# Energy Efficient Convolutional Neural Networks for EEG Artifact Detection

Mohit Khatwani[1], M. Hosseini[1], H. Paneliya[1], W. David Hairston[2], Nicholas Waytowich[2], and Tinoosh Mohsenin[1]

[1]Department of Computer Science & Electrical Engineering, University of Maryland, Baltimore County
[2]Human Research and Engineering Directorate, US Army Research Lab

***Abstract*—This paper proposes an energy efficient Convolutional Neural Network based architecture for detecting different types of artifacts in multi-channel EEG signals. Our method achieves an average artifact detection accuracy of 74% and precision of 92% across seven different artifact types which outperforms existing techniques in terms of classification accuracy as well as the more common ICA based solution in terms of computational complexity and memory requirements. We designed a minimal neural network processor whose Verilog HDL is configurable for implementing $2^n$ processing engines (PEs). We deployed our CNN on the processor, placed and routed on Artix-7 FPGA and examined different number of PEs at different operating frequencies. Our experiments indicate that utilizing 4 PEs operating at a clock frequency of 11.1 MHz is the optimal configuration for our hardware to yield the least classification energy consumption of 32 mJ accomplished in the maximum allowed prediction time of 1 Sec. We also implemented our CNN on TX2 NVIDIA Jetson platform and, by tweaking the CPU and the GPU frequencies, explored the minimum power and energy configuration. Our FPGA results indicate that the 4-PE implementation outperforms the low power configuration of TX2 by 65× in terms of power, and the low energy configuration of TX2 by 2× in terms of energy per classification. Our CNN-based FPGA implementation method also outperforms the ICA method by 11× in terms of energy consumption per classification.**

## I. Introduction

Electroencephalography (EEG) is one of the most popular neurophisiological recording techniques due to it's non-invasiveness, easy of use and relatively low cost. EEG is used in a wide variety of neuroscience and engineering applications such as the monitoring of brain disorders, detecting of stress or fatigue, and in brain-computer interfaces for augmentative communication [1]. One of the primary drawbacks of EEG is it's proneness for accumulating physiological and non physiological artifacts ranging from muscle activity to power line electrical noise [2]. These artifacts severely deteriorate the quality of underlying EEG signals and thus efficient detection and removal of these artifacts remains a critical issue for developing practical EEG-based neurotechnologies. In addition, these neurotechnologies will need to be deployed on low-power, embedded systems that a user can easily wear, requiring power efficient EEG processing algorithms.

Over the years there have been several approaches developed for the detection and isolation of artifacts within EEG, the most popular of which is Independent Component Analysis (ICA) [3]. Auto-regressive models have also recently been applied with linear classification models for artifact detection [4]. These techniques, however, are not fully automated and still require an expert person to laboriously label and tag the EEG artifacts. Recently, Jafari et al proposed an automated online artifact detection technique based on ICA and multi-instance learning (MIL) classifier that achieved a 91.2% artifact detection rate for eye brow raising artifacts in 8 seconds when running on a CPU [5]. ICA-based techniques require large datasets in addition to manual identification of artifacts. Additionally, the large computational complexity and memory requirements for the ICA make it not suited for real-time applications on embedded hardware [6].

Convolutional neural networks (CNNs), which have primarily been used for image classification, have recently become a popular technique for processing and classifying EEG signals [7], [8], [9]. In this paper, we develop an energy efficient CNN for artifact detection and implement it on embedded hardware. We propose a CNN-based model which is able to detect multiple artifacts from multi-channel EEG data, and explore hardware platforms for an energy efficient implementation. Specifically, we design a specialized hardware architecture for neural network inference composed of a main on-chip memory, that accommodates the neural network model as well as the associated feature maps, and is reconfigurable for implementing $2^n$ processing engines (PEs). We also explore the least number of PEs that meet the a processing time-limit of 1 Sec with the lowest energy consumption per forward pass. We compare our CNN-based model with optimal hardware implementation on an FPGA to a commercial of-the-shelf (COTS) implementation on an Nvidia TX2 board. The main contributions of this paper are twofold:

1) An energy efficient CNN architecture for automated EEG artifact detection with low computational and memory requirements.
2) A configurable hardware architecture with $2^n$ Processing Engines (PEs) implemented in Verilog, placed and routed on Artix-7 FPGA.

