uBrain: A Unary Brain Computer Interface

Di Wu, Jingjie Li, Zhewen Pan, Younghyun Kim, and Joshua San Miguel
di.wu@ece.wisc.edu,{jingjie.li,zhewen.pan,younghyun.kim,jsanmiguel}@wisc.edu
Department of ECE, University of Wisconsin–Madison
Madison, WI, USA

ABSTRACT

Brain computer interfaces (BCIs) have been widely adopted to enhance human perception via brain signals with abundant spatial-temporal dynamics, such as electroencephalogram (EEG). In recent years, BCI algorithms are moving from classical feature engineering to emerging deep neural networks (DNNs), allowing to identify the spatial-temporal dynamics with improved accuracy. However, existing BCI architectures are not leveraging such dynamics for hardware efficiency. In this work, we present uBrain, a unary computing BCI architecture for DNN models with cascaded convolutional and recurrent neural networks to achieve high task capability and hardware efficiency. uBrain co-designs the algorithm and hardware: the DNN architecture and the hardware architecture are optimized with customized unary operations and immediate signal processing after sensing, respectively. Experiments show that uBrain, with negligible accuracy loss, surpasses the CPU, systolic array and stochastic computing baselines in on-chip power efficiency by 9.0×, 6.2× and 2.0×.

CCS CONCEPTS

• **Hardware** → **Emerging interfaces**; **Application specific integrated circuits**; • **Computing methodologies** → **Special-purpose algebraic systems**; **Artificial intelligence**.

KEYWORDS

Unary computing, stochastic computing, temporal computing, brain computer interface, power efficiency, neural networks

ACM Reference Format:
Di Wu, Jingjie Li, Zhewen Pan, Younghyun Kim, and Joshua San Miguel. 2022. uBrain: A Unary Brain Computer Interface. In *The 49th Annual International Symposium on Computer Architecture (ISCA '22), June 18–22, 2022, New York, NY, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3470496.3527401

1 INTRODUCTION

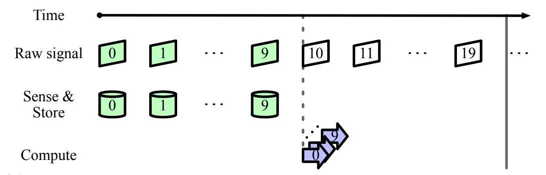
Since the 1920s [79, 83], brain signals have been used to understand creature behaviors, as they exhibit varying brain dynamics, i.e., signal patterns, in both the spatial and temporal dimensions. In the last 30 years, various brain signal modalities [67] have been leveraged to capture, analyze and control human brain activities via brain computer interfaces (BCIs) [1, 47]. In this work, we focus on electroencephalogram (EEG) signals, which are collected at the scalp without clinical surgery and dominate over 70% research in the last decade [22]. Recently, due to the high accuracy in identifying the spatial-temporal dynamics [70, 80, 97], deep neural networks (DNNs) [34] have attracted research interest in the BCI community to substitute classical feature engineering.

Table 1: Comparison of BCI hardware in terms of the task capability and hardware efficiency. L, M and H are short for low, medium and high, respectively. The diversity column additionally contains the supported operations.

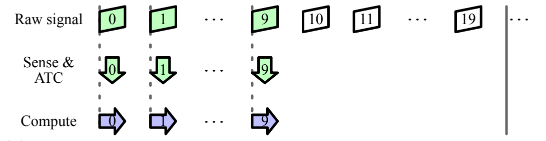
Platform	Task Capability		Hardware Efficiency		
	Accuracy	Diversity	Sense	Store	Compute
CPU [41, 81]	67 ∼ 98% [25, 97]	H (All)	L	L	L
HALO [29]	67 ∼ 85% [97]	H (SVM, FFT, etc.)	L	M	M
SC-SVM [25]	67 ∼ 75% [25]	L (SVM)	L	M	H
uBrain (ours)	91 ∼ 95%	H (CNN, RNN)	H	H	H

State-of-the-art BCI hardware is constructed as three stages, i.e., sense-store-compute, as in Table 1. First, the sense stage utilizes Analog-to-Digital Converters (ADCs) to transform the sampled analog EEG signals obtained by the sensor to digital binary data. Then the store stage uses a digital memory to buffer all the digitized data within a predefined time window. Finally, the compute stage interacts with the digital memory and calculates the results. As most EEG-based BCIs are designed to be lightweight, as well as responsive in real time to the brain activities [22, 67], they usually embed general-purpose CPUs or even GPUs [41, 81, 97], which can flexibly execute both classical feature engineering and emerging DNNs, offering varying accuracy and high task diversity. Meanwhile, in the computer architecture community, there is a trend to optimize the hardware efficiency of BCIs, targeting either the entire system [29, 54] or partial stages [25, 44, 74, 84]. As examples, Karageorgos et al. manufacture a reconfigurable binary computing BCI, HALO, with hardware optimized for diverse tasks [29]. Dedicated functional units are assigned to classical algorithms like Support Vector Machine (SVM) and Fast Fourier Transform (FFT), etc., while a micro-controller is responsible for extended operations with lower efficiency. And Han et al. design SC-SVM to use low-power stochastic computing [19, 91] to implement SVM [25].

