimulus in which the subject was simulating the corresponding MI movement. Therefore, each sample has a maximum shape of 64×640 corresponding to an EEG signal sampled with 64 channels during 4 s at 160 Hz. In order to provide a way to control the model size, three input data reduction parameters introduced in [20] which were used by authors in [22] are considered: 1) downsampling the recordings in a factor $ds \in \{1, 2, 3\}$; 2) using subsets of the channels, with $N_{chan} \in \{64, 38, 19, 8\}$; and 3) shortening the input time window to $T = \{3, 2, 1\}$.

Additionally, the data augmentation technique presented in [22] is also used, where the frames that would be discarded by the downsampling are used to form new recordings.

III. MODEL: ARCHITECTURE AND TRAINING

Fig. 1 shows the model architecture used in this work. It is the same introduced by authors in [22]. Basically, it is the EEGNet [19] but adapted to the input data reduction parameters of the Physionet Motor Movement/Imagery dataset, with LeakyReLU activations instead of ELU activation and without Batch Normalization and Dropout layers. These modifications allow to reduce the computational impact of the model, enabling its execution in low-end hardware. The details of the architecture layers are shown in Table I.

The validation strategy described on [17], [20], [22] is followed in this work. Firstly, samples are divided in 5 folds by subjects, i.e., all the samples of a certain subject are placed in the same fold. This allows to evaluate a global model with 5-fold Cross-Validation (CV), resulting in 5 models trained on an iterating 80% of subjects and validated with the remaining 20%. In order to obtain a unique figure-of-merit of each global model, the average of the validation accuracies of the 5 models is obtained. Finally, transfer learning is used to obtain subject-specific models. In this case, data of a certain subject are separated in 4 folds with equal class balance. The model obtained from the global validation fold including the subject is used as the starting point to get each subject-specific model. Thus, 4-fold CV is used to evaluate the subject-specific models. The average of the validation accuracy across the subjects and the folds is used as a general metric of the subject-specific model.

Training process is implemented with the high-level framework Keras [25] with Tensorflow 2.8 backend. The global models are trained with the Adam optimized for 100 epochs with batch size of 16 and a learning rate of 10^{-2}. Subject-specific models are trained for 5 more epochs, with the same training procedure and hyperparameters. In both stages, the model with best validation accuracy is saved. A 16-core Intel i7-10700 at 2.90 GHz with a NVIDIA Quadro P2000 GPU is used to launch these trainings.

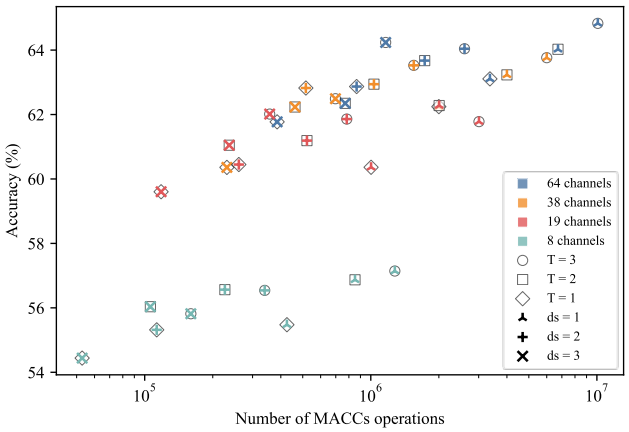

Fig. 2. Accuracy of the global models versus their number of Multyply-Accumulate (MACC) operations. The shape of the thin marker indicates the time window length in seconds, the shape of the bold marker is the downsampling factor and its color shows the number of channels. Note that the x-axsis is in log scale.

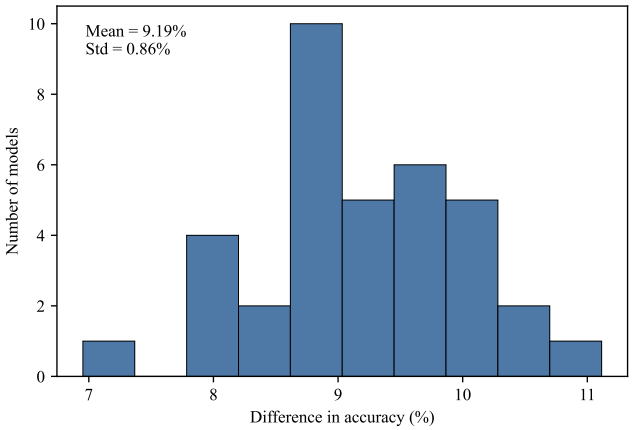

Fig. 3. Histogram of the improvement achieved by subject-specific models compared with the global model.

