A Sparsity-driven tinyML Accelerator for Decoding Hand Kinematics in Brain-Computer Interfaces

Adithya Krishna[1,2*], Vignesh Ramanathan[1*], Satyapreet Singh Yadav[1*], Sahil Shah[3], André van Schaik[2], Mahesh Mehendale[1] and Chetan Singh Thakur[1]

[1]Department of Electronic Systems Engineering, Indian Institute of Science, Bangalore, India
[2]International Centre for Neuromorphic Systems, The MARCS Institute, Western Sydney University, Australia
[3]Department of Electrical and Computer Engineering, University of Maryland, USA

***Abstract*—Despite advances in understanding the mechanisms of movement disorders, controlling voluntary movements remains challenging, with limited treatment options. However, the integration of machine learning (ML) accelerators into the brain-computer interface (BCI) pipeline offers promising solutions by harnessing the capabilities of ML algorithms to decode intended motor actions accurately. While there have been numerous efforts to classify intended hand movements into discrete classes, the challenging problem of decoding continuous hand kinematics has seen limited research efforts. Our work focuses on tackling this challenge using RAMAN, an energy-efficient tinyML accelerator. We demonstrate the successful decoding of macaque hand kinematics from the MC_Maze dataset on the Efinix Ti60 FPGA with RAMAN architecture. Our approach achieves an R2-score of 0.91 across 108 different maze configurations with a significantly low memory footprint of 230 KB, a latency of 86.7 ms while consuming 52.23 mW of power at a 5 MHz clock rate and 5.78 ms latency with the peak power efficiency of 102.34 GOp/s/W at 75 MHz and 75% pruning.**

***Index Terms*—BCI, Hand kinematics, Regression, tinyML accelerator, RAMAN, Sparsity, Energy-efficient, FPGA**

I. Introduction

Movement disorders encompass a range of conditions that affect the ability to control voluntary movements. These disorders, including stroke, paralysis, and spinal cord injuries, are characterized by intricate pathophysiological processes involving damage to the central nervous system, disruption of neural connections, and impaired motor control circuits [1], [2]. While medical research has made remarkable progress in understanding the underlying mechanisms of movement disorders, effective treatments remain limited. Studies have revealed that even the mental imagery of motor movements, known as motor imagery (MI), activates the same neural circuits as actual physical movement [3], [4]. These findings and the recent advancements in brain-computer interfaces (BCI) and neurostimulation techniques offer promising avenues for treating movement disorders.

Invasive BCI systems, as depicted in Figure 1, offer superior spatiotemporal resolution by directly recording intracortical and cortical brain activities [3], [5] while also providing the capability for direct brain stimulation. It also leverages signal processing and ML techniques to process neural data and decode the intended motor actions. This combination allows a closed-loop feedback system to significantly enhance neuroplasticity and restore motor control. The primary objective of an implantable BCI system is to achieve surgical placement within the brain. This necessitates a hardware-software codesign approach that prioritizes compactness, high accuracy, robustness, adaptability, real-time operation, low power consumption, and the ability to handle many recording channels [6].

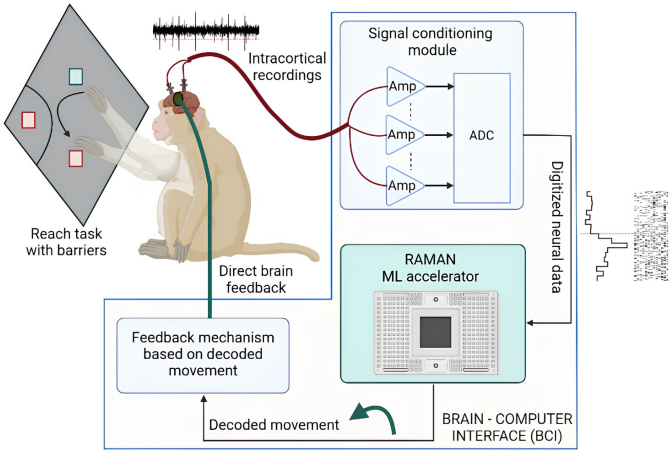

Fig. 1: Closed-loop BCI system: Real-time recording, decoding, and feedback. Decoding is performed by the RAMAN-based tinyML accelerator (highlighted in green) by extracting useful features from the digitized neural data to decode hand kinematics.

