# FPGA-based Deep-Learning Accelerators for Energy Efficient Motor Imagery EEG classification

Daniel Flood*, Neethu Robinson† and Shanker Shreejith*
*Department of Electronic & Electrical Engineering, Trinity College Dublin, Ireland
†School of Computer Science & Engineering, Nanyang Technological University, Singapore
email: *{floodd1, shankers}@tcd.ie, †nrobinson@ntu.edu.sg

***Abstract*—In recent years, Deep Learning has emerged as a powerful framework for analysing and decoding bio-signals like Electroencephalography (EEG) with applications in brain computer interfaces (BCI) and motor control. Deep convolutional neural networks have shown to be highly effective in decoding BCI signals for applications like two-class motor imagery decoding. Their deployment in real-time applications, however, requires highly parallel and capable computing platforms like GPUs to achieve high-speed inference, consuming a large amount of energy. In this paper, we explore a custom deep learning accelerator on an off-the-shelf hybrid FPGA device to achieve similar inference performance at a fraction of the energy consumption. We evaluate different optimisations at bit-level, datapath and training using a state-of-the-art deep convolutional neural network as our baseline model to arrive at our custom precision quantised deep learning model, which is implemented using the FINN compiler from Xilinx. The accelerator, deployed on a Xilinx Zynq Ultrascale+ FPGA, achieves a significant reduction in power consumption ($\approx 17\times$), sub 2 ms decoding latency and a near-identical decoding accuracy (statistically insignificant reduction of 2.5% average) as the reported baseline subject-specific classification accuracy on an N ($= 54$) subject motor imagery EEG (MI-EEG) dataset compared to the Deep CNN model on GPU, making our approach more appealing for low-power real-time BCI applications. Furthermore, this design approach is transferable to other deep learning models reported in BCI research, paving the way for novel applications of real-time portable BCI systems.**

***Index Terms*—Brain-Computer Interfaces, Deep Learning, Field Programmable Gate Arrays, AI Accelerators**

## I. Introduction

Recent years have seen a significant rise in the application of deep learning based inference and classification approaches across multiple domains with significant improvements in the fields of computer vision, and communication systems among others [1]. The ability of deep neural networks (DNNs) to learn and recognise intricate patterns has been a key enabler, allowing non-domain experts to analyse complex features from raw sensor data in many applications. Brain Computer Interface (BCI) is one of the many areas within the domain of neuro- and bio-engineering, where deep learning based decoding has enabled significant improvements in classification accuracy and generalisability.

BCI applications rely on the ability to decode neural activity and map the brain states to the activation of controls on the application's physical interface [2], [3]. Non-invasive acquisition technique like Electroencephalography (EEG) is widely used in BCI paradigms like motor imagery (MI) where characteristic Sensorimotor rhythm (SMR) activations are monitored to decode and classify right and left-hand motor imagery through data-driven approaches including traditional linear/non-linear classifiers, neural networks, transfer learning and deep learning [4], [5]. Recent research indicates that deep learning based techniques are gaining popularity in building EEG-BCI systems to improve decoding accuracy, with researchers validating the performance of different network models on a range of EEG datasets [6]–[13], eliminating the complex signal processing and feature engineering steps required in conventional BCI. Research in this area mostly focuses on enhancing classification accuracy with little attention given to the deployment challenges in real-world mobile applications. In all cases, the processing overheads of deep learning based methods are accelerated by deploying them on Graphical Processing Units (GPU) in PC based BCI. However, PC based BCI and deployments on GPU are expensive and power-hungry, often restricting EEG-BCI applications to laboratory environments. Offloading complex signal processing tasks of conventional EEG-BCI to platforms like field programmable gate array (FPGA) has been shown as an alternative approach to improve the cost and portability of EEG-BCI [14], [15]. FPGA-based implementations have also shown promising results with deep learning based EEG-BCI applications, enabling low-latency decoding in a compact footprint [16], [17]. However, most such implementations require significant design effort and optimisations to achieve similar performance to PC based BCI, making it non-trivial for non-experts. Furthermore, specific customisation at layers restricts the scalability and adaptability of such FPGA accelerated EEG-BCI decoding schemes. In many application domains, FPGA acceleration of deep learning models has enabled low-latency low-power inference by optimising the data width of inputs, weights, biases and subsequently operators, while providing similar accuracy to floating-point models.