## II. EEG data and Artifacts

In order to assess and evaluate the accuracy of our technique, we used a previously recorded EEG dataset where participants manually performed a series of different occular or muscular artifacts (i.e. jaw clenching, eye blinking, etc.) EEG

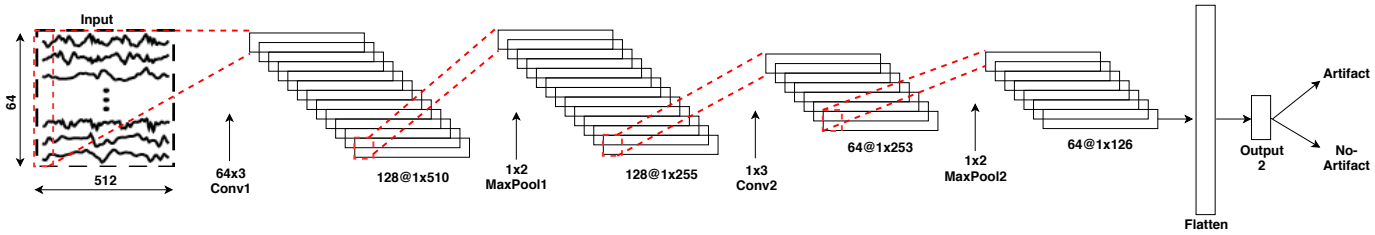


Fig. 1: Proposed CNN Architecture which consists of 5 layers. 2 convolutional layers, 2 max pool layers and 1 softmax layer.

data were recorded using a 64 channel BioSemi ActiveTwo system with a sampling rate of 512Hz. Participants were required to perform a series of noise-inducing body, facial, head or eye movements, which were gathered as part of a larger study [10]. It was up to the participants to determine the precise choreography of each movement and to perform movements which felt more natural to them. Each movement was performed as a separate set of 20 repetitions. A screen was put in place in order to remind the participants of the movement they should make. A male voice initially counted down from 3 at a rate of every 2 Sec followed by a tone every 2 Sec. This procedure was done for each set. The participants would make the movements in time with the vocal commands. They were advised to perform the tasks in the first second of the 2 Sec period and to relax in the remaining 1 Sec. Additionally, each participant performed a baseline recording session where they were instructed to keep still and look straight at a blank computer screen for around 8 seconds at the start of every run. EEG data from this baseline session was used as "clean" (or artifact-free) data. Artifacts considered are clenching jaw (CJ), move jaw (MJ), blink eyes (BE), move eyes leftwards (EL), move eyes rightwards (ER), raise eyebrows (RE) and rotate head (RH).

## III. CNN-based Artifact Detection Architecture

### A. EEG Pre-processing

Our CNN operates on the raw EEG data so that it can learn to extract the relevant features required for artifact detection. As such, minimal pre-processing of the raw EEG data was performed aside from removing the DC offset such that the EEG signals are centered around zero. EEG epochs of size 64×512 were extracted from the artifact and baseline data creating a total of around 250 trials per subject for each artifact type.

### B. Neural Network Architecture

The input to our CNN is a two dimensional EEG epoch (64 channels × 512 time points). The full network architecture, shown in Figure 1, consists of two convolutional layers, two max-pool layers and ends with a softmax layer for classification. The first 2-d convolution layer consists of 128 kernels of size 64×3. this ensures that the an adequate spatial filter is learned in the first layer. The second 2-d convolution layer learns 64 kernels of size 1×3. Each convolution layer is followed by a max-pooling layer with a pool size of 1×2. All layers are followed by a rectified linear unit (ReLU) activation function. The output of the second max pooling layer is then flattened into a single vector and passed to a softmax layer consisting of two outputs for binary classification to detecting whether or not an artifact is present in the EEG data.

All the layers of the network have their weights initialized from a normal distribution. The network was trained using the RMSprop optimization method and a learning rate of 0.0001. Categorical cross-entropy was used as the loss function. In total, the network has 65280 parameters, and requires 35.4 million operations (either multiplication or addition) in order to process one input frame. Note, that this is much lower than CNNs for image classification which usually contain millions of parameters [11].

## IV. Classification Analysis and Results

Our architecture is evaluated for 9 patients for 7 different artifacts using a transfer learning setting where models are trained and tested on EEG data from different subjects. This allows us to test the ability of our CNN model to generalize across subjects when detecting EEG artifacts, a task that is difficult for traditional methods. We trained a cross-subject model using a leave-one-subject-out cross validation procedure where data from one subject was held out for testing and data from the remaining subjects were used for training. On average 3790 samples where used for training and 490 samples where used for testing. We compare our results with previous work that uses an autoregresive model + SVM classifier for artifact detection on the same dataset [4].