Challenges, however, still exist and hinder the implementation of desirable BCIs with both high task capability and hardware efficiency as in Table 1. *First, the task capability in existing BCIs is imbalanced in terms of accuracy and diversity, given a reasonable BCI*



(a) Existing designs sense and store raw signals from multiple timestamps before compute.



(b) The proposed uBrain computes on sensed samples in bitstream format immediately without storing them.

Figure 1: BCI hardware stage. The vertical dashed and solid lines mark the earliest start and the latest end of the compute stage.

hardware budget. The accuracy of widely-adopted classical feature engineering algorithms on HALO and SC-SVM is not comparable to, e.g., on average 5.4% lower than, that of emerging DNNs running on CPUs [70]; however, CPUs exhibit significantly lower hardware efficiency. On the other hand, diverse BCI tasks can be executed on CPUs and HALO, while only a single task can run on hardware efficient SC-SVM. *Second, existing BCI hardware provides suboptimal hardware efficiency, due to lacking immediate signal processing after sensing.* Classical feature engineering on CPUs, HALO and SC-SVM, as well as most emerging DNNs on CPUs, operates on all binary data within a predefined time window, which need to be sensed and stored before compute, introducing a delay, i.e., no overlap, between the sense/store stage and the compute stage in Figure 1a. Such signal processing lagging behind sensing decreases the hardware efficiency at all stages. At the sense stage, existing BCIs require expensive high-resolution ADCs (12 bits or beyond [29, 67]), especially for massive signal channels (256 or 3072 channels in [8, 54]). At the store stage, a large memory is mandatory, with the size proportional to the window size. At the compute stage, the hardware works on a window of data without leveraging the time at the sense/store stage, increasing the running frequency and power.

Our proposal to address the above challenges is uBrain, a unary computing architecture for EEG-based BCIs by co-designing the DNN algorithm and hardware to achieve high task capability and hardware efficiency in Table 1. uBrain distinguishes itself from existing designs in two aspects. *First, uBrain executes emerging DNNs, instead of classical feature engineering, with customized unary operations, and achieves high task accuracy and diversity.* uBrain targets DNN architectures [97] with cascaded convolutional neural networks (CNNs) for spatial features and recurrent neural networks (RNNs) for temporal features, naturally offering higher accuracy than feature engineering. To further guarantee high accuracy, the DNN is optimized with customized unary operations: 1) no data overflow beyond the legal unary data range; 2) every operation is deliberately matched to a highly accurate yet simple unary computing unit. Moreover, uBrain performs multiple BCI tasks using DNNs with the same architecture but different weights. Such an algorithm-based approach requires neither fine-tuned software nor hardware reconfigurability, unlike with a CPU or HALO. *Second,* uBrain *enables simultaneous high efficiency in all hardware stages, due to immediate signal processing after sensing.* uBrain targets to optimize the efficiency for the DNN above, which shows high accuracy in identifying spatial-temporal dynamics, i.e., brain signal patterns. The adopted DNN allows unary bitstreams to flow easily from the sensor to unary computing units without complex interactions [99]. At the sense stage, we propose low-cost Analog-to-Temporal Conversion (ATC) to directly generate temporal-coded unary bitstreams from the sample at each timestamp in Figure 1b. These bitstreams are immediately computed on upon arrival without waiting and storing the entire window, reducing the overhead. At the compute stage, uBrain applies a hybrid unary binary architecture, which pipelines via inter-layer hardware time-division multiplexing (HTDM) and overlaps with the sense/store stage to lower the running frequency. The low frequency leads to reduced dynamic power and enables the use of low-leakage techniques for power efficiency. The hardware efficiency is further boosted by intra-layer HTDM, which reuses the computing units. Also, we design novel unary multipliers to improve the accuracy and efficiency. To summarize, in the context of DNN-based BCIs, uBrain ensures simultaneous high accuracy and efficiency, while prior works offer neither efficient execution, e.g., CPU and HALO, nor basic support, e.g., SC-SVM.

We list the contributions of this work as follows:

- This work identifies that existing BCIs are of task incapability and hardware inefficiency due to lacking immediate signal processing after sensing. We propose a unary computing BCI, uBrain, with algorithm-hardware co-design to overcome above deficiencies.
- uBrain optimizes an emerging DNN with customized unary operations for accurate and efficient execution. It also exhibits high task diversity by performing multiple tasks with the same DNN architecture, i.e., requiring neither fine-tuned software nor hardware reconfigurability.
- uBrain's hybrid unary binary architecture is able to perform immediate signal processing after sensing, with high hardware efficiency due to reduced running frequency, allowed by Analog-to-Temporal Conversion, inter-layer hardware time-division multiplexing (HTDM). The intra-layer HTDM and unary multiplier innovation further boost the efficiency by eliminating the hardware bottleneck in existing unary designs.

The following Section 2 reviews BCI and unary computing. Then Section 3 and Section 4 introduce uBrain's DNN and hardware architectures. Next, Section 5 evaluates the performance. Finally, Section 6 concludes this work.

2 BACKGROUND

In this section, we review the BCI signal modalities and frameworks, as well as unary computing concepts.