In this paper, we present a case for FPGA acceleration of a deep CNN model for binary classification of EEG-BCI for motor imagery. The contributions of this paper are as follows:

- Custom quantisation of the state of the art deep-CNN Braindecode architecture [13] for minimising latency and energy consumption without impacting inference accu-

978-1-6654-8356-8/22/$31.00 ©2022 IEEE

racy.
- Optimisations in the design phase, training and hardware generation using the FINN compiler flow [18].
- Quantify the performance, latency and energy consumption of the quantised model on a hybrid FPGA device.

The choice of Braindecode architecture as the baseline model is driven by the extensive analysis performed by the authors and the high classification accuracy of the model, as well as the network layers supported by the FINN compilation flow. We perform subject-dependent hold-out training and evaluate the decoding accuracy of motor imagery across 54 subjects (from the state-of-the-art EEG dataset [19]). Our results show that a highly quantised 2-bit model deployed as a custom accelerator can achieve under 2 ms decoding latency while consuming nearly 17 $\times$ less power when deployed on a hybrid Zynq Ultrascale+ XCZU7EV device when compared to the baseline model on an Nvidia GTX 1080. The results also show that the 2-bit model achieves this performance with the minimal trade-off in decoding accuracy of 2.5% averaged across the 54 subjects, making it an efficient design paradigm for low-power mobile BCI-MI applications.

## II. Background and related work

### A. Brain Computer Interface

EEG is a widely used noninvasive data acquisition method for capturing brain activity for BCI research. A BCI system decodes neural activity through a variety of algorithms and tools to interpret the brain states, which are then mapped to activation commands for visualisation and/or for controlling external devices. Deep learning methods have enabled superior decoding performance of BCI signals when compared to conventional signal-processing based classification techniques, resulting in numerous tools and architectures being explored by researchers. The choice of brain signal representation, type of network model used and optimisations to hyperparameters have shown varying levels of success in reported methods in the literature, with no clear network/model providing a clear optimal solution [5], [13] . Many researchers have also experimented with data representation and transformation of the time series EEG data to minimise information loss and/or computational cost of the decoding engine. In [6], [7], the authors apply Hilbert transformation to raw EEG signal and use the envelope as input to a convolutional neural network (CNN) model to decode neural information. In [8], the authors use a recurrent-CNN model to decode cognitive load from multi-spectral image inputs that were generated from the EEG signal. Deep CNNs have also been utilised to decode attention and consciousness parameters from EEG data and to enable the generalisation of the model through inter-subject transfer learning [9], [10]. In [11], the authors convert the time series EEG signal into 2D images using short-term Fourier transform combining the time, frequency and location information of the signal, which is then fed to a 1D CNN and stacked auto-encoders to classify the signal. In [13], the authors propose the deep ConvNet architecture to decode raw EEG signals by organising them as a 2D array in the time-space domain without requiring specific feature engineering to decode the information. In [20], authors evaluated MI classification using deep ConvNet in subject-specific, subject-independent and subject-adaptive experiments and reported higher performance than conventional machine learning approaches. Targeting real-world deployment of such decoders, several studies have explored network compression techniques to reduce the implementation complexity of deep networks [21], [22]. Additionally, hardware acceleration techniques have been proposed for deep networks based on both application specific integrated circuit (ASIC) and field programmable gate array (FPGA) implementations. Our approach explores further low-level optimisations to the state-of-the-art deep CNN architecture aiming to minimise computational complexity and energy consumption of deep learning based BCI-MI decoding.