Fig. 2 includes the distribution of the 36 models resulting from the input data reduction parameters combination in terms of accuracy and Multiply-Accumulate (MACC) operations. As it can be noticed, the number of channels has coarse effects on model accuracy, with $N_{chan} = 8$ being an extreme case where the model performance drops significantly. On the contrary, T and ds enable slight control of the model accuracy while significantly affecting on the number of MACC operations, which is an illustration of the model complexity in terms of computational impact.

Finally, Fig. 3 contains the histogram of the accuracy differences between the global and the subject-specific models, showing an average improvement of 9.19±0.86%. This proves the variability of the EEG signals between subjects, justifying the development of personalized EEG-based models for MI BCI.

IV. MCU Implementation

As commented in Section I, the target model implementation device is the ARM Cortex M4F MCU [26], a low-cost and low-power processor available within the Arduino Nano 33 Sense board [27], which also has a 256 KB Random Access Memory (RAM) and a 1 MB flash memory. Additionally, among other peripherals, there is a 12-bit 200 kSa/s Analog to Digital Converter (ADC) with up to 8 channels. These characteristics of the Arduino Nano 33 Sense

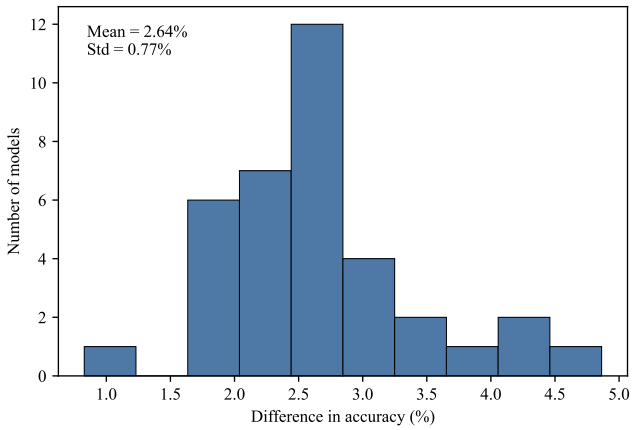

Fig. 4. Histogram of global models downgrade in accuracy due to the 8-bit representation compared to the floating-point representation.

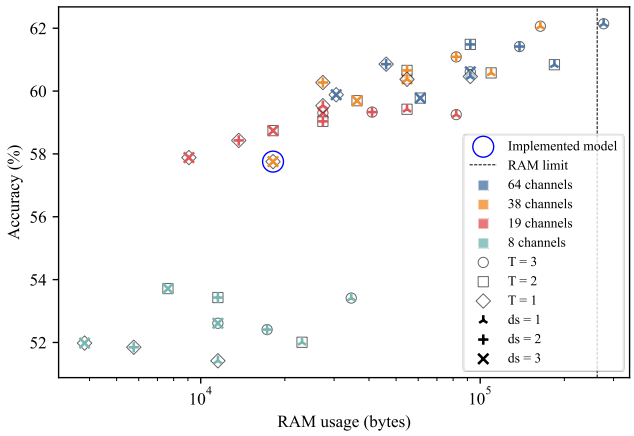

Fig. 5. Accuracy of the quantized global models versus their Random Access Memory (RAM) consumption on the Arduino Nano 33 Sense, whose RAM limit is marked in the dashed vertical line. The shape of the thin marker indicates the time window length in seconds, the shape of the bold marker is the downsampling factor and its color shows the number of channels. The circle in blue marks the implemented model, the same as in [20] for comparison purposes. Note that the x-axis is in log scale.

make them a suitable device for our purpose of implementing a real-time EEGNet-based BCI prototype.