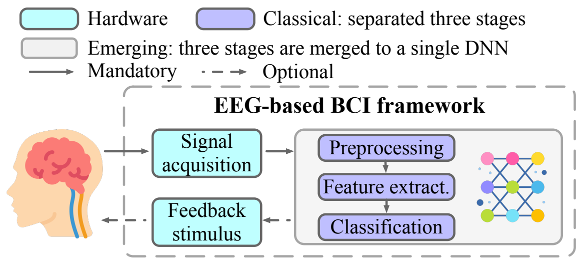


Figure 2: Classical and emerging EEG-based BCI frameworks.

2.1 Brain Computer Interface

BCI is defined as a complete software-hardware system to manipulate brain signals with distinguished modalities to control computers for communication, classification, prediction, control and beyond in real time [14, 22, 51, 67, 73].

2.1.1 BCI Signal Modality. Depending on the electrode location, brain signals have varying modalities [23, 43, 49, 53, 67, 76], e.g., Electroencephalogram (EEG), ECoG and SNA signals, etc., which are collected at scalp, arachnoid or dura and cerebral cortex using non-, semi- and fully-invasive electrodes. More invasiveness imposes better signal quality [39] but growing BCI power restrictions (tens of milliwatts) [29]. Among those, EEG signals attract most research interests [22] with applications in motor imagery [28, 77], seizure prediction [4, 48], emotion recognition [2, 86], sleep monitoring [72, 80], speech synthesis [6, 88], odor memorization [66], etc. In this work, we choose EEG signals to prove uBrain's task diversity using motor imagery and seizure prediction tasks, which are also present in HALO [29].

2.1.2 EEG-Based BCI Framework. Classical BCI frameworks for EEG signals based on feature engineering consist of five stages [5, 15, 36, 65, 67–69] in Figure 2, formulating a closed-loop system. The signal acquisition and feedback stimulus stages involve hardware electrodes, while others utilize either general-purpose software [41, 81] or specialized hardware [29]. Raw signals are sampled by sensors at 10 $\sim$ 5000Hz [87] and later digitized by ADCs with 12 $\sim$ 16 bits [29, 67]. Then the digital data are sliced into windows and denoised in the pre-processing stage. Next, useful features for the target task are extracted from the denoised signals. The following stage performs classification via classical machine learning algorithms to determine the proper action. According to the task purpose, the optional feedback electrodes will send out the expected stimulus back to the subject. Also, adjacent stages require communication if not integrated together. The latency of BCIs ranges from tens of milliseconds [29, 89] to several seconds [4, 10, 18, 31, 37, 64, 88] according to the task. However, human reaction and movement time[1] to events is usually between 200 $\sim$ 400ms [11, 27, 33]. Therefore, we bound the BCI latency budget to 250ms in this work.

[1]The reaction and movement time are the elapsed time from the event occurrence to the event realization and from the event realization to the event reaction for a subject [11, 27, 33], respectively.

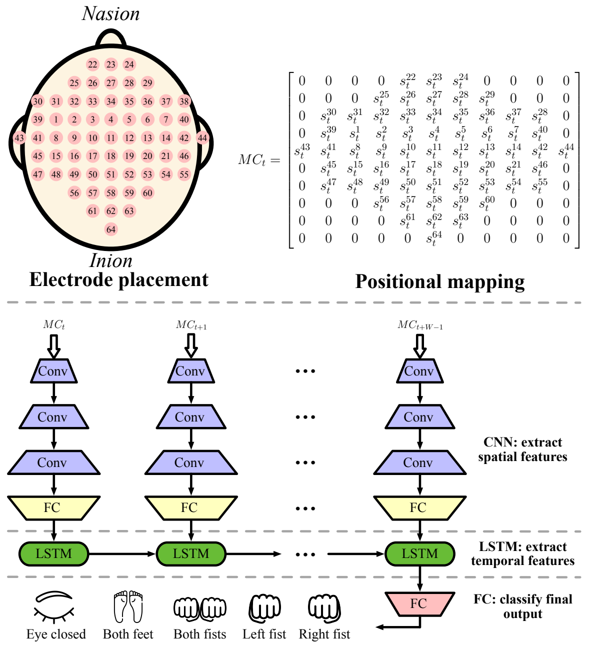


Figure 3: DNN with a cascaded CNN-LSTM architecture to classify motor imagery based on EEG signals. Positional mapping maintains the relative position among electrodes in the mesh clip at timestamp t (MC_t).

As EEG signals vary according to location, and also fluctuate through time, both spatial and temporal features exist in EEG signals, for which DNNs like convolutional neural networks (CNNs) [20, 36, 40] and recurrent neural networks (RNNs) [6, 42, 56, 80] are leveraged, achieving 5.4% accuracy boost on average [70]. Besides the accuracy benefit, DNNs can further merge multiple stages in classical frameworks, including pre-processing, feature extraction and classification, into a single stage in Figure 2, lowering the complexity.