### B. FPGA acceleration of deep learning models

Deep learning architectures like CNNs have proven to be a powerful tool for solving challenging computer vision problems [23] as well as in non-vision embedded applications like wireless networks [24] and network security [25]. A key challenge is to embed these models close to the sensors to enable timely inference while consuming minimal power. An efficient way of deploying deep learning in such applications is to devise custom accelerators on platforms like FPGAs, which offer the flexibility to update the design post-deployment at much lower power consumption than GPU deployments. However, optimising bespoke deep learning designs for FPGA deployment requires considerable design effort to optimise the low-level architecture for efficient use of FPGA resources and to minimise off-chip storage. Early researchers attempted to address these challenges by reducing the bit-width of parameters through quantisation [26], deep compression [27] and combinations of these through manual intervention. Tools like FINN [18] and LUTNET [28] map high-level representations of deep neural networks to custom hardware on FPGAs, automating many areas of the mapping flow. FINN allows designers to exploit custom quantisation at each supported layer of the network (down to 1-bit fully binarised network) and employ a variety of transformations in the compilation flow to create highly efficient custom accelerators that can be easily deployed on hardware using frameworks like PYNQ or as a standalone custom design. FINN also enables quantisation aware training of the model using the Brevitas library to minimise the impact of reduced precision on the inference accuracy of the model [29]. Vendor tools like Vitis-AI from Xilinx can compile high-level deep learning models to executable designs that runs on off-the-shelf deep learning processing units (DPU) on Xilinx FPGAs [30]. While Vitis-AI and DPU flow provide a design flow similar to GPU deployments allowing non-experts to deploy them with minimal effort, this flow enforces a fixed 8-bit quantisation across all layers of the model and supports only post-quantisation optimisation of weights/biases, which could lead to a reduction

in the inference accuracy. In this paper, we use the FINN flow to exploit a fully custom design to maintain high decoding accuracy while optimising the energy and resource footprint of the deep CNN model on the FPGA.

## III. System Design and Architecture

### A. BCI Motor Imagery EEG Dataset

For our evaluation, we use the EEG dataset from the Department of Brain and Cognitive Engineering, Korea University which is openly available for BCI research. The dataset was captured using a BrainAmp recorder from fifty-four subjects, both male and female, in the age group of 24–35 [19]. The test recorded the BCI signals from the participants while performing two-class MI using 62 Ag/AgCl electrodes at a sampling rate of 1 kHz, using a well-established test protocol in BCI research [31]. For each subject, combining the data from all the recording sessions, the dataset consisted of 400 trials of motor imagery EEG with an equal number of right and left-hand trials. In our evaluation, we used a hold-out approach, in which we split the data into non-overlapping $\{50\text{-}25\text{-}25\}\%$ sets thereby obtaining training, validation and evaluation sets of 200, 100 and 100 trials respectively. Each trial consists of 4 seconds of EEG data and each such segment is further downsampled by a factor of 4 for network training and testing. For our evaluation, we use data from all the 62 channels for each participant to train and quantify the classification accuracy of our model for each subject as well as the average inference accuracy across all users.

### B. BCI - DeepCNN model

As mentioned, we use the state-of-the-art deep CNN model proposed by [13] as our baseline model for decoding EEG-MI. Fig. 1 shows the simplified architecture of this model, composed of 5 network layers. The first convolutional-pooling layer filters the input across the time and spatial domains, without any activation layer between them. The output of these layers are batch normalised before passing through the Exponential Linear Unit (ELU) activation function and a max-pooling layer with a stride of (3,1) and an identify layer. The subsequent convolutional-pooling layers follow a similar setup with a single convolutional layer performing temporal filtering and dropout layers to improve the learning. The final dense layer maps the outputs to one of the two classes (left-hand/right-hand) followed by the LogSoftmax activation function that outputs the predicted class based on the dense node output with highest probability.

### C. Optimising the model for FINN compiler

Prior to mapping the design to hardware, we evaluated the contribution of each layer of the network to the overall decoding accuracy and also to determine the compatibility of the operators with the FINN compiler. The 2-D asymmetric convolution filters used in the spatial convolution in the ConvPool-1 layer of the design were replaced with 1-D temporal filters since asymmetric 2-D kernels are unsupported by FINN; however, we also observed that this modification