Tensorflow Lite 2.8 is used to map the high-level model described in Keras to a description compatible with this low-end hardware. Also, it enables the optimization of the model for inference. In this work 8-bit integers are used during inference to represent all the model weights, activations, and feature maps, reducing the memory footprint in a factor of 4 and speeding up the inference runtime. Due to the quantization effect that appears because of the reduced resolution –compared to the 32-bit floating point used during training– a downgrade in model performance is expected. Fortunately, this tool allows to obtain the quantized model accuracy without implementing the model on the MCU. Fig. 4 shows the histogram of the differences in accuracy between the floating-point global models and their analogues using an 8-bit quantization, with a mean downgrade of 2.64±0.77%.

Although this architecture is compact and has a small number of weights (for $N_{chan} = 64$, $T = 3$, and $ds = 1$ there are 1,348 weights), the number of feature maps is still relatively high. For this reason, the RAM is expected to be the limiting resource in this implementation. In other to verify this, we have characterized the required RAM for inference on the ARM Cortex M4F in each model. Fig. 5 contains these results,

TABLE II. MODEL $N_{CHAN} = 38$, $T = 1$, $DS = 3$ IMPLEMENTATION COMPARATION

	Accuracy (%)	Latency (ms)	Clock cyc./inf. ($\cdot 10^6$)	En./inf. (mJ)
Wang et al. [20]	62.51	101	8.08	4.28
This work	57.76	137	8.77	2.55

also including the accuracy of the quantized global models. As it is shown, all the models fit on the Arduino Nano 33 Sense RAM, except for the biggest model with $N_{chan} = 64$, $T = 3$, and $ds = 1$ parameters. Additionally, it can be noticed that the RAM approximately scales linearly with N_{chan} and T, and inversely to ds, according to the following numerical approximation:

$$\mathrm{RAM}_{\mathrm{EEGNet}} \cong 1{,}440 \cdot N_{\mathrm{chan}} \cdot T \,/\, ds \tag{1}$$

Finally, in order to properly compare our work with the existing ARM Cortex M4F implementation [20], the same model is considered for its fully implementation: $N_{chan} = 38$, $T = 1$, and $ds = 3$, which is the one circled in blue in Fig. 5. The implementation comparison is available on Table II. Our model has 4.75% lower accuracy, significantly reduced due to the 8-bit integer quantization. The latency of the model was measured using the clock of the Arduino Nano 33 Sense, obtaining 137 ms, 36% slower than the other implementation. It must be remarked that the clock in [20] is 80 MHz while in the Arduino Nano 33 Sense the clock is 64 MHz. If we compare the number of clock cycles it takes to have an inference, our implementation only has 8% more cycles.

V. ENERGY CONSUMPTION CHARACTERIZATION

In this section, the experiment to characterize the energy consumption per inference is described. The setup is included in Fig. 6. The Tektronix DPO4104 oscilloscope is used to measure the current of the Arduino Nano 33 Sense USB power supply through the Tektronix TCP0030 Hall-effect probe. Additionally, a probe is monitoring the USB supply voltage. Then, an arbitrary number of inferences separated by a 30 ms delay are launched in the Arduino. The current profile of a single inference is available in Fig. 7.

To determine if the Arduino is at inference or not, a smoothed version of the current is obtained. Concretely, a Savitzky–Golay [28] filter of 100 coefficients and polynomial order of 1 is used. The obtained filtered version of the current allows the determination of a current threshold, fixed to 27.7 mA. Thus, if for a certain time point the smoothed current is greater than the threshold, the processor is considered to be under inference at that time point. If it is not, the processor is in passive mode, i.e., in a delay between inferences.

Then, the raw current and the measured voltage are multiplied to obtain the instantaneous power. Its average on the time points at inference and in passive mode provide respectively the inference power and average passive power, whose values are $\overline{P_{\mathrm{inf}}} = 145 \pm 29$ mW and $\overline{P_{\mathrm{pass}}} = 127 \pm 26$ mW. Then, the energy per inference can be computed as:

$$E_{\mathrm{inf}} = \Delta t \cdot \left(\overline{P_{\mathrm{inf}}} - \overline{P_{\mathrm{pass}}} \right) \tag{2}$$

where Δt is the latency, 137 ms. Therefore, the energy per inference estimation is 2.55 ± 5.35 mJ, 40% less than the energy reported in [20]. The high uncertainty is due to the high variability on the current measurement, as shown in Fig. 7.