Zhang et al. [97] introduce a DNN architecture in Figure 3 with cascaded CNN and Long Short-Term Memory (LSTM) [26] to simultaneously learn the spatial and temporal features for motor imagery, i.e., classifying which motor is imagined in the brain. The brain signals are collected at 128Hz by 64 electrodes spreading out the scalp. Signals at timestamp t are organized as a mesh clip, MC_t, where the relative location among the electrodes is preserved with positional mapping to extract the spatial features with matrix convolution (Conv) and fully connected matrix multiplication (FC). Next, the cascaded LSTM layers extract the temporal features from the spatial features at total W timestamps. The spatial features at different timestamps can be extracted in parallel, while the adjacent temporal feature extraction is sequential. At the output, a head, i.e., a FC layer, is applied for final classification among five categories. In this work, uBrain uses a similar but lightweight DNN, compared to the original design from Zhang et al. [97]. The DNN is optimized with unary computing specific operations for hardware accuracy and efficiency, and trained on separate datasets for different tasks for high task diversity. As DNNs have been widely used in various

Rate coding (random bits)	A	1100101010100110 (P=8/16, V_{uni}=0.5, V_{bi}=0.0)
Temporal coding (deterministic bits)	B	0000000011111111 (P=8/16, V_{uni}=0.5, V_{bi}=0.0)

Figure 4: Unary bitstreams with varying codings.

human signals [16], it is possible to extend uBrain to non-EEG signals with proper training, which is beyond the scope of this work.

One appeal in DNN-based BCIs is in-situ training for optimal personalization. However, Zhang et al. [97] showcase that their human test accuracy surpasses best classical accuracy by 8%, i.e., non-optimal yet accurate enough for personal use. As such, we leave in-situ training to future work.

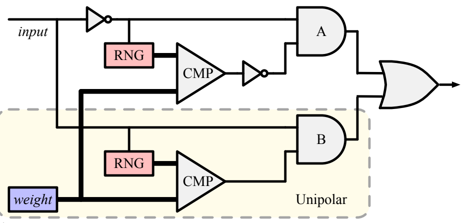
Another important aspect of BCIs is power consumption. Commercial BCI systems [24, 58, 59] can have around 200mW total power [58]. They spend 50 ∼ 100mW on signal acquisition, preprocessing and/or communication, but support no compute. uBrain is envisioned as the compute engine in HALO-like BCI systems, covering the signal acquisition to classification stages. uBrain imposes no changes to the interfaces of the communication and feedback stimulus modules and can be plugged into commercial BCI systems with minimal added complexity, classification error and power overhead (around 30mW as evaluated in Section 5.5). On-chip compute in uBrain further saves power in communication.

2.2 Unary Computing

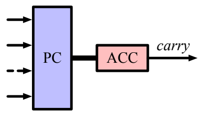
Unary computing works on serial bitstreams with extremely simple logic [91]. The bitstreams can be either rate- or temporal-coded in stochastic [19] or temporal computing [45], with applications in image processing [3], low-density parity-check [82, 90], DNN [50, 75], DNA sequencing [45], etc.

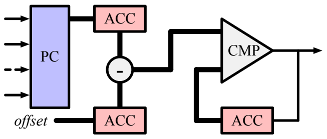
2.2.1 Data Representation. The rate- and temporal-coded bitstreams relate the data value to the ratio of bit 1s, but are featured with different bit distributions in Figure 4, i.e., random and deterministic distributions, respectively. The deterministic distribution requires all bit 1s ahead of all bit 0s, or vice versa. The bitstream value depends on not only the ratio of bit 1s, but also the polarity. The unipolar and bipolar bitstreams refer to unsigned and sign data with the legal value range of $[0, 1]$ and $[-1, 1]$, respectively. Given N-bit binary value P as the ratio of bit 1s, unipolar and bipolar bitstreams of length 2^N are valued $V_{uni} = P$ and $V_{bi} = 2P - 1$. uBrain applies both codings for high task accuracy and hardware efficiency.

2.2.2 Analog-to-Stochastic Conversion. Conventionally, rate- or temporal-coded bitstreams are generated by comparing the buffered source data with a random number generator or counter output in the digital domain in Figure 5a. However, the digital units to generate unary bitstreams are more expensive compared to the analog counterparts. There exist explorations into the low-cost conversion of analog signals into rate-coded bitstreams for image processing applications [3, 17, 30, 38, 63], i.e., analog-to-stochastic conversion. Onizawa et al. leverage Magnetic-Tunnel Junction (MTJ) devices [63], whose state switching is probabilistic. With proper configurations, a rate-coded bitstream with its value linear to the analog input amplitude can be generated, eliminating the need for



(a) Static multiplier. The RNG for A is only updated by input bit 0 continuously, and such conditional weight bitstream generation also applies to the RNG for B.





(b) Scaled adder. **(c) Non-scaled adder.**

Figure 5: Bipolar unary computing units. Thin and thick lines represent unary and binary connections. RNG: random number generator; CMP: comparator; PC: parallel counter; ACC: accumulator. Logic to generate input bitstreams is not shown.

digital hardware. Khatamifard et al. further connect the MTJ devices in the design of Onizawa et al. [63] with the analog memory to produce rate-coded bitstreams for image processing [30]. On the contrary, these works [3, 17, 38] leverage analog comparators for bitstream generation. The sensed raw data is compared to an analog reference signal, which can be produced by either feeding a digital RNG output to a Digital-Analog Converter [3] or a linear ramp generator [17].