Results in Table I show that our model detects all 7 artifacts by average accuracy of 73.6% which ranges between 64.3% and 84.8%. The raise and lower eyebrows artifact was easiest to detect with average accuracy of 84.8%, while the moving jaw had lowest detection accuracy of 64.3%. In Table I the autoregressive (AR) [4] baseline technique achieved 68.42% of average accuracy using the leave-one-subject-out cross-validation technique which ranges between 52% and 95%. Our CNN model achieves a statistically significant improvement in detection accuracy on 5 of the 7 EEG artifacts using a one sample, two-sided t-test with a significance threshold of $p < 0.05$.

In the next section we explore the most suitable platform that is most power/energy efficient for our CNN model. For this purpose, we implement our model on both an FPGA and an Nvidia TX2 with various configurations in terms of number of PEs and/or frequency range in order to find an implementation that meets the application deadline (<1 Sec classification window) with the least power and energy consumption budgets.

TABLE I: Detection accuracy and precision for different artifacts averaged for all patients using leave one subject out cross-validation technique among 9 patients. Results are compared against Auto-Regressive baseline technique [4]. Values in the parentheses indicate the standard deviation. Asterisks (*) indicate significant accuracy improvement over the AR technique.

| Artifact Code | CJ | MJ | BE | EL | ER | RE | RH | Average |
|---|---|---|---|---|---|---|---|---|
| **This work: Accuracy** | 74.1(9.2) | 64.3(11.8) | **76.1(10.9)*** | **77.5(9.6)*** | **72.6(9.1)*** | **84.8(7.9)*** | **73.3(14.2)*** | **73.6(11.8)*** |
| **AR: Accuracy [4]** | 95* | 76 | 74 | 49 | 52 | 75 | 58 | 68.42 |
| **This work: Precision** | 94.7(5.1) | 91.4(9.8) | 94.3(5.6) | 94.0(5.6) | 92.6(7.8) | 99.1(1.1) | 92.7(6.1) | 92.0(7.3) |

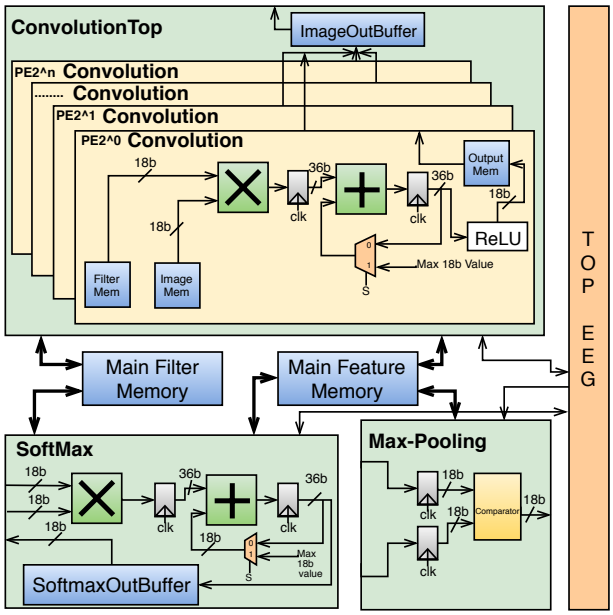


Fig. 2: Block diagram of FPGA hardware used to implement the proposed CNN model. The hardware architecture includes a top-level state-machine which controls the convolutionTop, convolution, max-pooling and Softmax blocks as well as all memory blocks. PE refers to number of convolutions that can be processed in parallel. Here n is an integer which ranges from 0 to 3.

## V. Hardware Architecture Design

The details of our hardware architecture and implementation are shown in Figure 2. The hardware consists of one shared memory and is reconfigurable to have up to $2^n$ processing engines (PE), where n is an integer ranging from 0 to 3. The primary operational components of the hardware consist of: (1) the Convolution block that performs the convolution layer operations with ReLU activation logic, (2) the Max-Pooling block that performs max-pooling operations and (3) the Fully-Connected block that includes a softmax activation function in the last layer for classification.