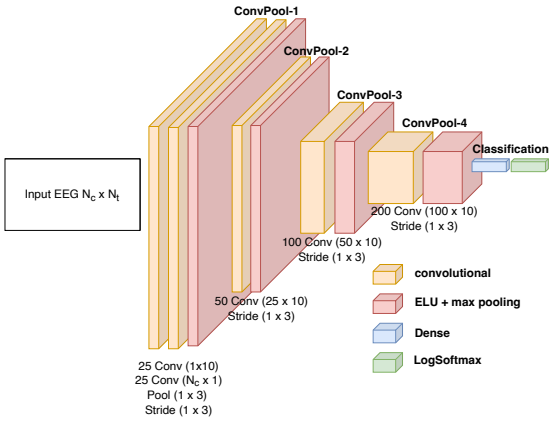


Fig. 1: State of the art deep CNN model for decoding EEG signals used as baseline model [13]

showed no appreciable change in the decoding accuracy of subject-specific test cases. ELU activation functions were replaced by the non-linear Rectified Linear Unit (ReLU) function at all layers. The kernels shapes of convolutional and max-pooling layers were adapted to enable optimal compilation of the network through FINN from (10,1) to (9,1) and (3,1) to (4,1) respectively. This ensured that the shape of the input data to each max-pooling layer was divisible by the kernel shape, as required by the FINN compiler. The strides of the max-pooling layers were adapted to match the kernel shapes at these layers to prevent the kernels from overlapping. The optimised model is further quantised and trained using the Brevitas library.

### D. Training the optimised quantised model

We trained the model using the "subject-specific" training strategy developed by [32]. In this mode, hold-out training is performed using the Brevitas framework, with the subject's data split into non-overlapping segments for training, validation and testing $\{50\text{-}25\text{-}25\}$. Brevitas integrates a set of abstractions to model non-standard precision within the datapath of the network at training time to perform quantisation aware training. The model was trained for 200 epochs using a batch size of 16 with *AdamW* optimiser. Each trained model was subsequently evaluated on the test fragment of the dataset, and is repeated across all 54 subjects to determine the mean accuracy. The input data was quantised to 16-bit precision.

The above steps were repeated for different quantisation levels of the model from 8 to 2-bit. The inference cost and the average accuracy of the model across 54 subjects were captured in each case to determine the configuration for hardware deployment. In this research, we restrict our evaluation to uniform quantisation across the layers to limit the design space. The final trained quantised models were exported in ONNX representation for implementation using the FINN compiler.

### E. Hardware generation and dataflow optimisations

To generate synthesisable model from the ONNX representations, the FINN compiler applies a series of transformations

on the ONNX graph to generate the dataflow architecture of the model. In our design, the layers of the graph were streamlined [33] and dataflow partitioned to isolate non-synthesisable layers (like input quantiser) from the synthesisable graph. The level of parallelism and performance were configured by choosing the degree of hardware resource folding, target performance (set to 1500 fps), clock frequency (set to 100 MHz) and maximum matrix-vector activation unit width (set to 20). FINN compiler then invokes high-level synthesis tools to generate the individual building blocks of the model and combines them to create the stitched IP block of the model for use in the deployment flow.

### *F. Deploying on the FPGA device*

With the stitched IP generated, standard vendor flow can be used to synthesise the hardware for deployment on an FPGA platform. For our experiment, we utilise the Python productivity for Zynq (PYNQ) framework to deploy the model on a Zynq Ultrascale+ XCZU7EV hybrid FPGA (ZCU104 development board). The hybrid Zynq family of devices integrate capable arm processors with a range of hardened peripheral logic and interface protocols on the processing system (PS) section of the device, allowing seamless software-based interfaces to external peripherals over standard interfaces like CAN, SPI and others. Custom logic blocks are implemented in the programmable logic (PL) region, which can then be accessed as custom peripherals/accelerators from the software application on the processor through a range of high-performance or low-bandwidth interface ports. High-level APIs enable software applications to offload tasks to such accelerators through non-blocking function calls from the software.

In case of PYNQ deployments, custom hardware circuits to be integrated on the PL region as overlays, mimicking the design flow using software libraries, with standardised APIs to interface with the overlay's capabilities. The PYNQ transformation invoked by the FINN compiler wires up the interfaces exposed by the design into the PYNQ shell allowing the model to be executed using the PYNQ runtime APIs. Subsequently, standard vendor flow is invoked to synthesise and implement the design and to generate the bitfile for the target device as well as the PYNQ runtime files to execute the model on the target device, both of which can then be copied to the device to execute the model. The PYNQ runtime are executed on top of a Linux kernel with a Python-based library framework which is used to interact with our quantised model on the PL for our experiments. The runtime also offers additional capabilities like active monitoring on the power rails on the board to capture highly accurate real-time power consumption of the models.

