electronics

MDPI

Article

Design and Evaluation of CPU-, GPU-, and FPGA-Based Deployment of a CNN for Motor Imagery Classification in Brain-Computer Interfaces

Federico Pacini [1,*], Tommaso Pacini [1], Giuseppe Lai [2], Alessandro Michele Zocco [1] and Luca Fanucci [1,*]

[1] Department of Information Engineering, University of Pisa, 56122 Pisa, Italy; tommaso.pacini@phd.unipi.it (T.P.); a.zocco@studenti.unipi.it (A.M.Z.)
[2] Department of Psychology, Goldsmiths, University of London, London WC1E 7HU, UK; glai002@gold.ac.uk
* Correspondence: federico.pacini@phd.unipi.it (F.P.); luca.fanucci@unipi.it (L.F.)

Abstract: Brain–computer interfaces (BCIs) have gained popularity in recent years. Among non-invasive BCIs, EEG-based systems stand out as the primary approach, utilizing the motor imagery (MI) paradigm to discern movement intentions. Initially, BCIs were predominantly focused on nonembedded systems. However, there is now a growing momentum towards shifting computation to the edge, offering advantages such as enhanced privacy, reduced transmission bandwidth, and real-time responsiveness. Despite this trend, achieving the desired target remains a work in progress. To illustrate the feasibility of this shift and quantify the potential benefits, this paper presents a comparison of deploying a CNN for MI classification across different computing platforms, namely, CPU-, embedded GPU-, and FPGA-based. For our case study, we utilized data from 29 participants included in a dataset acquired using an EEG cap for training the models. The FPGA solution emerged as the most efficient in terms of the power consumption–inference time product. Specifically, it delivers an impressive reduction of up to 89% in power consumption compared to the CPU and 71% compared to the GPU and up to a 98% reduction in memory footprint for model inference, albeit at the cost of a 39% increase in inference time compared to the GPU. Both the embedded GPU and FPGA outperform the CPU in terms of inference time.

Keywords: brain–computer interface; motor imagery; assistive technologies; convolutional neural network

Citation: Pacini, F.; Pacini, T.; Lai, G.; Zocco, A.M.; Fanucci, L. Design and Evaluation of CPU-, GPU-, and FPGA-Based Deployment of a CNN for Motor Imagery Classification in Brain–Computer Interfaces. *Electronics* **2024**, *13*, 1646. https://doi.org/10.3390/electronics13091646

Academic Editors: Pratheepan Yogarajah, Muthu Subash Kavitha, Lamiaa Abdel-Hamid and Ananthakrishnan Balasundaram

Received: 21 March 2024
Revised: 15 April 2024
Accepted: 22 April 2024
Published: 25 April 2024

1. Introduction

In recent years, Brain-Computer Interfaces (BCIs) have emerged as a groundbreaking technology within the broader Human-Machine Interface framework. They offer innovative communication and control options for individuals with severe neuromuscular disorders like amyotrophic lateral sclerosis, those with spinal cord injuries [1], or those recovering from strokes [2,3]. BCIs enable tasks such as controlling robotic limbs [4], exoskeletons [5], and wheelchairs [6]. Among the various modalities of BCIs, noninvasive approaches leveraging Electroencephalography (EEG) have gained significant attention due to their accessibility and clinical utility [7]. These painless systems decode users' intentions from EEG signals, captured through electrodes placed on the scalp.

The effectiveness of noninvasive BCIs relies heavily on employing various control paradigms, with Motor Imagery (MI) being particularly notable. MI involves mentally simulating specific motor actions by imagining the kinesthetic sensations, activating the motor cortex, and generating distinct EEG patterns [8,9]. Unlike visual imagery, MI evokes kinesthetic sensations of movement, resulting in the suppression of activity in the contralateral sensorimotor cortex across alpha (8–13 Hz) and beta (14–30 Hz) frequency bands [9–11]. This phenomenon plays a crucial role in decoding motor intentions within BCI systems.

Since the EEG is weak and has a poor signal-to-noise ratio, it is not easy to extract and classify features [12]. In EEG-based BCI research, feature extraction techniques play a crucial role in translating raw EEG signals into useful information. Common methods like common spatial patterns [13] assist in extracting spatial patterns related to different motor intentions, facilitating subsequent Machine Learning (ML) model training. However, the emergence of Deep Learning (DL) models, such as Convolutional Neural Networks (CNNs), presents a promising shift towards the automated extraction of spatial–spectral–temporal features from EEG data. This advancement has the potential to enhance BCIs' performance and usability significantly [14–17].

While initial BCI efforts primarily targeted nonembedded systems, typically relying on general-purpose computing platforms, there is an increasing need to adapt BCI technologies for deployment at the edge, near the sensors generating the data. Processing BCI algorithms at the edge offers several advantages over the traditional cloud or offline approach. Firstly, it significantly reduces latency by performing computations closer to the data source, thereby enhancing responsiveness and real-time performance crucial for seamless user interaction. This latency reduction is particularly vital in applications demanding rapid feedback, such as neuroprosthetics or assistive communication devices. Moreover, edge computing reduces the requirement for continuous high-bandwidth data transmission, easing network congestion and enabling BCI systems to function reliably in various environments, including remote or resource-constrained settings. Additionally, edge processing enhances data privacy, a crucial feature, especially in healthcare applications.

In this scenario, the acceleration at the edge of computationally intensive and power-hungry Deep Neural Networks (DNNs) for BCI applications is still an open challenge [18]. Hardware platforms for the execution of BCI algorithms at the edge encompass a diverse array of technologies tailored to enhance the efficiency and performance of neural signal processing. These platforms often integrate dedicated hardware components such as Central Processing Units (CPUs), Graphics Processing Units (GPUs), or Field-Programmable Gate Arrays (FPGAs) to expedite computationally intensive tasks like signal filtering, feature extraction, and classification directly at the edge. CPUs offer versatility and general-purpose computation suitable for basic BCI tasks but might lack efficiency for complex algorithms due to their sequential processing nature. GPUs excel in parallel processing, enhancing performance for tasks requiring extensive data parallelism, such as deep learning-based BCI models [19,20]. However, they can be power-hungry and less energy-efficient for lightweight edge devices. On the other hand, FPGAs provide hardware customization, offering low-latency, energy-efficient solutions tailored to specific BCI algorithms [21,22]. While their development requires specialized expertise and longer design cycles, FPGAs offer unparalleled efficiency for real-time BCI applications at the edge. Choosing among these accelerators depends on the specific requirements of the BCI system, including performance, power consumption, and resource constraints.