In this work, instead of leveraging rate-coded bitstreams from analog-to-stochastic conversion, we propose to generate temporal-coded bitstreams in unary computing with the help of Analog-to-Temporal Conversion (ATC) and achieve immediate signal processing after sensing. The temporal-coded bitstreams from ATC are directly consumed by the following compute units, overlapping the sense and compute stages. Such a behaviour is similar to the voltage-to-time conversion in time-based ADCs [52, 57, 96], which simply accumulates the bitstreams towards the final voltage with 4-11 bits [57] and exhibits orders of power reduction for data resolution higher than 8 bits [57]. The interaction between the sensor and the computing units is also simpler in uBrain via directly streaming than in a CPU via acknowledgement protocols [99].

2.2.3 Unary Multiplier and Adder. Working on bitstreams, unary computing units like the multiplier and adder can be extremely simple. Figure 5 presents the bipolar multiplier and adders to compose a highly accurate unary computing architecture for general matrix multiply (GEMM) in uGEMM [91, 92], which is a fully streaming design [93], requiring no interconversion between unary bitstreams and binary data before the final output. The static multiplier, with the weight statically buffered, accepts an arbitrary input bitstream and a rate-coded weight bitstream. AND gate A and B with the corresponding CMP and RNG act as unipolar multipliers and are responsible for input bit 0 and 1. The RNGs for A and B are separately updated by the input bit 0 and 1 continuously for high accuracy,

Table 2: uBrain DNN configuration.

Layer		Shape		Activation
		Input	Output	
CNN	Conv1	$(10, 1, 10, 11)$	$(10, 16, 10, 11)$	*HardReLU*
	Conv2	$(10, 16, 10, 11)$	$(10, 32, 10, 11)$	*HardReLU*
	FC3	$(10, 32 * 10 * 11)$	$(10, 256)$	*HardReLU*
RNN	MGU4	$(10, 256)$	$(10, 64)$	*HardSigmoid* *HardTanh*
Head	FC5	64	5	*HardTanh*
	FC6	64	2	*HardTanh*

i.e., the weight applies conditional bitstream generation [91]. Also, RNGs can be shared by weights with an identical input, as those RNGs are always synchronously updated by that input. On the other hand, the bipolar adders accept rate-coded bitstreams for accuracy. The scaled adder averages the input by first summing all input bits, and then accumulating the sum, whose carry bit is the output bit. The non-scaled adder calculates the sum of inputs by first summing all inputs, and then clipping the sum when the sum exceeds the legal bipolar data range. In this work, above designs will be uBrain subunits. In addition, we design novel multipliers for both rate and temporal coding for high task accuracy and hardware efficiency, by eliminating the hardware bottleneck to generate bitstreams from the costly RNG and CMP.

3 CUSTOMIZED DNN FOR UNARY COMPUTING

uBrain adopts the DNN in Table 2, which is similar to that in Figure 3 but lightweight: 1) it works on 10 samples as a window, and 2) it contains a CNN and RNN for spatial and temporal features, and two classification heads, one each per task. It is optimized with customized unary operations for high accuracy: 1) DNN data are constrained to $[-1, 1]$ to avoid compute overflow; 2) all operations match highly accurate yet simple unary computing units. The weight constraint is achieved using weight decay [35] during training, while the layer input/output constraint is illustrated as follows.

3.1 Convolutional Neural Network for Spatial Features

The CNN includes two Conv layers and one FC layer in Table 2. Conv layers have the kernel size of $(3, 3)$ and the stride size of 1. The shape parameters in $(10, c, 10, 11)$ from left to right represent that 1) the window size is 10, 2) the channel number is c, 3) the mesh clip size is 10-by-11 in Figure 3. For FC layers, the first parameter 10 is the window size, and the second is the input/output size. The activation function is *HardReLU*, instead of *ReLU* [55] in Equation 1. The *HardReLU* clips the output at 1, constraining it to $[-1, 1]$.

$$\begin{aligned} ReLU(x) &= \max(0, x) \\ HardReLU(x) &= \max(0, \min(1, x)) \end{aligned} \tag{1}$$

3.2 Recurrent Neural Network for Temporal Features

The RNN applies Minimal Gated Unit (MGU), which has only half the weights yet similar accuracy [98] to LSTM in Figure 3. The original MGU is formulated in Equation 2, where t is the timestamp, x and h are the input and output vectors, f is the forget gate vector, n is the new gate vector, and w is the weight matrix. $*$ is matrix multiplication, $\odot$ is elementwise multiplication, and $[,]$ is vector concatenation.

$$\begin{aligned} f_t &= Sigmoid(w_f * [h_{t-1}, x_t]) \\ n_t &= Tanh(w_n * [f_t \odot h_{t-1}, x_t]) \\ h_t &= (1 - f_t) \odot n_t + f_t \odot h_{t-1} \end{aligned} \tag{2}$$

However, this standard MGU yields neither high unary accuracy nor hardware area and power efficiency. We first replace *Sigmoid* and *Tanh* with *HardSigmoid* and *HardTanh*, which are initially proposed for binary neural networks with data limited to -1 and $+1$ [13]. *HardSigmoid* and *HardTanh* are formulated in Equation 3. They approximate *Sigmoid* and *Tanh* with piecewise linear functions and significantly simplify the unary hardware design. More specifically, *HardSigmoid* can be implemented with a two-input scaled adder in Figure 5b with high accuracy [91]. On the other hand, *HardTanh* means to directly forward the bitstream. For example, if multiple input bitstreams are added with the non-scaled unary adder in Figure 5c, *HardTanh* naturally exists at the output.