## IV. Experimental Results

In this section, we quantify the decoding accuracy of the quantised model(s) at different bit widths and compare them against our baseline deep CNN model. A workstation-class machine with an 8-core Intel i7-6700K CPU and Nvidia GTX 1080 GPU is used to train all models. Training the quantised model using the Brevitas library incurs almost double the time taken by the baseline deep CNN model. The baseline model (referred to as Deep CNN in results) was deployed on the Nvidia GTX 1080 GPU with the power measurements captured using the Nvidia Management Library (NVML). The quantised model (referred to as QdCNN-n bit, n specifying the quantised bit-width) was deployed on the Xilinx Zynq Ultrascale+ XCZU7EV FPGA on the ZCU104 development board. The power consumption of the QdCNN model while performing BCI decoding was measured using the PYNQ-PMBus package to monitor the power rails directly. The power measurements were averaged over 1000 decoding runs across all 54 subjects. We also quantify the latency of both approaches to detect the time taken to process each input sample and report the average latency over 1000 decoding cycles.

| QdCNN models | 2-bit | 4-bit | 6-bit | 8-bit |
|---|---|---|---|---|
| Normalised Inference Cost | 0.02 | 0.42 | 0.73 | 1.0 |

TABLE I: Inference cost of the quantised models (QdCNN) inferred through the FINN library and normalised to the 8-bit model.

### *A. Decoding accuracy vs Resource costs*

Fig. 2 captures the average decoding accuracy across all 54 subjects achieved by the QdCNN model using the Zynq Ultrascale+ FPGA as the target platform and compares them against the state-of-the-art Deep CNN model on the GTX 1080 GPU. We explore 8, 6, 4 and 2-bit uniform quantisation across all layers, with the BCI input quantised at 16-bit for the QdCNN model, while the baseline model was decoding at the native 32-bit precision. The plot shows that higher precision at layers does not provide any appreciable change in the average decoding accuracy compared to the lowest precision 2-bit model. To test the statistical significance of our proposed approach, we performed a two-sided Wilcoxon rank sum test. The test concluded that the 2.5% drop in classification accuracy observed in the case of the quantised model is statistically insignificant ($p = 0.6425$).

We further quantify inference costs incurred by each of the quantised models using a FINN utility function (inference cost) to estimate the memory footprint and operational complexity of each model. Inference cost captures the memory footprint and binary operations cost, which are normalised (against a baseline) and linearly combined to arrive at the normalised inference cost for each quantisation level. Table I shows the inference cost of different quantisation modes normalised to the 8-bit case. It can be seen that the inference cost of the 2-bit model is only 2% of the 8-bit model and is significantly below the 4-bit model. Combining the two results, we can infer that the QdCNN-2bit model offers the best trade-off between decoding accuracy and resource consumption.

Fig. 3 shows the decoding accuracy achieved by the QdCNN-2bit model for each subject and compares them

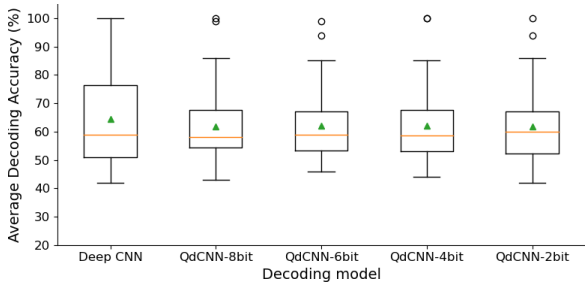


Fig. 2: Average decoding accuracy across all subjects of the baseline Deep CNN model against the proposed quantised model at different bit-precision's.

| Logic | #LUTs | #FF | #BRAMs | #DSP48 |
|---|---|---|---|---|
| QdCNN-2bit | 88201 | 110068 | 108 | 0 |
| wrapper | 9004 | 19154 | 0 | 0 |
| Total | 97205 | 129222 | 108 | 0 |
| (%) | 42.2 | 28.0 | 34.6 | 0 |

TABLE II: Hardware resource consumption of the 2-bit precision QdCNN model on the XCZU7EV FPGA.

against the baseline Deep CNN model. It can be observed that the quantised model offers similar decoding accuracy in many cases and outperforms the baseline model in some cases. The QdCNN-2bit achieves an average decoding accuracy of 61.8% across all 54 subjects and a median decoding accuracy of 60.0%, compared to the 64.2% mean and 59.0% median accuracy of the baseline model, making their generalised decoding performance almost identical. In the next section, we quantify the resource overheads, power consumption and latency of the QdCNN-2bit model when deployed on the Zynq FPGA.