Overall, the shift towards embedded BCI processing represents a paradigmatic advancement in neurotechnology, promising to democratize access to BCI capabilities while unlocking new opportunities for human–computer interaction and augmentation. By harnessing the computational power and efficiency of edge devices, embedded BCIs pave the way for a future where seamless brain–machine communication becomes not only feasible but also ubiquitous across diverse domains and user populations.

Authors' Contributions

This article aims to promote the adoption of BCI applications at the edge to ensure shorter response times, improved privacy, and the possibility of executing the algorithm also in the absence of connectivity. The primary contributions of this study are outlined as follows:

- Preliminary validation of a newly collected dataset composed of 29 patients using a BioSemi Active Two gel-based EEG cap.

- Training and fine-tuning of a CNN for motor imagery on the collected data in offline mode.
- Deployment and comparison of the trained CNN over multiple hardware technologies for edge computing: CPU, embedded GPU, and FPGA.

The following sections of this document are organized as follows: Section 2 outlines the system explanation, data utilized, neural network architecture, and methods proposed for comparing the various deployment architectures. Section 3 showcases the experimental findings. Section 4 comprises the discussion, while Section 5 contains the conclusions.

2. Materials and Methods

2.1. System Architecture

Aligned with the MI paradigm, the proposed system aims at recognizing the mental representation of limb movement. Specifically, our goal is to classify whether the participant has imagined moving the left hand or the right hand. The system configuration for this application is illustrated in Figure 1. Starting with the collection of EEG data from a participant, the information undergoes preprocessing to eliminate artifacts and enhance the signal-to-noise ratio. Subsequently, the preprocessed data are fed into the feature extraction block, responsible for processing and extracting features that accurately represent the information to be classified. The extracted features then serve as input to the classification process, and the resulting label is displayed on the screen. However, this is not the sole outcome; in certain scenarios, the actuation may manifest as a concrete action, such as moving a robotic arm or controlling a wheelchair. Regardless of the action, the outcome is fed back to the patient.

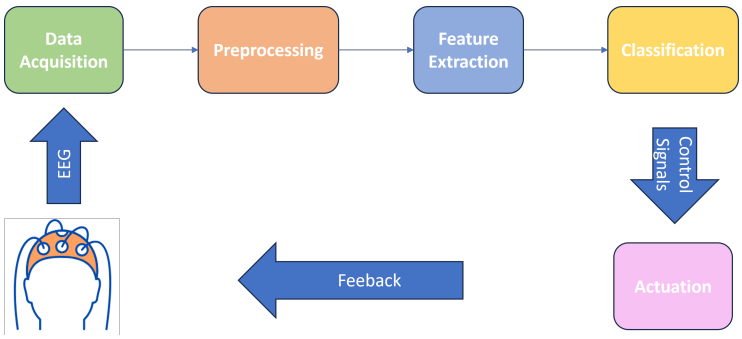

Figure 1. Brain–computer interface system overview.

2.2. Data

The overall dataset includes 29 right-handed healthy volunteers (females = 16; age = 18–40; mean age: 20.20; standard error of the mean, SEM = 0.66). Data collection was carried out at Goldsmiths, University of London, as part of a different study that received ethical approval.

The experimental paradigm was adapted from previous studies in the field of MI-based BCIs [23]. Participants were required to wear a gel-based EEG cap (BioSemi ActiveTwo, BioSemi Inc., Amsterdam, Netherlands BioSemi Active Two) [24] composed of 64 sensors and positioned on the scalp following the 10–20 electrode placement system. During the experiment, participants sat in a dimly lit room before a computer screen. Here, they were instructed to rehearse the kinesthetic sensations of left or right imaginary thumb abduction based on the presentation of an arrow on the screen (Figure 2). This MI task followed a real motor execution task of the same movements so that participants could familiarize themselves with kinesthetic sensations.

In the MI task, participants completed a total of 200 trials (100 for each thumb). The order of trials was randomly generated, with each trial lasting for 8 s (±250 ms), as depicted in Figure 2. During the initial two seconds, participants fixated on a cross presented on the screen. At the 2 s mark, the cross was replaced by an arrow indicating which thumb

to move, lasting for 4 s. During this window, participants were required to imagine the movements within the first second after the presentation of the cue. The cross replaced the arrow during the last two seconds.

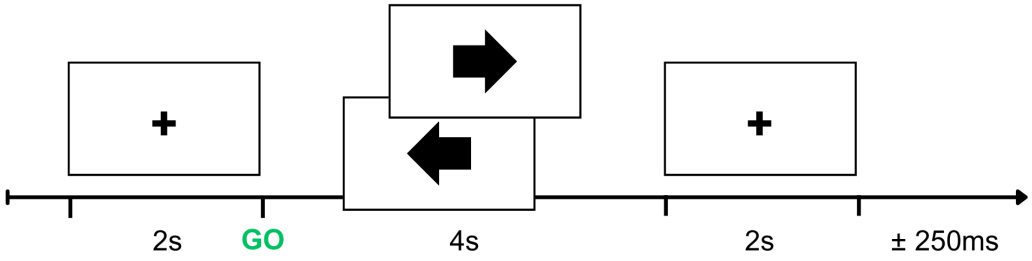

Figure 2. Illustration of the motor execution and imagery tasks.

The raw EEG data require a series of preprocessing steps to improve the signal-to-noise ratio. Following a standard preprocessing pipeline, we applied a band-pass filter in the frequency range of interest, 1–40 Hz (zero-phase, FIR design). The data were segmented around the timing of the experimental cues—the left/right arrows. This procedure, also known as epoching, improves the signal by (i) reducing nonstationarity at the local level [25] and (ii) reducing some of the noise present during the less relevant times of the experiment. Epoching occurred between −1 and 4 s around the cue.

The filtered EEG data include a series of artifacts that require further attention. Firstly, for very noisy EEG epochs where large muscle artifacts were present, we employed visual inspection. During this procedure, we manually rejected noisy EEG epochs and interpolated noisy channels with a spherical spline [26]. This process led to an average of 190 epochs (SEM = ±1.7) remaining for each participant, reflecting a 5 percent rejection rate. Finally, to correct for recurrent artifacts like eye blinks and eye saccades, independent component analysis (ICA, fastICA, [27]) was employed. Moreover, we removed the heartbeat artifact using a regression-based method, similar to a previous study [28].