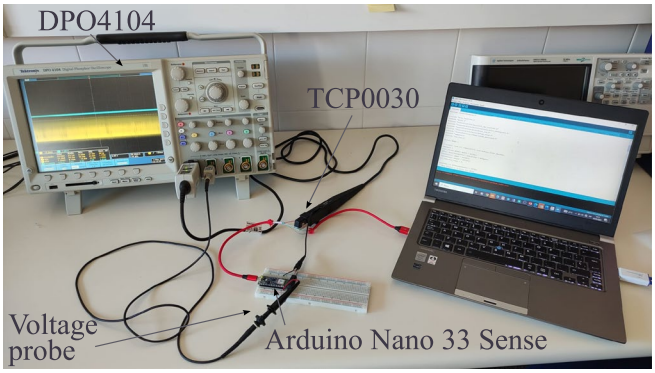

Fig. 6. Energy consumption characterization setup. A Tektronix DPO4104 oscilloscope is used to monitor the Arduino USB voltage and current, this last one with a Tektronix TCP0030 Hall-effect current probe. The laptop is used to launch inferences.

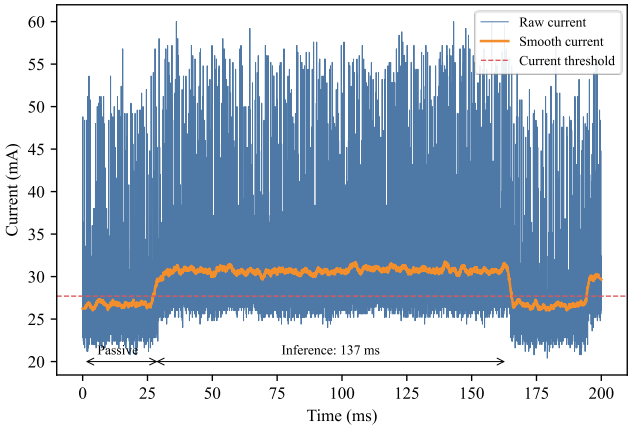

Fig. 7. Current profile of an inference. The measured current is show in blue; a smoothed version, after a Savitzky–Golay filter, is shown in orange. The red dashed red line sets the threshold to determine if the Arduino Nano 33 Sense is at inference.

VI. CONCLUSIONS

In this work a EEGNet-based model is implemented on an Arduino Nano 33 Sense for MI task identification, paving the way towards the development of a BCI under the MI paradigm using EEG signals. The model architecture is firstly introduced. It is an adaptation of the EEGNet [19] to work with the Physionet Motor Movement/Imagery dataset and reducing its computational impact by removing layers and changing the activation to the LeakyReLU. Additionally, the input data parameters introduced in [20] are used to control the computing intensity and the memory footprint, as shown in Fig. 2 and Fig. 5, respectively. The training procedure is also summarized. Two use-cases are considered: global model and subject-specific model. The first one is the result of the training across all the subjects' data, while the second one uses transfer learning to, starting from the global model, specialize to a subject-specific one, achieving an average improvement of 9.19±0.86%. This endorses the development of subject-specific models for EEG-based MI BCI.

Then, the implementation process is described. The Tensorflow Lite framework has been used to address it, enabling the mapping of the high-level floating-point model into a description compatible with the low-end characteristics of the ARM Cortex M4F available on the Arduino Nano 33 Sense. Therefore, this tool has enabled the use of 8-bit integer post-training quantization, reducing the memory footprint at the expense of model accuracy, with an average downgrade of 2.64±0.77%. The $N_{chan} = 38$, $T = 1$, and $ds = 3$ model has been

fully implemented to be compared with the ARM Cortex M4F implementation described in [20]. Although our model is 4.75% lower in accuracy, it takes a similar number of clock cycles to get the inference. Additionally, the method to characterize the energy consumption per inference is described, resulting on 2.55 mJ, a 40% lower than the [20] implementation.

As future work, further optimization of the model and implementation must be considered to attain better performance while reducing the latency and energy consumption. In this sense, techniques as pruning and quantization-aware training will be evaluated. Additionally, in order to advance towards a BCI based in low-end hardware, the usage of the 12-bit ADC available on the Arduino Nano 33 Sense must be studied.