As figure 2 shows, the Convolution block consists of one adder/subtractor, one multiplier, one cache for saving filters, input feature maps, output feature maps, a multiplexer and a state machine block. The input EEG data for the hardware model are converted to 18-bit fixed-point precision. We use 18-bit format because that's the default data format inside the block RAMs (BRAMs) for most Xilnix FPGAs. The model weights, trained offline on a standard machine using tensorflow, are converted from 32-bit floating point values to 18-bit fixed-point values. This serves to increase computational efficiency since floating-point arithmetic is complex in hardware and requires more resources and execution time. EEG data is then passed through the Convolution block to be processed by the convolution and ReLU activation functions. The output of the ReLU function is truncated to 18-bits and saved into the feature map memory. This data is then pased as input to the The Max-Pooling block, which contains a few registers and a comparator. After max-pooling, the results are saved into the main feature map memory. Finally, the Fully-Connected block, which is used only in the last layer of the neural network in this work, consists of a serial dot product engine and dynamic sorting logic for the softmax activation function. After finishing computation for the softmax layer, the results are saved into the main memory which overwrites the outdated intermediate data from previous layer.

## VI. Hardware Implementation and Results

### A. FPGA implementation

The hardware abstraction is implemented using Verilog HDL, with 1, 2, 4, and 8 number of Processing Engines (PEs), and each configuration is synthesized, placed and routed on low power Artix-7-200T FPGA using Xilinx Vivado tools. The choice of FPGA was such that its on-chip memories are sufficient to store the neural network model as well as the intermediate data. We implemented our case study with different numbers of PEs and decided that 4 is the optimal amount that not only meets the 1 Sec processing time limit for this application but also yields the lowest energy consumption per forward pass through the network.

Table II provides the implementation results for 1PE, 2PE, 4PE and 8PE. The clock frequency for each PE configuration is set such that the prediction deadline of 1 Sec is met. The results show that the least amount of power and energy consumption are obtained from PE4 operating at 11.1 MHz with the maximum allowed execution process time of 1 Sec. Figure 3 demonstrates the impact of increasing number of PEs on power, energy and processing latency using a fixed clock frequency of 37.7 MHz. Using our CNN-based model implementation, we show that increasing the number of PEs leads to an increase in power consumption, a decrease in processing latency, and a upward concave shape for energy consumption with the PE4 configuration consuming the least amount of energy. A power consumption breakdown of post-place and route implementation for 4PE on the Artix-7 FPGA was calculated using Vivado power tool. 83% of the power was consumed by the on-chip BRAM memories of the FPGA.

TABLE II: Implementation results on Xilinx Artix-7 FPGA with different number of PEs. For each number of PE config. The frequency is set such that the processing takes near 1 second to detect the artifact and meet the deadline.

| | Config.1 | Config.2 | Config.3 | Config.4 |
|---|---|---|---|---|
| **No. of PEs** | 1 | 2 | **4** | 8 |
| **Frequency (MHz)** | 37.7 | 18.5 | **11.1** | 5.55 |
| **Power (mW)** | 54 | 36 | **32** | 35 |
| **Energy (mJ)** | 54 | 36 | **32** | 35 |
| **No. of Slices** | 3423 | 3863 | **4635** | 6576 |
| **BRAM** | 107 | 139 | **204** | 334 |
| **DSP** | 32 | 32 | **40** | 95 |

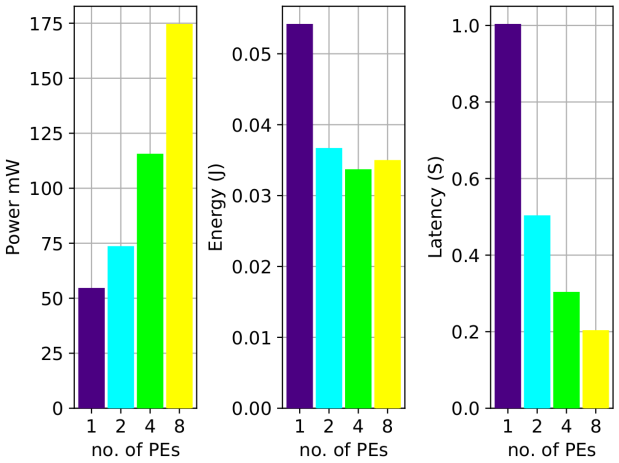


Fig. 3: Effect of number of used PE on power, latency and energy consumption. For all configurations in this plot, the clock frequency is set to 37.7 MHz

### B. TX2 implementation

In order to evaluate the energy-latency trade-off on other COTS products, the trained tensorflow model was deployed on an Nvidia Jetson TX2 SoC with different frequency selection for the CPU cores and for the GPU. The TX2 board features a 256-core Pascal GPU, 8GB of LPDDR4 memory with a 128-bit interface, and a CPU complex that combines a dual-core NVIDIA Denver 2 and a quad-core ARM Cortex-A57. The result of our model implementation of the TX2 board, shown in Table III, indicates that the lowest power consumption is obtained when both the CPU and GPU are set to their minimal frequency range, whereas the minimum energy per classification is obtained when the CPU and GPU are configured to their highest performance.