$$\begin{aligned} HardSigmoid(x) &= \max(0, \min(1, (x + 1)/2)) \\ HardTanh(x) &= \max(-1, \min(1, x)) \end{aligned} \tag{3}$$

Our MGU architecture with customized unary operations is shown in Equation 4, where all additions, i.e., the accumulation in matrix multiplication and conventional addition, are non-scaled additions, which naturally invoke *HardTanh*.

$$\begin{aligned} f_t &= HardSigmoid(HardTanh(w_f * [h_{t-1}, x_t])) \\ n_t &= HardTanh(w_n * [f_t \odot h_{t-1}, x_t]) \\ h_t &= HardTanh((1 - f_t) \odot n_t + f_t \odot h_{t-1}) \end{aligned} \tag{4}$$

3.3 Head for Multiple Tasks

To reuse the same DNN architecture for different tasks, i.e., motor imagery and seizure prediction, we assign one head to each task. The head input, of size 64, is the MGU output in the last timestamp. On the other hand, the head output is of size 5 for motor imagery and 2 for seizure prediction. These head layers, i.e., FC5 and FC6, apply the non-scaled adder for accumulation, equivalent to appending *HardTanh*.

The above operations are low cost yet yield high accuracy in unary computing. Together with the multiplier innovation in the next section, all uBrain operations match high-quality, i.e., accurate and efficient, unary computing units.

4 UBRAIN ARCHITECTURE

Our uBrain architecture, which operates on bipolar unary bitstreams, proposes fundamental improvements in Analog-to-Temporal Conversion, inter-layer and intra-layer hardware time-division multiplexing (HTDM) and unary multipliers. With these innovations,

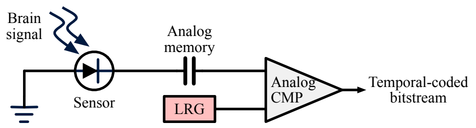
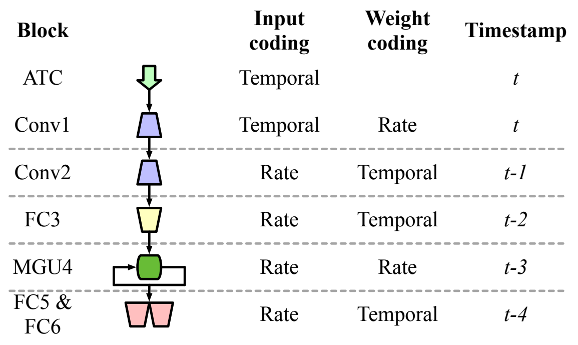


Figure 6: Analog-to-temporal conversion built on analog memory and comparator (CMP), and linear ramp generator (LRG).



Block	Input coding	Weight coding	Timestamp
ATC	Temporal		*t*
Conv1	Temporal	Rate	*t*
Conv2	Rate	Temporal	*t-1*
FC3	Rate	Temporal	*t-2*
MGU4	Rate	Rate	*t-3*
FC5 & FC6	Rate	Temporal	*t-4*

Figure 7: Hybrid unary binary architecture in uBrain. The dashed lines separate the hardware into blocks. Each block is standalone hardware, corresponding to one layer in Table 2 and having binary output calculated by internal unary computing units. Each arrow, except the one from ATC to Conv1, represents a double buffer connecting adjacent hardware blocks.

uBrain achieves immediate signal processing after sensing for high hardware efficiency.

4.1 Analog-to-Temporal Conversion

The proposed low-cost Analog-to-Temporal Conversion (ATC), which is based on simple off-the-shelf subunits, is shown in Figure 6. The sensor samples the brain signals. At each timestamp, one sample is stored and compared to a ramp signal to obtain a temporal-coded bitstream in Figure 4, which offers better accuracy control than a rate-coded bitstream in existing designs [3, 17, 30, 38, 63], due to less randomness. These bitstreams are then directly consumed by the Conv1 layer without costly digital buffers. Moreover, ATC naturally allows Conv1 to utilize our proposed static multiplier for temporal coding so as to save more hardware.

4.2 Inter-Layer Hardware Time-Division Multiplexing

Unlike prior fully streaming unary computing architectures [91], uBrain is a hybrid unary binary architecture [93] (shown in Figure 7) where interconversion between unary bitstreams and binary data is mandatory before the final output. uBrain separates the hardware into blocks with equal runtime, bounded by the inverse of the input sampling rate. Such separation using double buffering enables both easy accuracy control and hardware pipelining, which further allows layerwise bitstream coding and inter-layer HTDM. As in

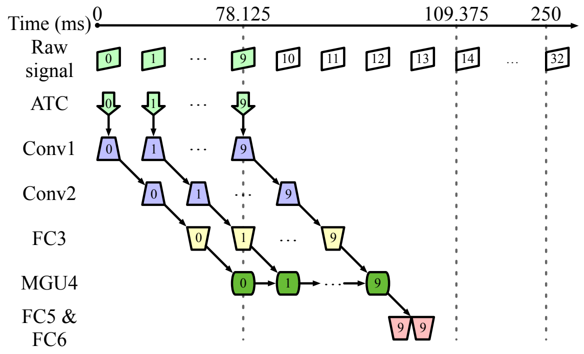


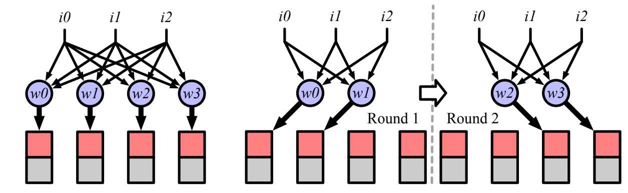
Figure 8: uBrain spatial-temporal dataflow with inter-layer hardware time-division multiplexing. Each number is a timestamp. Data from each timestamp proceed through the hardware, except that FC5 and FC6 compute on last timestamp results. In other words, a single layer maps to only one hardware block.