REFERENCES

[1] Xiaorong Gao, Dingfeng Xu, Ming Cheng, and Shangkai Gao, "A BCI-based environmental controller for the motion-disabled," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 11, no. 2, pp. 137–140, Jun. 2003, doi: 10.1109/TNSRE.2003.814449.

[2] E. P. Torres, E. A. Torres, M. Hernández-Álvarez, and S. G. Yoo, "EEG-Based BCI Emotion Recognition: A Survey," *Sensors*, vol. 20, no. 18, p. 5083, Sep. 2020, doi: 10.3390/s20185083.

[3] K. Aboalayon, M. Faezipour, W. Almuhammadi, and S. Moslehpour, "Sleep Stage Classification Using EEG Signal Analysis: A Comprehensive Survey and New Investigation," *Entropy*, vol. 18, no. 9, p. 272, Aug. 2016, doi: 10.3390/e18090272.

[4] G. Pires, M. Torres, N. Casaleiro, U. Nunes, and M. Castelo-Branco, "Playing Tetris with non-invasive BCI," in *2011 IEEE 1st International Conference on Serious Games and Applications for Health (SeGAH)*, Braga, Portugal: IEEE, Nov. 2011, pp. 1–6. doi: 10.1109/SeGAH.2011.6165454.

[5] A. A. Ghodake and S. D. Shelke, "Brain controlled home automation system," in *2016 10th International Conference on Intelligent Systems and Control (ISCO)*, Coimbatore, India: IEEE, Jan. 2016, pp. 1–4. doi: 10.1109/ISCO.2016.7727050.

[6] K. Szocik and T. Wójtowicz, "Human enhancement in space missions: From moral controversy to technological duty," *Technol. Soc.*, vol. 59, p. 101156, Nov. 2019, doi: 10.1016/j.techsoc.2019.101156.

[7] G. Pfurtscheller and C. Neuper, "Motor imagery and direct brain-computer communication," *Proc. IEEE*, vol. 89, no. 7, pp. 1123–1134, Jul. 2001, doi: 10.1109/5.939829.

[8] D. L. Schomer, F. H. Lopes da Silva, and E. Niedermeyer, Eds., *Niedermeyer's electroencephalography: basic principles, clinical applications, and related fields*, Seventh edition. New York, NY: Oxford University Press, 2018.

[9] S. Liu, G. Mingas, and C.-S. Bouganis, "An Unbiased MCMC FPGA-Based Accelerator in the Land of Custom Precision Arithmetic," *IEEE Trans. Comput.*, vol. 66, no. 5, pp. 745–758, May 2017, doi: 10.1109/TC.2016.2630682.

[10] A. Kachenoura, L. Albera, L. Senhadji, and P. Comon, "Ica: a potential tool for bci systems," *IEEE Signal Process. Mag.*, vol. 25, no. 1, pp. 57–68, 2008, doi: 10.1109/MSP.2008.4408442.

[11] W.-Y. Hsu and Y.-N. Sun, "EEG-based motor imagery analysis using weighted wavelet transform features," *J. Neurosci. Methods*, vol. 176, no. 2, pp. 310–318, Jan. 2009, doi: 10.1016/j.jneumeth.2008.09.014.

[12] A. Ishfaque, A. J. Awan, N. Rashid, and J. Iqbal, "Evaluation of ANN, LDA and Decision trees for EEG based Brain Computer Interface," in *2013 IEEE 9th International Conference on Emerging Technologies (ICET)*, Islamabad, Pakistan: IEEE, Dec. 2013, pp. 1–6. doi: 10.1109/ICET.2013.6743513.

[13] X. An, D. Kuang, X. Guo, Y. Zhao, and L. He, "A Deep Learning Method for Classification of EEG Data Based on Motor Imagery," in *Intelligent Computing in Bioinformatics*, D.-S. Huang, K. Han, and M. Gromiha, Eds., in Lecture Notes in Computer Science, vol. 8590. Cham: Springer International Publishing, 2014, pp. 203–210. doi: 10.1007/978-3-319-09330-7_25.

[14] S. Sakhavi, C. Guan, and S. Yan, "Parallel convolutional-linear neural network for motor imagery classification," in *2015 23rd European Signal Processing Conference (EUSIPCO)*, Nice: IEEE, Aug. 2015, pp. 2736–2740. doi: 10.1109/EUSIPCO.2015.7362882.