Before training, the preprocessed EEG epochs were down-sampled at 256 Hz and cropped between 0.0 and 2.0 s postcue. This temporal window was selected based on previous research, indicating that alpha and beta power suppression in the contralateral sensorimotor cortex occurs within the first two seconds postcue [23,29,30]. Moreover, during our task, participants were instructed to initiate the movement immediately after the cues, imagining kinesthetic sensations within the first second.

Finally, to increase the number of training examples, the cleaned epochs were subjected to a commonly used data augmentation technique [31]. This involved employing a windowing approach with an overlapping factor of 75 percent. As a result, five examples were generated from each original, each maintaining a one-second duration (1: 0.0–1.0 s; 2: 0.25–1.25 s; 3: 0.5–1.5 s; 4: 0.75–1.75 s; 5: 1.00–2.00 s). The original label was assigned to all five new examples.

2.3. Convolutional Neural Network Architecture

For our MI classification task, we employed the EEGNet neural network [32]. Illustrated in Figure 3, EEGNet is a specialized 4-layer CNN designed specifically for EEG-based BCIs. This CNN is designed to process EEG input with C channels and T samples per channel. The C parameter is determined by the number of electrodes employed during data collection, while the parameter T is influenced by the time-window frame and the sampling frequency. To clarify, if the time window is x_s seconds and the sampling frequency is x_f Hz, the number of samples would be $x_s \times x_f$. Notably, this CNN stands out as it incorporates both temporal and spatial convolution layers.

The first layer of the CNN employs a temporal convolution, namely, a traditional convolution with a kernel height of one and a width proportional to T. Specifically, to capture frequency information at 2 Hz and beyond, the kernel width is configured as $T/2$. The objective of this layer is to extract features from individual channels over time. If we

denote F_1 as the number of temporal filters applied to the CNN input, this layer generates F_1 feature maps with dimensions identical to the input.

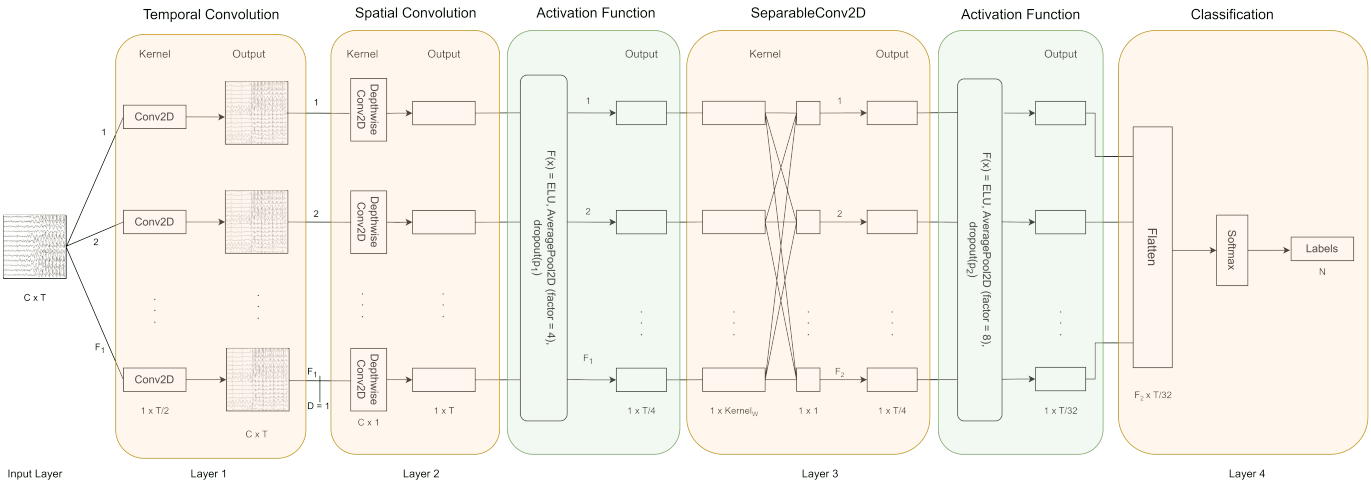

Figure 3. Architecture of EEGNet neural networks.

The second layer utilizes a spatial convolution, specifically a depthwise convolution with a kernel height of C and a width of 1. In this configuration, the layer directly learns spatial filters for each temporal filter, aiming to capture spatial correlations among different input channels. An added advantage of employing a depthwise convolution is the reduction in the number of trainable parameters because these convolutions are not fully connected to all preceding feature maps. An essential parameter for this layer is D, representing the depth parameter, which governs the number of spatial filters to be learned for each feature map. For visual clarity, Figure 3 represents $D = 1$.

The result from the second layer serves as input to an activation function that computes the exponential linear unit (ELU). Subsequently, it applies a pooling operation with a reduction factor of 4 and a dropout with a probability of p_1 to mitigate overfitting.

The third layer utilizes a separable convolution, which involves a sequence of depthwise and pointwise convolutions. The pointwise convolution is a specialized operation with a kernel size of 1×1. The objective of this layer is, through the first convolution, to learn distinct features over time (similar to the spatial convolution layer), and through the second one, to effectively combine them; this is achieved thanks to the 1×1 convolutions, which takes as input all the previously obtained filters, combined linearly. An essential parameter for this layer is F_2, representing the number of pointwise filters that determines the quantity of feature maps produced by this layer.

Once more, the result from the third layer is utilized as input to an activation function, which implements the exponential linear unit (ELU), followed by a pooling operation with a factor of 8 and dropout with a probability of p_2.

Given that the primary objective of this CNN is to categorize EEG signals into a predefined set of N classes, the final layer incorporates a softmax function. This function takes as input all the preceding feature maps and produces N distinct probabilities. Each probability is associated with the likelihood that the input signals belong to the i-th class.

EEGNet was implemented using Tensor Flow (TF) 2.0 [33], a well-established framework for machine learning development. The training of EEGNet took place on a computer equipped with an Intel Core i7-7500U CPU and 16 GB of RAM. Hyperparameter tuning was performed using the Optuna framework [34], and the final selection of hyperparameters is detailed in Appendix A in Table A1. A model was trained for each participant with the hyperparameter values specified in Table A1, incorporating early stopping. Individual training is necessary due to the collected data, which are highly dependent on the participant given the difference in the brain activities, potentials of the neurons, and their varying

abilities to visualize movement. Sporty individuals, accustomed to engaging their muscles, often exhibit higher proficiency in imagining movements.

Figure A2 in Appendix A illustrates the training process. The weights of the trained models were saved and subsequently utilized for testing.