The ML accelerator plays a critical role in BCI, and its design and implementation are greatly influenced by the factors mentioned earlier. These factors guide the objectives of achieving low memory requirements and minimizing multiply-accumulate (MAC) operations to reduce power consumption while simultaneously ensuring high decoding performance. Increasing the recording capacity of BCI systems leads to a substantial increase in data dimensionality [6]. The BCI hardware, especially the ML accelerator, must exhibit scalability to handle high-dimensional data without compromising power consumption and physical space. Furthermore, the ML algorithm should be robust to mitigate the risk of model overfitting as data dimensionality increases [7], [8].

*Equal contribution.

Many studies in motor action classification, both on-chip and off-chip, emphasize the use of separate feature extraction and classification stages [9]–[15]. However, the dependence on handcrafted feature engineering presents challenges in managing large datasets and does not fully capture the variability of neural signals, thereby limiting the system's scalability for diverse individuals and tasks. The current landscape of on-edge deep learning algorithm implementations, such as convolutional neural networks (CNN), primarily focuses on classification tasks with binary or quaternary classes [16]. These implementations often utilize shallow CNN architectures with fewer layers to address memory limitations [17]. While hardware-friendly architectures like EEGNet [18] have shown successful deployment on field-programmable gate array (FPGA) platforms through techniques like quantization and efficient resource utilization, they lack important features such as pruning, leveraging weights and activation sparsity, and configurability for optimizing accuracy, power, and latency trade-offs [19]–[21]. While on-edge implementations of ML algorithms for classification tasks in hand decoding have been widely explored, there has been relatively less focus on addressing the more challenging task of real-time on-edge decoding of continuous hand kinematics. Certain approaches attempt to tackle this regression problem by converting it into a classification problem by quantising kinematic signals [6], which may not yield optimal results. A few studies with decoded hand kinematics often employ sophisticated networks like recurrent neural networks (RNN) and transformer models, although on-edge demonstrations are missing [22], [23].

In this work, we address the challenge of on-edge decoding of continuous hand kinematics by utilizing a tinyML accelerator named RAMAN, a *R*e-configurable and sp*A*rse tiny*M*L *A*ccelerator for infere*N*ce proposed in [24]. The architecture is designed to handle both classification and regression tasks on neural signals with the following features: (a) It leverages weight and activation sparsity to reduce storage, power, and latency, (b) Inspired by Gustavson's algorithm [25], RAMAN employs a dataflow that maximizes input and output activation reuse, minimizing memory access and data movement costs, (c) It offers an instruction memory and a dedicated instruction set to store and program different network topologies, and (d) Our hardware-software co-design approach incorporates hardware-aware weight pruning to improve MAC utilization and performs quantization of weights and activations to enhance hardware efficiency.

The structure of this paper is as follows. Section II describes the details of the dataset. Section III presents the software model architecture and implementation details. In Section IV, we delve into the design and implementation details of the FPGA. The results and a comprehensive comparison study are outlined in Section V. Section VI concludes the paper and summarises the key findings and contributions.

II. Dataset Description

We utilized the MC_Maze dataset [26], which comprises four recording sessions of a macaque engaging in delayed centre-out reaches. During the experiment, the macaque was seated on a chair, fixating its eyes on a frontoparallel screen positioned $\sim$ 27cm away. The experiment started with a fixation period followed by target on-set, which involved presenting a maze configuration to the macaque, consisting of at most three targets, with only one target being reachable, as depicted in Fig. 1. Upon appearance, the targets underwent slight jittering for a variable duration ranging from 0 to 1000 ms, allowing the macaque to plan its movement. The cessation of the jitter served as the cue for the macaque to initiate hand movement and perform the reach, which lasted approximately 200 to 600 ms, depending on the maze configuration. Each recording session comprised 27 different maze configurations, resulting in a variety of reach scenarios. These maze configurations included: (a) configurations with a single reachable target and no barriers, (b) configurations with a single reachable target and barriers, and (c) configurations with at most three targets and barriers, of which one was reachable while the others were not. This paradigm allowed the examination of neural activity during movement preparation and execution.