### *B. Hardware utilisation, Power consumption and Latency*

To compute the resource consumption, the QdCNN-2bit model is synthesised targeting the XCZU7EV FPGA device and integrated using the PYNQ framework, as described in section III-F. Table II shows the overall resources consumed by the QdCNN-2bit model on the FPGA and the resources incurred by the supporting logic for interfaces to the ARM cores on the PS. The ARM cores are executing Linux OS and a Python support framework (through the PYNQ image) that offers APIs to communicate with the hardware model deployed on the programmable logic region of the device. The resource consumption shows that the compact model consumes nearly 42% of the general purpose resources (LookUp Tables or LUTs) on the device, leaving sufficient resources for any custom logic/processing blocks to interface directly with BCI-MI sensors/peripherals.

We further quantify the active power consumption of our model while performing BCI-MI decoding by measuring the real-time power consumption from the power rails on the board. For this experiment, the samples from our subject dataset is fed as inputs to the model sequentially, mimicking a streaming processing setup with the EEG-BCI sensors directly

| Device | Model | Power (W) | Latency (ms) |
|---|---|---|---|
| XCZU7EV FPGA | 2-bit QdCNN | 2.52±0.03 | 1.93±0.06 |
| Nvidia GTX 1080 | Deep CNN | 42.69±1.33 | 1.11±0.09 |

TABLE III: The performance of the FPGA and GPU during model inference.

connected to the board. We use the same strategy to monitor the power consumption of the baseline Deep CNN model on the GTX 1080 GPU, where the real-time power consumption of the GPU alone was measured using the Python wrappers for NVML. This setup also allows us to measure the latency incurred for decoding each time sample for both designs. All measurements are averaged across 1000 samples and across all subjects. The QdCNN model was clocked at 100 MHz while the GTX 1080 was clocked at its base clock (1607 MHz). The results are tabulated in table III. It can be seen that the QdCNN-2bit on FPGA achieves 16.9 $\times$ reduction in power consumption over baseline model on the GTX 1080 GPU. Also, the QdCNN achieves sub 2 ms decoding latency when averaged across multiple runs which is slightly below the baseline model due to higher parallelism enabled by the GPU. Note that typical end-to-end delays in real-time BCI-MI systems are orders of the magnitude higher. The significant reduction in power consumption and the near identical performance makes our approach substantially better than the state-of-the-art for low-power real-time mobile BCI applications. Also, the design approach can be easily applied to other deep learning models described in BCI literature with potentially similar advantages in decoding accuracy, latency and power consumption.

## V. Conclusion

Deep learning has emerged as the design tool of choice for building generalised neuroengineering systems for wide range of applications. While researchers in the area have explored ways to improve decoding accuracy, deploying deep learning accelerators for low-power, real-time and mobile BCI applications is an open problem. In this paper, we evaluate the deployment of deep learning for BCI-MI using hybrid FPGAs, exploiting optimisations to the deep learning model for generating a light-weight efficient design for the FPGA. We explore deep quantisation through the FINN compiler tools and Brevitas training library to generate quantised variants of the state-of-the-art deep CNN model with minimal impact on decoding accuracy. Our results show that a 2-bit quantised model on the FPGA achieves $\approx 17\times$ reduction in power consumption with low-decoding latency ($< 2$ ms) while achieving nearly identical decoding accuracy as the state-of-the-art deep CNN model on a GPU, making our approach ideal for low-power real-time BCI decoding using deep learning techniques.

## References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T. M. Vaughan, "Brain–computer interfaces for communication and control," *Clinical neurophysiology*, vol. 113, no. 6, pp. 767–791, 2002.