2.4. Hardware Deployment and Benchmarking

The acceleration of BCI applications based on CNNs poses a significant challenge due to the computational complexity of these workloads. This challenge is further amplified when deploying BCI systems at the edge, such as on battery-based platforms, where additional constraints on power consumption and resource utilization must be considered. To address this open research question, we accelerated the trained EEGNet model using different hardware technologies. The goal of this study was to provide a comprehensive analysis of the trade-offs offered by each solution.

The selected hardware technologies include a CPU, GPU, and FPGA. To compare the different solutions together, we selected the following metrics that were collected on all three systems:

- Accuracy.
- Inference time.
- Power consumption.
- Memory footprint.

The following sections will delve into each device, providing more details to fully characterize the proposed experiment.

2.4.1. CPU

CPUs are one of the most popular components of computing devices, engineered to accommodate a wide array of instructions. While they offer advantages in flexibility and the ability to handle diverse computing tasks, their primary drawback lies in power consumption. This factor holds significant importance in embedded systems. Additionally, due to their focus on supporting a broad spectrum of instructions, CPUs are not fully optimized for ML computing, particularly for operations like matrix additions and multiplications. The execution of a CNN on a CPU comprises the classical programming flow, namely, compilation/interpretation and execution of binary instructions. For our tests, we utilized a desktop computer equipped with an Intel Core i7-7500U CPU.

2.4.2. GPU

Originally developed for graphics computing, GPUs have recently found application in machine learning due to their ability to optimize matrix operations and high levels of parallelization. Different GPU brands offer specific tools, such as NVIDIA's CUDA [35], to execute specialized and highly optimized operations. To broaden the scope and enhance the significance of the comparison across various architectures, we decided to utilize a GPU on an embedded device. Specifically, we chose the NVIDIA Jetson Nano 2GB Developer Kit [36] because it was designed for accelerating ML inference and is equipped with an NVIDIA Maxwell 128 CUDA Core GPU. Due to memory constraints imposed by the Jetson Nano, model optimization was necessary. We utilized the TF Lite framework 2.10, specifically designed for on-device inference. This framework takes a trained model and a representative dataset, producing an optimized version of the model through quantization, which involves reducing the model's arithmetic precision, along with hardware-specific optimization techniques. An interpreter utilizing GPU manufacturer APIs (CUDA in our case) then executes inference on the device. For an overview of the process, refer to Figure 4.

2.4.3. FPGA

FPGAs are reconfigurable hardware devices that can be programmed to perform CNN inference efficiently. Indeed, the programmable logic enables the design of accelerators customized for the specific model. These devices achieve high performance and power

efficiency since they allow one to push further the specificity of the design up to the model level. The main drawback of FPGA technology regards the long development time and the high design costs necessary to configure the programmable logic to implement an accelerator for a DNN model.

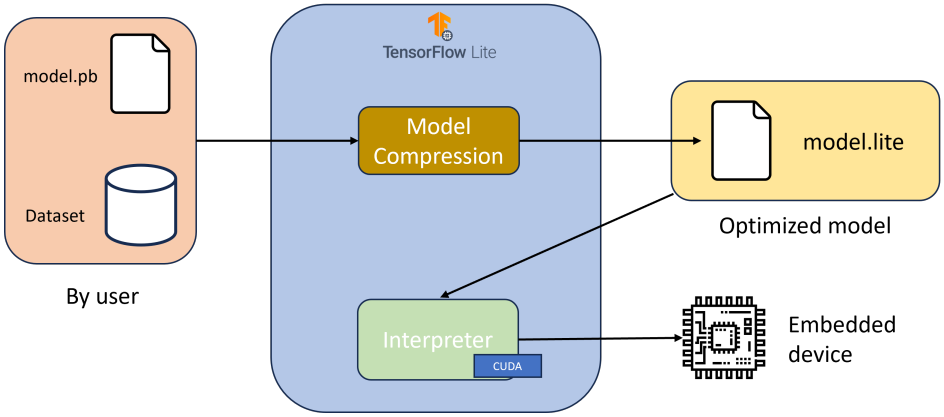

Figure 4. Overview of the model optimization for inference on an embedded device with NVIDIA GPU support.

To turn around this problem, the trained EEGNet model was implemented on this technology exploiting FPG-AI [37,38], a novel end-to-end tool flow for the automatic deployment of CNNs on FPGAs.

FPG-AI receives as input the pretrained network model, the application dataset, and the FPGA device chosen for acceleration. The user can also provide additional constraints in terms of accuracy, inference time, and resource occupancy. The tool flow first applies post-training model compression techniques to shrink the model size and shift from floating-point to fixed-point arithmetic. The featured accelerator is a fully handcrafted, third-party IP-free hardware description language (HDL)-based architecture referred to as the modular deep learning engine (MDE). By undertaking a comprehensive design space exploration (DSE) and leveraging a highly detailed analytical hardware model, FPG-AI autonomously generates a CNN-specific hardware accelerator, adept at efficiently harnessing FPGA resources while maintaining high-performance levels and adhering to the user's constraints. The list of supported layers of the tool flow is reported in [37]. For an overview of the process, refer to Figure 5.

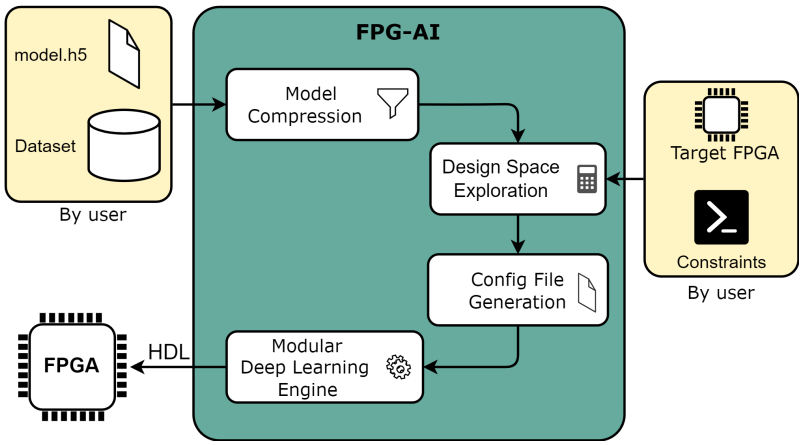

Figure 5. FPG-AI end-to-end tool flow.