Neural activity was recorded from the dorsal premotor cortex and primary motor cortex of the macaque using two 96-channel Utah arrays. The recorded signals were sampled at 30 kHz and underwent offline spike sorting, followed by manual evaluation of sorted units based on stability and signal quality. Additionally, cursor position, eye position, hand position, target onset time, go cue time, and movement onset time were recorded. Hand velocity was calculated after data capture by differentiating the hand position over time.

III. Model: Architecture and Implementation

We employ the MobileNetV1 [27] architecture for predicting the hand velocity using the neural spiking data. This architecture uses depthwise separable convolutions comprising depth-wise (DW) and point-wise (PW) convolutions as building blocks with lower parameters and MAC operations than a standard convolution (CONV). We thin out the base MobileNetV1 architecture by scaling down the number of channels in each layer by a factor of $0.25\times$. Our experiments showed that this significantly reduced our solutions' computational complexity and latency without affecting the performance. It is well known that neural networks are over-parameterized, and many connections can be removed without any significant loss in accuracy [28]. Most MAC operations (92.64%) are distributed in our network's PW layers. We implement a structured pruning strategy to remove a fixed percentage of output channels in these layers in an iterative manner. Neural networks have many degrees of freedom, making them suitable candidates for compression [29]. We compress our model to 8-bit, making them smaller and more efficient. We perform quantization-aware training (QAT) to reduce errors introduced during quantization, which emulates the behavior during inference-time quantization. Since convolution and batch norms are linear operations, we fold their parameters together to reduce unnecessary parameters and operations during inference. These architectural design considerations reduce our model size from 3.23M to 0.06M

parameters and 32-bit to 8-bit, providing greater than 99% compression and latency reduction.

We resample the original MC_Maze dataset using a bin size of 5 ms and create trials considering a window size of (-130 ms, 370 ms) around movement onset time. We split the resulting 2295 trials using a 60/20/20 ratio to create the train, validation, and test sets. The spiking data is smoothed using a Gaussian kernel of 50 ms, and the hand velocity is lagged by 80 ms before splitting into trials. We implement channel dropout as a data augmentation technique wherein we set a certain number of the spiking channels to zero before feeding it to the network. We use the Adam optimizer [30] with an initial learning rate of 0.01 for training and pruning our model and scale this down by a factor of 100 for quantization-aware training. We use the L1-loss for optimization and reduce the learning rate by half upon loss saturation. We prune our model to 75% sparsity iteratively in 20 steps, finetuning the model for 100 epochs after each step. To determine the scales for compression, we consider the max values of the tensor and perform per-channel quantization. As a post-processing step, we smoothen the predictions of our network using a moving average filter of window size 5. We split a single trial into five slices and performed computations on them.

IV. FPGA Implementation

The top-level architecture of the RAMAN tinyML accelerator is shown in Fig. 2. The design consists of Compute, Memory and Control subsystems. The computing subsystem encompasses a processing element (PE) array to perform MAC operations, an activation sparsity engine to leverage sparsity in activations, and a post-processing module (PPM) for the rectified linear unit (ReLU) activation, quantization, pooling, bias addition, and residual addition. The memory subsystem comprises on-chip global memory to store layer parameters and input/output activations of the network, activation and parameter cache to exploit temporal data reuse and instruction memory to store layer configuration instructions for programming RAMAN. The control subsystem includes a top-level controller coordinating data traffic and issuing control signals to individual modules. A detailed description of each module's functionality is provided in [24].

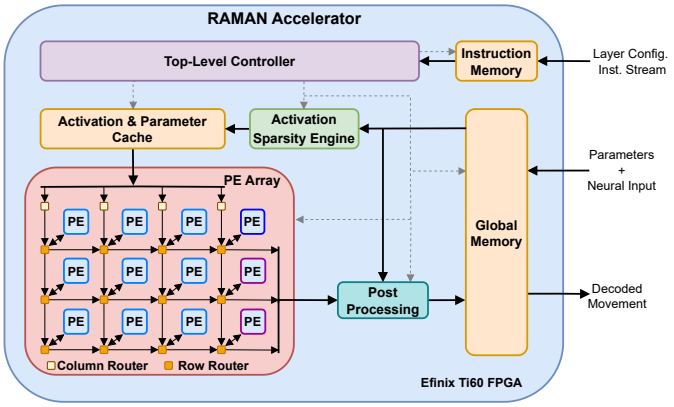

Fig. 2: RAMAN top-level architecture.