Figure 7, uBrain uses temporal coding for Conv1 inputs and all weights in layers other than Conv1 and MGU4. Weights, whose amount is higher than input, applying temporal coding reduces the hardware cost as temporal coding using counters is generally cheaper than rate coding using RNGs. Then CNN blocks always pass the output to the next block, while the RNN (MGU4) block feeds the output to both itself and two head blocks, i.e., FC5 and FC6. At a timestamp t, the data timestamps of different hardware blocks are labelled in Figure 7. In total, data from 5 consecutive timestamps can co-exist simultaneously. uBrain's spatial-temporal dataflow is presented in Figure 8. Each block labelled with the timestamp is multiplexed at each timestamp, i.e., inter-layer HTDM. With input sampling frequency f_s = 128Hz and window size W = 10, total $T = 14/128 * 1000 = 109.375$ms is required for computation, less than half of the 250ms time budget in Section 2.1.2. Overlapping the sense/store and compute stages drops the running frequency, decreases the dynamic power and enables low-leakage techniques for power efficiency.

4.3 Conv and FC Layers

All Conv and FC layers in the CNN and heads are GEMM operations followed by hard activations, i.e., *HardReLU* or *HardTanh*, and consist of both rate and temporal coding for unary computing. The input and weight bitstreams participate in unary computing and are finally converted back to binary outputs, followed by the correspondent activation functions. Examples for those layers are drawn in Figure 9.

4.3.1 Static Multiplier for Temporal Coding. The inputs in all layers are rate-coded by comparing the input and the RNG output, except those in the Conv1 layer, which are temporal-coded bitstreams from ATC. The proposed static multiplier for temporal coding, which has a statically buffered source data applying temporal coding, is shown in Figure 10. This design implicitly forces conditional bitstream generation [91]. More specifically, in Figure 5a, continuous logic 1s first update the bottom RNG with the top RNG disabled, then



(a) No intra-layer HTDM.

(b) Intra-layer HTDM.

Figure 9: Conv and FC architecture without and with intra-layer hardware time-division multiplexing (HTDM). Thin and thick lines represent unary and binary connections. (a) Disabling intra-layer HTDM means an equal number of unary computing units in circle and double buffered outputs in rectangle; (b) intra-layer HTDM by halving the computing units once needs two rounds of compute, with each for half outputs. Intra-layer HTDM increases the running frequency to maintain runtime.

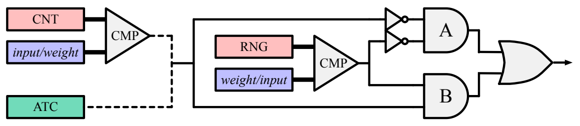


Figure 10: Static multiplier for temporal coding. By hybridizing rate and temporal coding, conditional bitstream generation is enforced implicitly at lower cost than that in Figure 5a.

continuous logic 0s update the top RNG with the bottom RNG disabled. This is is equivalent to one RNG used first by logic 1s and then by logic 0s with an implicit always-on enable, which is exactly the case in Figure 10. As such, our design offers identical accuracy and half cost compared to that in Figure 5a. Considering the large portion of multiplication in DNNs, this design significantly reduces the total hardware area and power. Moreover, a single RNG and CNT can be aggressively shared by all multipliers, further reducing the hardware ratio of RNG and CNT to almost zero, i.e., eliminating the hardware bottleneck for bitstream generation in existing unary designs [30, 91, 93]. This is achieved by that 1) for all rate-coded bitstreams, given identical initial RNG states, continuously updated RNGs, i.e., with an implicit always-on enable, always have identical outputs, and 2) all temporal-coded bitstreams are generated using counters, which are always identical. Note that such hybrid coding for static binary data is symmetric, i.e., the coding for inputs and weights can be exchanged with each other without accuracy drop. Considering practical hardware implementations with a limited fanout, the less expensive temporal coding favors whichever with a higher quantity for more savings. As such, we assign temporal coding to weights, except those in Conv1.

4.3.2 Binary Accumulation. The unary products are accumulated to binary data using the parallel counter logic followed by an accumulator in Figure 5c. Then the binary outputs are processed with the activation functions. With double buffering, the computation of adjacent layers can be pipelined and overlapped. For example, in Figure 9a, the red buffers collect the binary results from the current layer, while the gray buffers generate rate-coded input bitstreams for the next layer.