To successfully process EEGNet with FPG-AI, modifications had to be performed on the tool flow to ensure support for a broader set of layers' configurations. More specifically, FPG-AI was updated to support convolutional kernels with generic shapes, improving the tool flow's usability in diverse application domains. The selected device for the characterization of EEGNET on FPGA technology is the Xilinx Ultrascale+ ZU7EV FPGA [39], a commonly used device for development and prototyping purposes. No specific constraints were provided on resource utilization and postquantization accuracy. With no directives, FPG-AI automatically generates as output the configuration with the highest computational parallelism achievable for the target CNN-FPGA pair that does not degrade the accuracy by more than 5%.

3. Results

The trained EEGNet models were deployed on the three different architectures following the strategies listed in Section 2.4. For each participant, the model's inference was executed on the testing dataset, and the metrics values were recorded. For the sake of comparison, we report the average values among the participants.

3.1. Inference on CPU

The trained models, having a 32-bit floating-point value representation, were deployed without alterations on the desktop computer. Consequently, the test accuracy remains unchanged. The average values of the participants are available in Table 1.

Table 1. Aggregate results of model deployment on the desktop computer.

	Average Inference Time (ms)	Average Power Consumption (W)	Average Memory Footprint (Mb)	Average Test Accuracy
Average over participants	121.57	13.22	476	81

3.2. Inference on GPU

To deploy the trained models on the NVIDIA Jetson Nano 2GB Developer Kit, the procedure described in Section 2.4.2 was followed. Due to limitations in resources, model optimization was necessary. Leveraging the TF Lite framework, an optimized model was generated. Specifically, a 16-bit floating-point arithmetic was chosen, reducing the model's footprint while maintaining negligible impact on accuracy. Consequently, test accuracy remains unaffected. The Jetson Nano offers two operational modes, 5 W and 10 W power consumption, with lower power consumption naturally resulting in longer inference times. For a comprehensive analysis, both modes were evaluated, and the average results of the participants can be found in Table 2.

Table 2. Results of model deployment on the NVIDIA Jetson Nano 2GB Developer Kit.

	5 W Mode Average Inference Time (ms)	10 W Mode Average Inference Time (ms)	Average Memory Footprint (Mb)	Average Test Accuracy
Average over participants	15.85	12.21	415	81

Considering the two operational modes, passing from 5 W to 10 W resulted in an average relative inference time enhancement of 23%.

3.3. Inference on FPGA

To deploy the trained models on the Xilinx Ultrascale+ ZU7EV FPGA, the process described in Section 2.4.3 was followed. Due to resource constraints, a trade-off between model resource usage and accuracy was made. In particular, the weights were represented on 4 bits and the input values on 8 bits. The average results of the participants are detailed in Table 3.

Table 3. Results of model deployment on the Xilinx Ultrascale+ ZU7EV FPGA.

	Average Inference Time (ms)	Average Power Consumption (W)	Average Memory Footprint (Mb)	Average Test Accuracy
Average over participants	12.97	1.47	6.37	77

The synthesized resources for allowing the model inference are reported in detail in Figure A3.

For further investigations, we calculated the average number of clock cycles spent on each convolutional layer for an inference run; the results are graphically represented in Figure A1.

4. Discussion

To facilitate the comparison among the different approaches, we reported the mean values for each approach together in Table 4. For easier comparison, we also calculated a figure of merit (FOM) for each approach, which is defined as the product value between the inference time and the power consumption. Given that both metrics are inversely proportional, a single value makes it easier to decide which strategy yields the best result. The smaller the value is, the better it is.

Table 4. Results aggregation among the different platform deployments.

	CPU	FPGA	Jetson Nano	
Power consumption (W)	13.22	1.47	5	10
Inference time (ms)	121.57	19.97	15.85	12.21
FOM (W × s)	1.61	29.36×10^{-3}	79.25×10^{-3}	122.1×10^{-3}
Memory footprint (Mb)	476	6.37	415	
Test accuracy	81%	77%	81%	

From an FOM ranking point of view, the FPGA appears to be the best solution, followed by the Jetson Nano (GPU-based) and finally the CPU. This aligns with our assumption: transitioning from a general-purpose platform (CPU) to a specialized one (FPGA) decreases the figure of merit (FOM). Considering power consumption, FPGA implementation, facilitated by FPG-AI frameworks, achieved a reduction of nearly 89% compared to CPU implementation, and 71% and 85% reduction for 5 W and 10 W Jetson Nano modes, respectively. A similar pattern emerges when considering the memory footprint required for model inference. Moving from CPU to GPU, we notice a decrease of approximately 12% in resource usage, mostly attributed to model optimization and the adoption of the TF Lite framework. However, the most significant reduction in resources occurs when transitioning to the FPGA, achieving a remarkable 98% decrease. However, in terms of inference time, Jetson Nano emerges as the optimal solution, with 39% less inference time

than the GPU. Upon further investigation, we observed that for the FPGA, approximately 96% of total clock cycles during inference are due to CNN first-layer computing, as depicted in Figure A1. This is due to the underlying FPG-AI framework's microarchitecture, which was conceived for handling large-dimensional filters in terms of height but not for those with significant width, as seen in the first layer. Given the relative novelty of the framework, further optimizations can be reasonably expected in terms of inference time, potentially widening the gap between GPU- and FPGA-based deployments. In terms of accuracy, both the CPU and Jetson Nano yield similar results. This parity is attributed to the use of the TF Lite framework for deployment on the Jetson Nano, where the required quantization was minimal and did not notably affect the outcome. Conversely, the optimization applied by FPG-AI marginally impacted accuracy, with the severity of the effect heavily dependent on the specific use case under analysis.

5. Conclusions

Brain–computer interfaces (BCIs) have gained popularity in recent years, promising to alleviate, and sometimes circumvent, neuromuscular disorders. Among noninvasive BCIs, electroencephalography (EEG) stands out as the primary approach, utilizing the motor imagery (MI) paradigm to discern movement intentions. Early BCIs primarily focused on nonembedded systems. However, there is now a growing demand for edge computing, offering advantages such as privacy, the ability to function even without remote connectivity, reduced bandwidth for data transmission, and real-time responsiveness. Consequently, we undertook a comparison of different deployment strategies—specifically, deployment on a CPU, embedded GPU, and FPGA. Utilizing a newly acquired EEG dataset focused on the MI paradigm, we trained EEGNet, a convolutional neural network (CNN), on an unconstrained device. Subsequently, we deployed this trained CNN on a CPU, GPU, and FPGA, collecting associated metrics. By defining a figure of merit (FOM) as the product of power consumption and inference time, we observed that the FPGA yielded the best results. Upon analyzing the factors contributing to the FOM, the FPGA exhibited a remarkable 89% reduction in power consumption compared to the CPU and a 71% reduction compared to the embedded GPU. Moreover, transitioning to the FPGA enabled a reduction in memory footprint of approximately 98%. However, the FPGA incurred an inference time 39% higher than the GPU, attributed to the specific implementation of the FPG-AI framework used for compressing and optimizing the CNN, which lacks optimization for convolutional filters with large widths. Nevertheless, the FPGA's inference time could be reduced by modifying the underlying FPG-AI framework microarchitecture, potentially widening the gap between GPU- and FPGA-based deployment.