V. Results

This section presents our approach's quantitative and qualitative results and the resource utilization details on the FPGA.

TABLE I: Quantitative comparison of our model (0.25× width multiplier, 75% pruning, 8-bit quantized, and operation folding). The hyperparameter tuning of the Ridge Regression model is done using grid search. Our model achieves the best R2-score with significantly reduced size and latency. The MAC operations are listed in Millions.

Model	Parameters	MAC	R2-Score(↑)
Ridge Regression (RR)	366	0.08	0.64
Base MobileNetV1	3.23M	618.39	0.91
Ours	*0.063M*	*40.92*	*0.91*

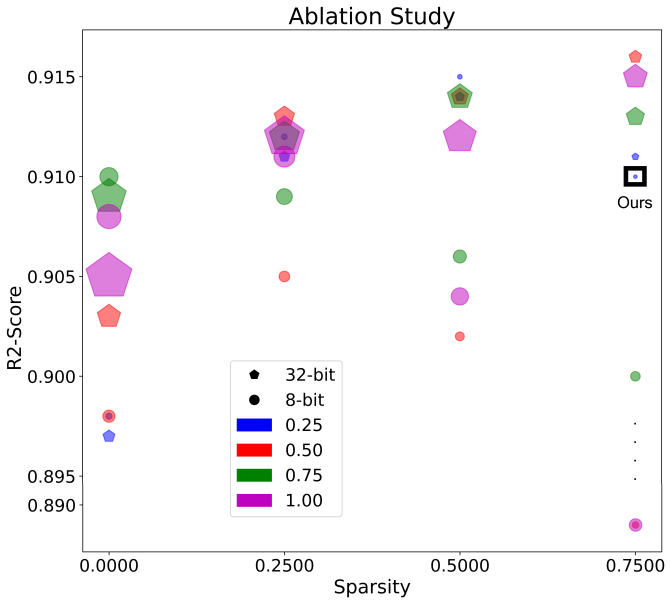

Fig. 3: Ablation study of our approach for different width multipliers, pruning ratios, and bit-widths. The size of the dots is proportional to the model size.

A. Benchmarking

Table I compares our model's computational efficiency and task performance against baseline approaches. We can see that the Ridge Regression (RR) model is not sufficiently complex enough to capture the relationship between the neural data and hand kinematics. Our model can surpass the performance of the Base MobileNetV1 model across all metrics supporting our architectural modification choices. Fig. 3 shows the ablation results of our neural network. We can see that pruning is helping the model generalize better, achieving a higher R2-score on the test set.

Interestingly, the performance of larger models with higher sparsities is more adversely affected by quantization. We compare the predicted hand velocities of the Ridge Regression model and ours against the ground truth (GT) in Fig. 4. We can see that the trajectories predicted by our model are consistent with the ground truth, whereas the Ridge Regression predictions have a significant offset. The Pearson correlation coefficients of our model predictions are significantly higher, especially for the hand velocity along the y-direction.