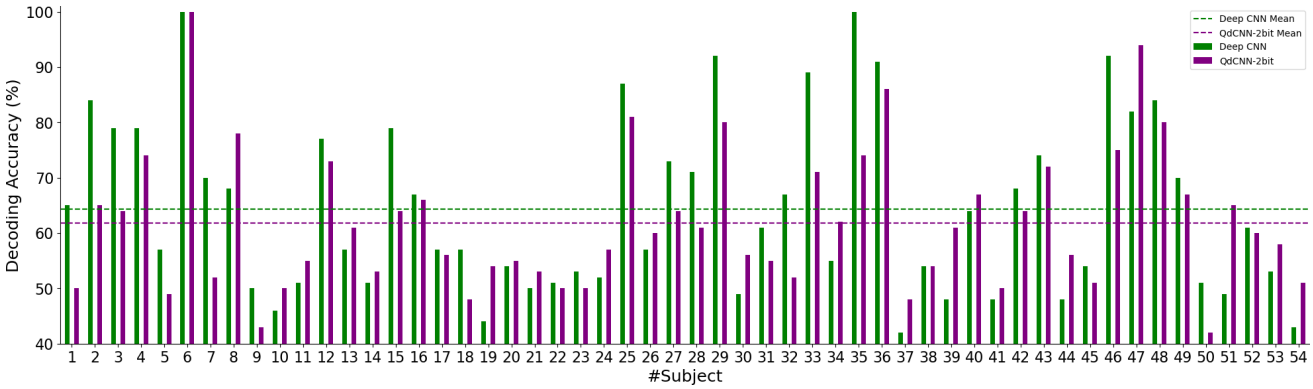


Fig. 3: Decoding accuracy of each subject using the proposed 2-bit quantised model (QdCNN-2bit) compared to the state-of-the-art Deep CNN model (Deep CNN) [13].


[3] D. J. McFarland and J. R. Wolpaw, "Brain-computer interfaces for communication and control," *Communications of the ACM*, vol. 54, no. 5, pp. 60–66, 2011.

[4] B. He, B. Baxter, B. J. Edelman, C. C. Cline, and W. Y. Wenjing, "Non-invasive brain-computer interfaces based on sensorimotor rhythms," *Proceedings of the IEEE*, vol. 103, no. 6, pp. 907–925, 2015.

[5] F. Lotte, L. Bougrain, A. Cichocki, M. Clerc, M. Congedo, A. Rakotomamonjy, and F. Yger, "A review of classification algorithms for EEG-based brain–computer interfaces: a 10 year update," *Journal of neural engineering*, vol. 15, no. 3, p. 031005, 2018.

[6] S. Sakhavi, C. Guan, and S. Yan, "Learning temporal information for brain-computer interface using convolutional neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 11, pp. 5619–5629, 2018.

[7] S. Sakhavi and C. Guan, "Convolutional neural network-based transfer learning and knowledge distillation using multi-subject data in motor imagery BCI," in *Proc. International Conference on Neural Engineering (NER)*, pp. 588–591, 2017.

[8] P. Bashivan, I. Rish, M. Yeasin, and N. Codella, "Learning representations from EEG with deep recurrent-convolutional neural networks," *arXiv preprint arXiv:1511.06448*, 2015.

[9] F. Fahimi, Z. Zhang, W. B. Goh, T.-S. Lee, K. K. Ang, and C. Guan, "Inter-subject transfer learning with an end-to-end deep convolutional neural network for EEG-based BCI," *Journal of neural engineering*, vol. 16, no. 2, p. 026007, 2019.

[10] M. Lee, S.-K. Yeom, B. Baird, O. Gosseries, J. O. Nieminen, G. Tononi, and S.-W. Lee, "Spatio-temporal analysis of EEG signal during consciousness using convolutional neural network," in *Proc. International Conference on Brain-Computer Interface (BCI)*, pp. 1–3, 2018.

[11] Y. R. Tabar and U. Halici, "A novel deep learning approach for classification of EEG motor imagery signals," *Journal of neural engineering*, vol. 14, no. 1, p. 016003, 2016.