Author Contributions: Conceptualization, F.P., T.P., G.L. and L.F.; methodology, F.P., G.L. and L.F.; software, F.P., T.P., A.M.Z. and G.L.; validation, F.P. and G.L.; formal analysis, F.P. and G.L.; investigation, A.M.Z.; resources, A.M.Z.; data curation, G.L.; writing—original draft preparation, F.P., T.P., G.L. and A.M.Z.; writing—review and editing, F.P., T.P. and L.F.; funding acquisition, L.F. All authors have read and agreed to the published version of the manuscript.

Funding: This study was partially supported by the Italian Ministry of Education and Research (MUR), Act 11 December 2016, n. 232, in the framework of the FoReLab project (Department of Excellence), CUP I53C23000340006.

Institutional Review Board Statement: The study received ethical approval on the 24th June 2021 with number PS240621GLS by the ethics committee of the psychology department at Goldsmiths, University of London.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: Relevant data is contained within the article. For additional data, please send an email to federico.pacini@phd.unipi.it

Acknowledgments: Commercial-free icons present in the images were downloaded from flaticon.com.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. List of hyperparameters used for EEGNet training.

Parameter	Value
T	512
C	64
D	2
$Kernel_w$	16
F_1	8
F_2	16
p_1	0.25
p_2	0.25
N	2

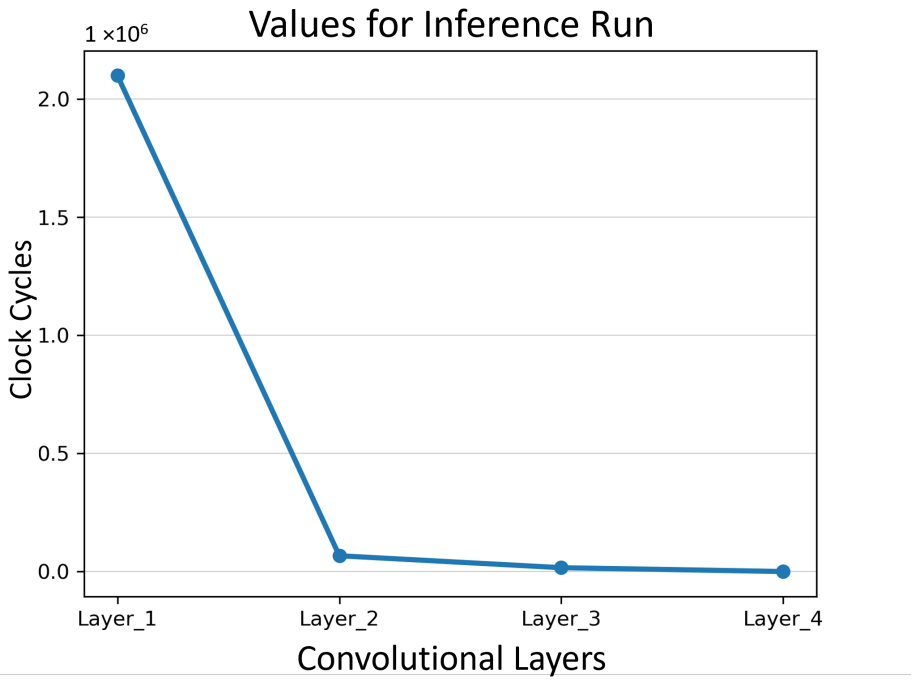

Figure A1. Clock cycles spent in each convolutional layer per inference run on FPGA technology.