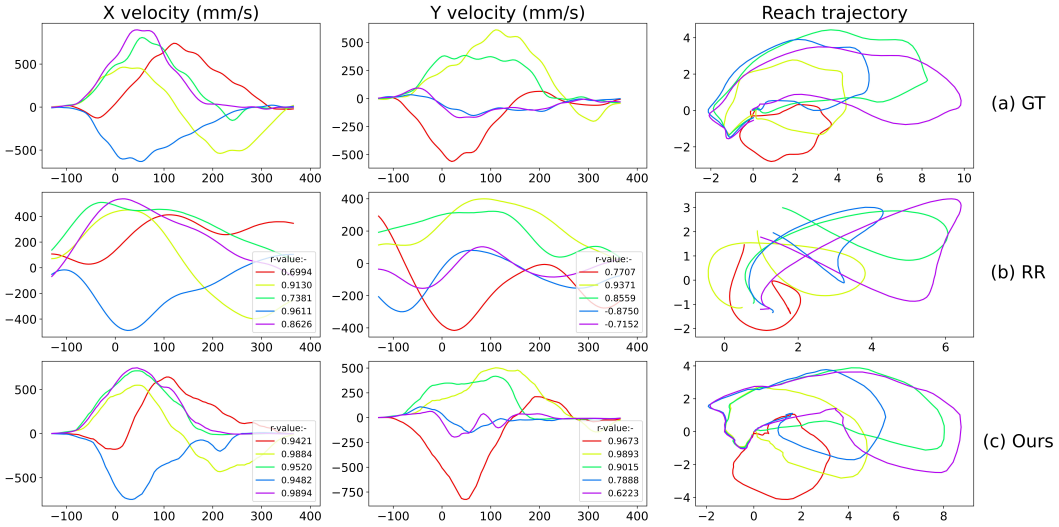

Fig. 4: Comparison of the ground truth (GT) against Ridge Regression (RR) and our model predictions. Each color in the plot represents a different trial. In the legend, we show the Pearson correlation coefficient (r-value) for the predictions against ground truth.

B. FPGA Implementation Results

The MobileNetV1 model, trained on the MC_Maze dataset with a width multiplier of 0.25×, was programmed on RAMAN and subsequently deployed on an Efinix Ti60 FPGA [31]. The hardware specifications of the RAMAN architecture and resource utilization details are presented in Table II.

TABLE II: Specifications and resource utilization.

Platform	Efinix Ti60
Layers Supported	CONV, DW, PW, FC and Max/Average pooling.
Number of PEs	12 (4 MACs/PE)
Reg-file Memory	0.896 KB
Clock Rate	5 - 75 MHz
Precision	Weights & Activations: 8b fixed point, Partial-sums: 24b fixed point.
Power (mW)	75 MHz: 138.25 (Dynamic: 90.6, Static: 47.65) 5 MHz: 52.23 (Dynamic: 6.8, Static: 45.43)
XLR cells	52261 (85.96% util.), 37.23k LUTs & 8.6k FFs
DSPs	61 (38.12% util.)
Memory Blocks (10Kb Blocks)	184 (Parameters: 90, Activations: 60, Cache: 27, Rest: 7)
MAC Operations (in Millions)	40.9 (CONV: 0.131, DW: 2.86, PW: 37.89, FC: 0.0102)
Throughput	14.16 GOp/s @75MHz
Latency (ms)	Without sparsity: 13.5 @75MHz, 202.5 @5MHz With sparsity: 5.78 @75MHz, 86.7 @5MHz
Parameter Memory	No pruning: 221.65 KB 75% PW Pruning: 100.88 KB
Peak Efficiency	102.34 GOp/s/W (1251 Inferences/J) @75MHz.

The PW layer is the most computationally intensive, as depicted by Table II. Thus, we exploit sparsity specifically in the PW layer in reducing latency by skipping processing cycles with zero activations and weights and minimizing storage by applying weight pruning and model compression techniques at the compile time. These zero skipping and model compression schemes reduce latency by 57% and memory storage by 54%, respectively. The power consumption is estimated to be 138.25 mW at 75 MHz; however, the latency reductions achieved by leveraging sparsity enable the design to operate at a much lower frequency lowering the power consumption. The architecture could run as low as 5 MHz for our target application (decoding hand kinematics) and still achieve the required latency target by consuming just 52.23 mW (6.8 dynamic + 45.43 static) of power. The theoretical peak throughput of the design is 7.2 GOp/s, with 48 MACs operating at 75 MHz. However, since operations involving zero are skipped in the PW layer, we achieve an effective throughput of 14.16 GOp/s with a peak power efficiency of 102.34 GOp/s/W. The architecture utilizes 37.23k 4-input look-up tables (LUTs) and 8.6k flip flops (FFs), which are mapped as eXchangeable Logic and Routing (XLR) cells on Efinix FPGA.