## VII. Conclusion

In this paper, we proposed an energy efficient CNN-based architecture for detecting artifacts in EEG that achieved an average accuracy of 74% across all artifacts and subjects. On average, our CNN architecture significantly outperformed the baseline auto-regressive method [4] on the majority of the tested artifact types. Additionally, our CNN method is a fully automated technique which doesn't require manual labeling of the EEG trials which was required in [4]. We implemented the CNN processing hardware in Verilog, reconfigurable with

TABLE III: Implementation results of power, energy and latency on TX2 board with different CPU and GPU frequency setting. All four combinations of the minimum and maximum frequency for the CPU and GPU are tested.

| | Config.1 | Config.2 | Config.3 | Config.4 |
|---|---|---|---|---|
| **CPU Freq. (MHz)** | 346 | 346 | 2035 | **2035** |
| **GPU Freq. (MHz)** | 140 | 1300 | 140 | **1300** |
| **Latency (mS)** | 84.1 | 68.5 | 17.9 | **11.3** |
| **Throughput (fps)** | 11.9 | 14.6 | 55.9 | **88.5** |
| **Power (W)** | 2.1 | 2.9 | 6.1 | **6.8** |
| **Energy (mJ)** | 177 | 199 | 110 | **76** |

$2^n$ number of PEs, and determined that 4 PEs operating at frequency of 11.1 MHz implemented on Artix-7 FPGA is the optimal configuration to meet all of our application requirements (power, latency, and energy). We also implemented our CNN on the TX2 board and examined four frequency ranges for CPU and GPU. Our optimal implementation on the FPGA indicates that the 4-PE implementation is 65× more power efficient and 2× more energy efficient than TX2 when configured for the minimum power and energy consumption, respectively. When compared to other related work, [5], the energy consumption on the FPGA for one classification is 11× that of ICA method on the same dataset.

## VIII. Acknowledgement

This research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-10-2-0022.

## References

[1] B. J. Lance *et al.*, "Brain x2013;computer interface technologies in the coming decades," *Proceedings of the IEEE*, vol. 100, no. Special Centennial Issue, pp. 1585–1599, May 2012.

[2] M. K. Islam, A. Rastegarnia, and Z. Yang, "Methods for artifact detection and removal from scalp eeg: a review," *Neurophysiologie Clinique/Clinical Neurophysiology*, vol. 46, no. 4, pp. 287–305, 2016.

[3] T.-P. Jung, S. Makeig, C. Humphries, T.-W. Lee, M. J. Mckeown, V. Iragui, and T. J. Sejnowski, "Removing electroencephalographic artifacts by blind source separation," *Psychophysiology*, vol. 37, no. 2, pp. 163–178, 2000.

[4] V. Lawhern, W. D. Hairston, K. McDowell, M. Westerfield, and K. Robbins, "Detection and classification of subject-generated artifacts in eeg signals using autoregressive models," *Journal of neuroscience methods*, vol. 208, no. 2, pp. 181–189, 2012.

[5] A. Jafari *et al.*, "An eeg artifact identification embedded system using ica and multi-instance learning," in *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 1–4.

[6] R. Islam, W. D. Hairston, T. Oates, and T. Mohsenin, "An eeg artifact detection and removal technique for embedded processors," in *2017 IEEE Signal Processing in Medicine and Biology Symposium (SPMB)*, Dec 2017, pp. 1–3.

[7] V. J. Lawhern, A. J. Solon, N. R. Waytowich, S. M. Gordon, C. P. Hung, and B. J. Lance, "Eegnet: A compact convolutional network for eeg-based brain-computer interfaces," *CoRR*, vol. abs/1611.08024, 2016. [Online]. Available: http://arxiv.org/abs/1611.08024

[8] A. Jafari, A. Ganesan, C. S. K. Thalisetty, V. Sivasubramanian, T. Oates, and T. Mohsenin, "Sensornet: A scalable and low-power deep convolutional neural network for multimodal data classification," *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1–14, 2018.

[9] A. Page, C. Shea, and T. Mohsenin, "Wearable seizure detection using convolutional neural networks with transfer learning," in *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on*, 2016.

[10] K.-m. Su *et al.*, "Eeg-annotate: Automated identification and labeling of events in continuous signals with applications to eeg," *Journal of neuroscience methods*, vol. 293, pp. 359–374, 2018.

[11] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556