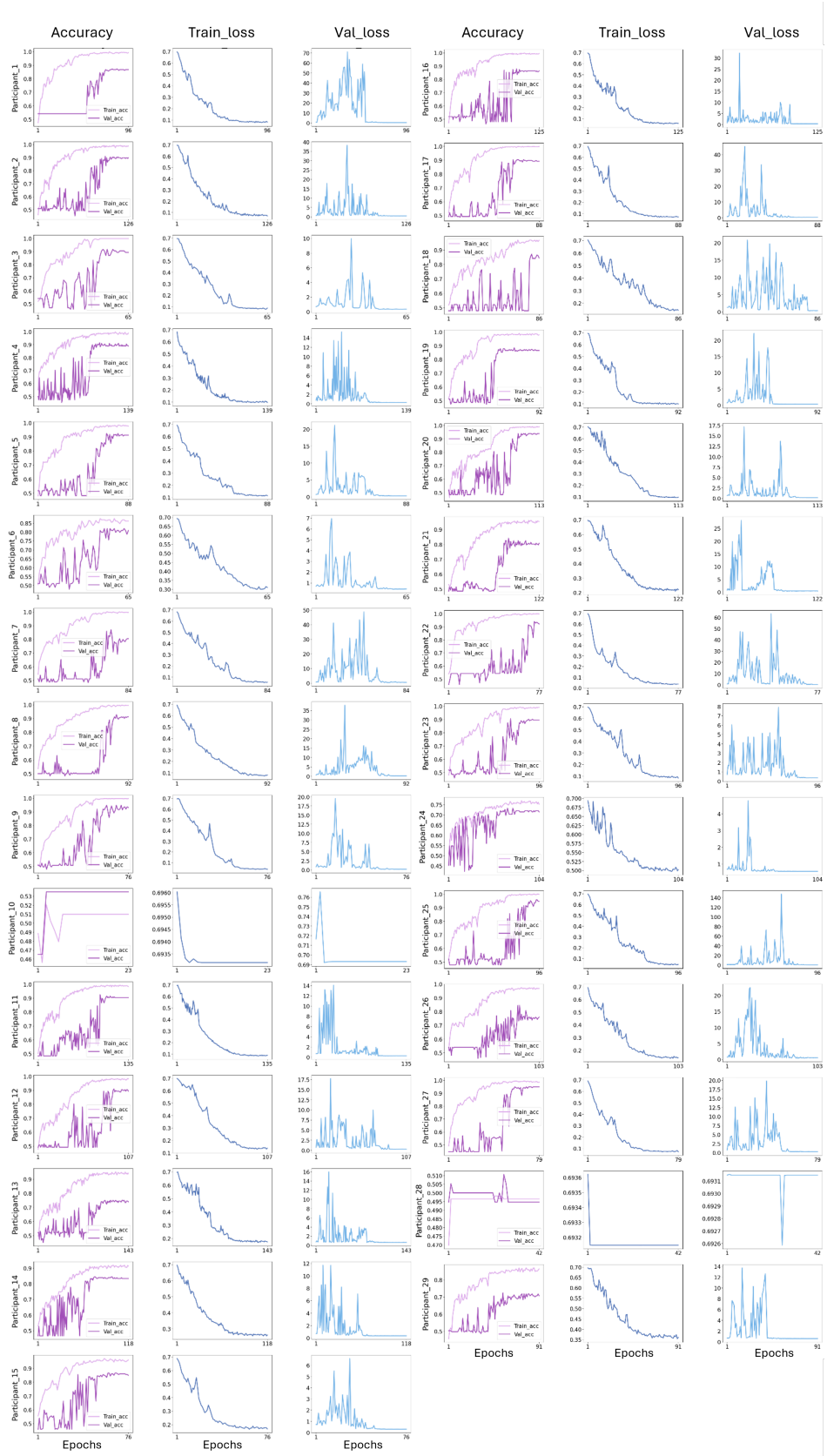

Figure A2. Training process results for participants.

Resource	Utilization	Available	Utilization %
LUT	28,892	230,040	12.54
LUTRAM	28	101,760	0.03
FF	12,964	460,800	2.81
BRAM	4	312	1.28
URAM	15	96	15.63
DSP	1 093	1 728	63.25
BUFG	1	544	0.18

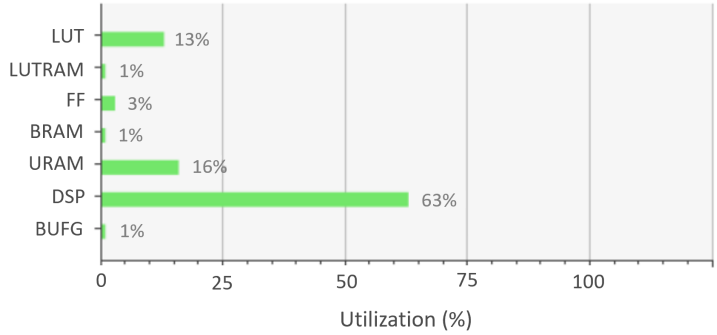

Figure A3. Resource usage on the Xilinx Ultrascale+ ZU7EV FPGA for the accelerator generated by FPG-AI in the highest parallelism configuration.

References

1. Zhang, R.; Wang, Q.; Li, K.; He, S.; Qin, S.; Feng, Z.; Chen, Y.; Song, P.; Yang, T.; Zhang, Y.; et al. A BCI-Based Environmental Control System for Patients with Severe Spinal Cord Injuries. *IEEE Trans. Biomed. Eng.* **2017**, *64*, 1959–1971. [CrossRef]
2. Biasiucci, A.; Leeb, R.; Iturrate, I.; Perdikis, S.; Al-Khodairy, A.; Corbet, T.; Schnider, A.; Schmidlin, T.; Zhang, H.; Bassolino, M.; et al. Brain-actuated functional electrical stimulation elicits lasting arm motor recovery after stroke. *Nat. Commun.* **2018**, *9*, 2421. [CrossRef] [PubMed]
3. Ang, K.K.; Guan, C. Brain-computer interface for neurorehabilitation of upper limb after stroke. *Proc. IEEE* **2015**, *103*, 944–953. [CrossRef]
4. Cho, J.H.; Jeong, J.H.; Shim, K.H.; Kim, D.J.; Lee, S.W. Classification of hand motions within EEG signals for non-invasive BCI-based robot hand control. In Proceedings of the 2018 IEEE international conference on systems, man, and cybernetics (SMC), Miyazaki, Japan, 7–10 October 2018; pp. 515–518.
5. Frolov, A.A.; Mokienko, O.; Lyukmanov, R.; Biryukova, E.; Kotov, S.; Turbina, L.; Bushkova, Y. Post-stroke rehabilitation training with a motor-imagery-based brain-computer interface (BCI)-controlled hand exoskeleton: A randomized controlled multicenter trial. *Front. Neurosci.* **2017**, *11*, 253346. [CrossRef]
6. Li, Y.; Pan, J.; Wang, F.; Yu, Z. A hybrid BCI system combining P300 and SSVEP and its application to wheelchair control. *IEEE Trans. Biomed. Eng.* **2013**, *60*, 3156–3166.
7. Tariq, M.; Trivailo, P.M.; Simic, M. EEG-based BCI control schemes for lower-limb assistive-robots. *Front. Hum. Neurosci.* **2018**, *12*, 312. [CrossRef]
8. Decety, J. The neurophysiological basis of motor imagery. *Behav. Brain Res.* **1996**, *77*, 45–52. [CrossRef] [PubMed]
9. Yang, Y.J.; Jeon, E.J.; Kim, J.S.; Chung, C.K. Characterization of kinesthetic motor imagery compared with visual motor imageries. *Sci. Rep.* **2021**, *11*, 3751. [CrossRef] [PubMed]
10. Lotze, M.; Halsband, U. Motor imagery. *J. Physiol.* **2006**, *99*, 386–395. [CrossRef]
11. Ridderinkhof, K.R.; Brass, M. How kinesthetic motor imagery works: A predictive-processing theory of visualization in sports and motor expertise. *J. Physiol.* **2015**, *109*, 53–63. [CrossRef]
12. Vaid, S.; Singh, P.; Kaur, C. EEG signal analysis for BCI interface: A review. In Proceedings of the 2015 Fifth International Conference on Advanced Computing & CommunicationTechnologies,Haryana, India, 21–22 February 2015; pp. 143–147.
13. Blankertz, B.; Tomioka, R.; Lemm, S.; Kawanabe, M.; Muller, K.r. Optimizing Spatial filters for Robust EEG Single-Trial Analysis. *IEEE Signal Process. Mag.* **2008**, *25*, 41–56. [CrossRef]
14. Al-Saegh, A.; Dawwd, S.A.; Abdul-Jabbar, J.M. Deep learning for motor imagery EEG-based classification: A review. *Biomed. Signal Process. Control* **2021**, *63*, 102172. [CrossRef]
15. Zhang, X.; Yao, L.; Wang, X.; Monaghan, J.; Mcalpine, D.; Zhang, Y. A survey on deep learning based brain computer interface: Recent advances and new frontiers. *arXiv* **2019**, arXiv:1905.04149.
16. Altaheri, H.; Muhammad, G.; Alsulaiman, M.; Amin, S.U.; Altuwaijri, G.A.; Abdul, W.; Bencherif, M.A.; Faisal, M. Deep learning techniques for classification of electroencephalogram (EEG) motor imagery (MI) signals: A review. *Neural Comput. Appl.* **2023**, *35*, 14681–14722. [CrossRef]