VI. Conclusions

In this study, we introduced RAMAN, an energy-efficient tinyML accelerator for continuous hand kinematics decoding. Our approach, incorporating depthwise separable convolutions and architectural design strategies like channel thinning, pruning, and quantization-aware training, achieved significant improvements in model size, complexity, latency, and maintained a high-performance level with an R2-score of 0.91. RAMAN on Efinix Ti60 FPGA demonstrated peak power efficiency (102.34 GOp/s/W at 75 MHz), and consumed 52.23 mW power at a 5 MHz clock rate with a 230 KB memory footprint. The results highlight RAMAN's effectiveness and efficiency for real-time movement analysis and brain-computer interfaces, with future plans for integration into the BCI pipeline for closed-loop testing.

VII. Acknowledgements

We acknowledge the financial support provided by Pratiksha Trust under grant FG/SMCH-22-2106 in facilitating this research.

REFERENCES

[1] S. J. Murphy and D. J. Werring, "Stroke: causes and clinical features," *Medicine (Abingdon)*, vol. 48, no. 9, pp. 561–566, Aug. 2020.

[2] R. Mane, T. Chouhan, and C. Guan, "BCI for stroke rehabilitation: motor and beyond," *J Neural Eng*, vol. 17, no. 4, p. 041001, Aug. 2020.

[3] S. N. Flesher, J. E. Downey, J. M. Weiss, C. L. Hughes, A. J. Herrera, E. C. Tyler-Kabara, M. L. Boninger, J. L. Collinger, and R. A. Gaunt, "A brain-computer interface that evokes tactile sensations improves robotic arm control," *Science*, vol. 372, no. 6544, pp. 831–836, May 2021.

[4] F. Di Rienzo, C. Collet, N. Hoyek, and A. Guillot, "Impact of neurologic deficits on motor imagery: a systematic review of clinical evaluations," *Neuropsychology review*, vol. 24, pp. 116–147, 2014.

[5] F. R. Willett, D. T. Avansino, L. R. Hochberg, J. M. Henderson, and K. V. Shenoy, "High-performance brain-to-text communication via handwriting," *Nature*, vol. 593, no. 7858, pp. 249–254, May 2021. [Online]. Available: https://doi.org/10.1038/s41586-021-03506-2

[6] M. Shaeri, A. Afzal, and M. Shoaran, "Challenges and Opportunities of Edge AI for Next-Generation Implantable BMIs," in *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2022, pp. 190–193.

[7] F. Lotte, M. Congedo, A. Lécuyer, F. Lamarche, and B. Arnaldi, "A review of classification algorithms for EEG-based brain–computer interfaces," *Journal of Neural Engineering*, vol. 4, no. 2, p. R1, jan 2007. [Online]. Available: https://dx.doi.org/10.1088/1741-2560/4/2/R01

[8] 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, apr 2018. [Online]. Available: https://dx.doi.org/10.1088/1741-2552/aab2f2

[9] C. L. León, "Multilabel classification of EEG-based combined motor imageries implemented for the 3D control of a robotic arm," Ph.D. dissertation, Université de Lorraine, 2017.

[10] C. Lindig-Leon and L. Bougrain, "A multi-label classification method for detection of combined motor imageries," in *2015 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 2015, pp. 3128–3133.

[11] G. Liu, D. Zhang, J. Meng, G. Huang, and X. Zhu, "Unsupervised adaptation of electroencephalogram signal processing based on fuzzy C-means algorithm," *International Journal of Adaptive Control and Signal Processing*, vol. 26, no. 6, pp. 482–495, 2012.

[12] A. Schlögl, C. Vidaurre, and K.-R. Müller, "Adaptive methods in BCI research-an introductory tutorial," *Brain-Computer Interfaces: Revolutionizing Human-Computer Interaction*, pp. 331–355, 2010.

[13] U. Shin, L. Somappa, C. Ding, Y. Vyza, B. Zhu, A. Trouillet, S. P. Lacour, and M. Shoaran, "A 256-Channel 0.227 μJ/class Versatile Brain Activity Classification and Closed-Loop Neuromodulation SoC with 0.004 mm 2-1.51 μW/channel Fast-Settling Highly Multiplexed Mixed-Signal Front-End," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 65. IEEE, 2022, pp. 338–340.