[12] A. Echtioui, W. Zouch, M. Ghorbel, C. Mhiri, and H. Hamam, "A Novel Ensemble Learning Approach for Classification of EEG Motor Imagery Signals," in *Proc. International Wireless Communications and Mobile Computing (IWCMC)*, pp. 1648–1653, 2021.

[13] R. T. Schirrmeister, J. T. Springenberg, L. D. J. Fiederer, M. Glasstetter, K. Eggensperger, M. Tangermann, F. Hutter, W. Burgard, and T. Ball, "Deep learning with convolutional neural networks for EEG decoding and visualization," *Human brain mapping*, vol. 38, no. 11, pp. 5391–5420, 2017.

[14] K.-K. Shyu, P.-L. Lee, M.-H. Lee, M.-H. Lin, R.-J. Lai, and Y.-J. Chiu, "Development of a low-cost FPGA-based SSVEP BCI multimedia control system," *IEEE Transactions on biomedical circuits and systems*, vol. 4, no. 2, pp. 125–132, 2010.

[15] K. Belwafi, F. Ghaffari, R. Djemal, and O. Romain, "A hardware/software prototype of EEG-based BCI system for home device control," *Journal of Signal Processing Systems*, vol. 89, no. 2, pp. 263–279, 2017.

[16] C. Heelan, A. V. Nurmikko, and W. Truccolo, "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 *Proc. International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 1070–1073, 2018.

[17] R. R. Shrivastwa, V. Pudi, and A. Chattopadhyay, "An FPGA-based brain computer interfacing using compressive sensing and machine learning," in *IEEE Symposium on VLSI (ISVLSI)*, pp. 726–731, 2018.

[18] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, "FINN: A framework for fast, scalable binarized neural network inference," in *Proc. International Symposium on Field-Programmable Gate Arrays (FPGA)*, pp. 65–74, 2017.

[19] M.-H. Lee, O.-Y. Kwon, Y.-J. Kim, H.-K. Kim, Y.-E. Lee, J. Williamson, S. Fazli, and S.-W. Lee, "EEG dataset and OpenBMI toolbox for three BCI paradigms: an investigation into BCI illiteracy," *GigaScience*, vol. 8, no. 5, p. giz002, 2019.

[20] K. Zhang, N. Robinson, S.-W. Lee, and C. Guan, "Adaptive transfer learning for EEG motor imagery classification with deep Convolutional Neural Network," *Neural Networks*, vol. 136, pp. 1–10, 2021.

[21] R. Vishnupriya, N. Robinson, R. Reddy, and C. Guan, "Performance Evaluation of Compressed Deep CNN for Motor Imagery Classification using EEG," in *Proc. International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pp. 795–799, 2021.

[22] H. Cecotti and A. Gräser, "Neural network pruning for feature selection-Application to a P300 Brain-Computer Interface," in *ESANN*, Citeseer, 2009.

[23] K. Lee, J. Zung, P. Li, V. Jain, and H. S. Seung, "Superhuman accuracy on the SNEMI3D connectomics challenge," *arXiv preprint arXiv:1706.00120*, 2017.

[24] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Communications surveys & tutorials*, vol. 21, no. 3, pp. 2224–2287, 2019.

[25] H. M. Song, J. Woo, and H. K. Kim, "In-vehicle network intrusion detection using deep convolutional neural network," *Vehicular Communications*, vol. 21, 2020.

[26] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.

[27] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[28] E. Wang, J. J. Davis, P. Y. Cheung, and G. A. Constantinides, "LUTNet: Rethinking inference in FPGA soft logic," in *Proc. International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 26–34, 2019.

[29] A. Pappalardo, "Xilinx/brevitas," 2021.

[30] Xilinx, "Zynq DPU v3.2," 2020.

[31] G. Pfurtscheller and C. Neuper, "Motor imagery and direct brain-computer communication," *Proceedings of the IEEE*, vol. 89, no. 7, pp. 1123–1134, 2001.

[32] K. Zhang, N. Robinson, S.-W. Lee, and C. Guan, "Adaptive transfer learning for eeg motor imagery classification with deep convolutional neural network," *Neural Networks*, vol. 136, pp. 1–10, 2021.

[33] Y. Umuroglu and M. Jahre, "Streamlined deployment for quantized neural networks," *arXiv preprint arXiv:1709.04060*, 2017.