17. Saibene, A.; Caglioni, M.; Corchs, S.; Gasparini, F. EEG-based BCIs on motor imagery paradigm using wearable technologies: A systematic review. *Sensors* **2023**, *23*, 2798. [CrossRef] [PubMed]
18. Khademi, Z.; Ebrahimi, F.; Kordy, H.M. A review of critical challenges in MI-BCI: From conventional to deep learning methods. *J. Neurosci. Methods* **2023**, *383*, 109736. [CrossRef]
19. Wilson, J.A.; Williams, J.C. Massively parallel signal processing using the graphics processing unit for real-time brain-computer interface feature extraction. *Front. Neuroeng.* **2009**, *2*, 653. [CrossRef]
20. Raimondo, F.; Kamienkowski, J.E.; Sigman, M.; Slezak, D.F. CUDAICA: GPU optimization of infomax-ICA EEG analysis. *Comput. Intell. Neurosci.* **2012**, *2012*, 2. [CrossRef] [PubMed]
21. Shyu, K.K.; Lee, P.L.; Lee, M.H.; Lin, M.H.; Lai, R.J.; Chiu, Y.J. Development of a low-cost FPGA-based SSVEP BCI multimedia control system. *IEEE Trans. Biomed. Circuits Syst.* **2010**, *4*, 125–132. [CrossRef]
22. Heelan, C.; Nurmikko, A.V.; Truccolo, W. FPGA implementation of deep-learning recurrent neural networks with sub-millisecond real-time latency for BCI-decoding of large-scale neural sensors (104 nodes). In Proceedings of the 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Honolulu, HI, USA, 18–21 July 2018; pp. 1070–1073.
23. Sannelli, C.; Vidaurre, C.; Müller, K.R.; Blankertz, B. A large scale screening study with a SMR-based BCI: Categorization of BCI users and differences in their SMR activity. *PLoS ONE* **2019**, *14*, e0207351. [CrossRef]
24. Ins, B. BioSemi Active Two EEG Cap, 2001. Available online: https://www.biosemi.com/products.htm (accessed on 21 April 2024) .
25. Klonowski, W. Everything you wanted to ask about EEG but were afraid to get the right answer. *Nonlinear Biomed. Phys.* **2009**, *3*, 1–5. [CrossRef] [PubMed]
26. Perrin, F.; Pernier, J.; Bertrand, O.; Echallier, J.F. Spherical splines for scalp potential and current density mapping. *Electroencephalogr. Clin. Neurophysiol.* **1989**, *72*, 184–187. [CrossRef] [PubMed]
27. Hyvärinen, A.; Oja, E. Independent component analysis: algorithms and applications. *Neural Netw. Off. J. Int. Neural Netw. Soc.* **2000**, *13*, 411–430. [CrossRef]
28. Arnau, S.; Sharifian, F.; Wascher, E.; Larra, M.F. Removing the cardiac field artifact from the EEG using neural network regression. *Psychophysiology* **2023**, *60*, e14323. [CrossRef]
29. Yuan, H.; Liu, T.; Szarkowski, R.; Rios, C.; Ashe, J.; He, B. Negative covariation between task-related responses in alpha/beta-band activity and BOLD in human sensorimotor cortex: An EEG and fMRI study of motor imagery and movements. *NeuroImage* **2010**, *49*, 2596–2606. [CrossRef] [PubMed]
30. De Lange, F.; Jensen, O.; Bauer, M.; Toni, I. Interactions between posterior gamma and frontal alpha/beta oscillations during imagined actions. *Front. Hum. Neurosci.* **2008**, *2*,269. [CrossRef] [PubMed]
31. Lashgari, E.; Liang, D.; Maoz, U. Data augmentation for deep-learning-based electroencephalography. *J. Neurosci. Methods* **2020**, *346*, 108885. [CrossRef] [PubMed]
32. Lawhern, V.J.; Solon, A.J.; Waytowich, N.R.; Gordon, S.M.; Hung, C.P.; Lance, B.J. EEGNet: A compact convolutional neural network for EEG-based brain–computer interfaces. *J. Neural Eng.* **2018**, *15*, 056013. [CrossRef]
33. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. Available online: https://www.tensorflow.org/ (accessed on 21 April 2024).
34. Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; Koyama, M. Optuna: A next-generation hyperparameter optimization framework. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining,Anchorage, AK, USA, 4–8 August 2019; pp. 2623–2631.
35. NVIDIA; Vingelmann, P.; Fitzek, F.H. CUDA, release: 10.2.89, 2020. Available online: https://developer.nvidia.com/cuda-toolkit (accessed on 21 April 2024).
36. NVIDIA. Jetson Nano 2GB Developer Kit, 2020. Available online: https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-2gb-devkit (accessed on 21 April 2024) .
37. Pacini, T.; Rapuano, E.; Fanucci, L. FPG-AI: A Technology-Independent Framework for the Automation of CNN Deployment on FPGAs. *IEEE Access* **2023**, *11*, 32759–32775. [CrossRef]
38. Pacini, T.; Rapuano, E.; Tuttobene, L.; Nannipieri, P.; Fanucci, L.; Moranti, S. Towards the Extension of FPG-AI Toolflow to RNN Deployment on FPGAs for On-board Satellite Applications. In Proceedings of the 2023 European Data Handling & Data Processing Conference (EDHPC), Juan-Les-Pins, France, 2–6 October 2023; pp. 1–5. [CrossRef]
39. Xilinx, A. Xilinx Ultrascale+ ZU7EV Datasheet, 2022. Available online: https://docs.xilinx.com/v/u/en-US/ds891-zynq-ultrascale-plus-overview (accessed on 21 April 2024) .