[14] B. Zhu, M. Farivar, and M. Shoaran, "Resot: Resource-efficient oblique trees for neural signal classification," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 14, no. 4, pp. 692–704, 2020.

[15] A. R. Aslam and M. A. B. Altaf, "A 10.13 μJ/Classification 2-channel deep neural network based SoC for negative emotion outburst detection of autistic children," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 15, no. 5, pp. 1039–1052, 2021.

[16] W.-C. Fang, K.-Y. Wang, N. Fahier, Y.-L. Ho, and Y.-D. Huang, "Development and validation of an EEG-based real-time emotion recognition system using edge AI computing platform with convolutional neural network system-on-chip design," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 4, pp. 645–657, 2019.

[17] A. Bahr, M. Schneider, M. A. Francis, H. M. Lehmann, I. Barg, A.-S. Buschhoff, P. Wulff, T. Strunskus, and F. Faupel, "Epileptic seizure detection on an ultra-low-power embedded RISC-V processor using a convolutional neural network," *Biosensors*, vol. 11, no. 7, p. 203, 2021.

[18] 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," *Journal of neural engineering*, vol. 15, no. 5, p. 056013, 2018.

[19] L. Feng, H. Shan, Y. Zhang, and Z. Zhu, "An Efficient Model-Compressed EEGNet Accelerator for Generalized Brain-Computer Interfaces with Near Sensor Intelligence," *IEEE Transactions on Biomedical Circuits and Systems*, 2022.

[20] A. Tsukahara, Y. Anzai, K. Tanaka, A. Homma, and Y. Uchikawa, "A Design and Trial Production of EEGNet based EEG Pattern Recognition Processor using FPGA," in *2020 IEEE 8th R10 Humanitarian Technology Conference (R10-HTC)*. IEEE, 2020, pp. 1–4.

[21] A. C. Hernandez-Ruiz, D. Enériz, N. Medrano, and B. Calvo, "Motor-Imagery EEGNet-Based Processing on a Low-Spec SoC Hardware," in *2021 IEEE Sensors*. IEEE, 2021, pp. 1–4.

[22] M. R. Keshtkaran, A. R. Sedler, R. H. Chowdhury, R. Tandon, D. Basrai, S. L. Nguyen, H. Sohn, M. Jazayeri, L. E. Miller, and C. Pandarinath, "A large-scale neural network training framework for generalized estimation of single-trial population dynamics," *Nature Methods*, pp. 1–6, 2022.

[23] J. Ye and C. Pandarinath, "Representation learning for neural population activity with Neural Data Transformers," *arXiv preprint arXiv:2108.01210*, 2021.

[24] A. Krishna, S. R. Nudurupati, D. G. Chandana, P. Dwivedi, A. van Schaik, M. Mehendale, and C. S. Thakur, "RAMAN: A Re-configurable and Sparse tinyML Accelerator for Inference on Edge," *arXiv*, 2023. [Online]. Available: http://arxiv.org/abs/2306.06493

[25] G. Zhang, N. Attaluri, J. S. Emer, and D. Sanchez, "Gamma: Leveraging Gustavson's algorithm to accelerate sparse matrix multiplication," in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2021, pp. 687–701.

[26] M. Churchland and M. Kaufman, "MC_Maze: macaque primary motor and dorsal premotor cortex spiking activity during delayed reaching," [Data set], 2022. [Online]. Available: https://doi.org/10.48324/dandi.000128/0.220113.0400

[27] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[28] V. Ramanathan, P. Dwivedi, B. Katabathuni, A. Chakraborty, and C. S. Thakur, "QUICKSAL: A small and sparse visual saliency model for efficient inference in resource constrained hardware," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 1678–1688.

[29] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," *arXiv preprint arXiv:2103.13630*, 2021.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[31] Efinix, "TI60 Data Sheet," Online, 2023. [Online]. Available: https://www.efinixinc.com/docs/titanium60-ds-v2.7.pdf