[15] Y. R. Tabar and U. Halici, "A novel deep learning approach for classification of EEG motor imagery signals," *J. Neural Eng.*, vol. 14, no. 1, p. 016003, Feb. 2017, doi: 10.1088/1741-2560/14/1/016003.

[16] R. T. Schirrmeister *et al.*, "Deep learning with convolutional neural networks for EEG decoding and visualization: Convolutional Neural Networks in EEG Analysis," *Hum. Brain Mapp.*, vol. 38, no. 11, pp. 5391–5420, Nov. 2017, doi: 10.1002/hbm.23730.

[17] H. Dose, J. S. Møller, H. K. Iversen, and S. Puthusserypady, "An end-to-end deep learning approach to MI-EEG signal classification for BCIs," *Expert Syst. Appl.*, vol. 114, pp. 532–542, Dec. 2018, doi: 10.1016/j.eswa.2018.08.031.

[18] C. Zhang, Y.-K. Kim, and A. Eskandarian, "EEG-inception: an accurate and robust end-to-end neural network for EEG-based motor imagery classification," *J. Neural Eng.*, vol. 18, no. 4, p. 046014, Aug. 2021, doi: 10.1088/1741-2552/abed81.

[19] V. J. Lawhern, A. J. Solon, N. R. Waytowich, S. M. Gordon, C. P. Hung, and B. J. Lance, "EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces," *J. Neural Eng.*, vol. 15, no. 5, p. 056013, Oct. 2018, doi: 10.1088/1741-2552/aace8c.

[20] X. Wang, M. Hersche, B. Tomekce, B. Kaya, M. Magno, and L. Benini, "An Accurate EEGNet-based Motor-Imagery Brain–Computer Interface for Low-Power Edge Computing," in *2020 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*, Bari, Italy: IEEE, Jun. 2020, pp. 1–6. doi: 10.1109/MeMeA49120.2020.9137134.

[21] T. Schneider, X. Wang, M. Hersche, L. Cavigelli, and L. Benini, "Q-EEGNet: an Energy-Efficient 8-bit Quantized Parallel EEGNet Implementation for Edge Motor-Imagery Brain-Machine Interfaces," in *2020 IEEE International Conference on Smart Computing (SMARTCOMP)*, Bologna, Italy: IEEE, Sep. 2020, pp. 284–289. doi: 10.1109/SMARTCOMP50058.2020.00065.

[22] A. C. Hernandez-Ruiz, D. Eneriz, N. Medrano, and B. Calvo, "Motor-Imagery EEGNet-Based Processing on a Low-Spec SoC Hardware," in *2021 IEEE Sensors*, Sydney, Australia: IEEE, Oct. 2021, pp. 1–4. doi: 10.1109/SENSORS47087.2021.9639747.

[23] A. Tsukahara, Y. Anzai, K. Tanaka, and Y. Uchikawa, "A design of EEGNet-based inference processor for pattern recognition of EEG using FPGA," *Electron. Commun. Jpn.*, vol. 104, no. 1, pp. 53–64, Mar. 2021, doi: 10.1002/ecj.12280.

[24] G. Schalk, D. J. McFarland, T. Hinterberger, N. Birbaumer, and J. R. Wolpaw, "BCI2000: A General-Purpose Brain-Computer Interface (BCI) System," *IEEE Trans. Biomed. Eng.*, vol. 51, no. 6, pp. 1034–1043, Jun. 2004, doi: 10.1109/TBME.2004.827072.

[25] F. Chollet *et al.*, "Keras." 2015. Accessed: May 02, 2023. [Online]. Available: https://keras.io

[26] ARM Ltd., "ARM Cortex-M4 datasheet." 2020. Accessed: Feb. 05, 2023. [Online]. Available: https://documentation-service.arm.com/static/62053f0a0ca305732a3a5c17?token=

[27] Arduino, "Arduino Nano 33 BLE Sense datasheet." 2023. Accessed: May 02, 2023. [Online]. Available: https://docs.arduino.cc/static/a2e56e30ed91586573e05927db7da6e4/ABX00031-datasheet.pdf

[28] A. Savitzky and M. J. Golay, "Smoothing and differentiation of data by simplified least squares procedures.," *Anal. Chem.*, vol. 36, no. 8, pp. 1627–1639, 1964.