4.3.3 Intra-Layer Hardware Time-Division Multiplexing. As each output operates on the same inputs but different weights, a fully parallel design in Figure 9a is a SIMD architecture via broadcasting, and the SIMD degree is the output count. Therefore, HTDM allows trade-offs among area, frequency and power. For example, in Figure 9a, there are 4 outputs calculated from identical inputs. Given target input sampling frequency f_s, the maximum runtime of each layer block is $t_b = 1/f_s$. With data resolution of n_b, the bitstream length is $l_b = 2^{n_b}$. When disabling intra-layer HDTM, all outputs are simultaneously calculated, and the running frequency of this layer block is $f_b = f_s * l_b = f_s * 2^{n_b}$. If we halve the number of computing units once in Figure 9b, $H_b = 1$, instead of $H_b = 0$ in Figure 9a, the running frequency is $f_b = f_s * 2^{n_b+1}$. In general, halving the hardware H_b times leads to the running frequency $f_b = f_s * 2^{n_b+H_b}$. Such a trade-off offers more opportunities to meet the power budget. Moreover, scheduling the halved computing units according to the output is done with a state machine, significantly cheaper than the micro-controller used in existing designs [29].

uBrain applies varying levels of intra-layer HTDM to different layers, based on their weight count. The Conv1 and Conv2 layers have much less weights than the computing units, and aggressive intra-layer HTDM is applied to the computing units, with all weights stored on-chip. The FC3 layer with total 256 outputs has over 0.9M weights, and the computing units must be multiplexed to reduce the cost. We halve the computing units $\log_2 256 = 8$ times and compute only 1 output at a time. Additionally, all weights are stored in an off-chip DRAM, and double buffering is applied to the weights to hidden the latency. uBrain requires much lower DRAM bandwidth than existing designs, exhibiting higher efficiency at the store stage in spite of this additional DRAM to store the DNN model. For the final head layers, i.e., FC5 and FC6, due to their small size, no intra-layer HTDM is employed.

4.4 MGU Layer

The MGU4 layer in RNN applies rate coding for all inputs and weights, leading to a fully streaming, thus fully parallel, architecture with no interconversion between unary bitstreams and binary data before the final output, as shown in Figure 11. The rate-coded input bitstream x_t is generated from the double buffer in the previous FC3 layer. The concatenation of x_t and other bitstreams is in unary domain and requires no hardware. The two matrix multiplications (MatMul) are both implemented with bipolar static multiplier in Figure 5a with the non-scaled adder in Figure 5c for accumulation, so that each MatMul naturally has a *HardTanh* followed. Then a *HardSigmoid*, implemented as the scaled adder in Figure 5b, generates f_t after MatMul-w_f. The final non-scaled adder also has a subsequent, yet inherent, *HardTanh*. The input to this adder is separated into three parts. The first part is n_t, the output of MatMul-w_n. The second part is $f_t \odot h_{t-1}$, generated by the static multiplier from Figure 5a, as now the h_{t-1} acts as the statically buffered weight. This product is then concatenated with x_t as the input to MatMul-w_n. The third part is $-f_t \odot n_t$, which is implemented by inverting the bipolar elementwise product of $f_t \odot n_t$ calculated in the proposed in-stream multiplier painted with green.

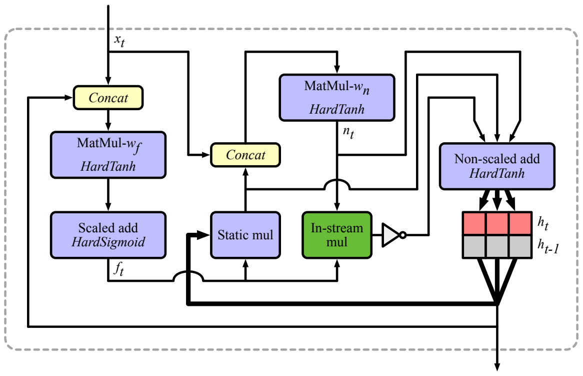


Figure 11: MGU4 architecture in Equation 4. Rounded rectangles are unary computing units, while rectangles are binary memory units with double buffering. Thin and thick lines represent unary and binary connections, respectively. The concatenation (*Concat*) in unary domain requires no hardware. MatMul-w_f and -w_n are two matrix multiplications for weight w_f and w_n.

Here the in-stream [94, 95] multiplier refers to that both input and output are arbitrarily rate-coded bitstreams with arbitrary correlation. Eventually, the sum are converted back to the binary domain as h_t. The h_{t-1} is double buffered and converted to the unary domain, acting as both the output to the heads, i.e., FC5 and FC6, and the feedback to the MGU4 itself.

In this architecture, the fully streaming behavior is enabled by double buffering, while the accuracy is determined by the computing units. Though the static multiplier, scaled and non-scaled adders from [91] are proved to be highly accurate, the elementwise in-stream multiplier in green, which takes arbitrarily rate-coded input bitstreams may not offer high accuracy, if implemented naively. Here, the two bitstreams are f_t from *HardSigmoid* and n_t from MatMul-w_n, and no explicit conditional bitstream generation in Figure 5a can be enforced for high accuracy. To address this, we propose an in-stream multiplier with approximate conditional bitstream generation in Figure 12. This in-stream multiplier differs from the static multiplier in that the static multiplier has a statically buffered binary data, while our in-stream multiplier has an approximated binary data using a shift register and a parallel counter. One of the two input bitstreams remains unchanged, e.g., *input 0*, while the other input bitstream, e.g., *input 1*, is fed to the shift register, which keeps a record of the recent history. Then the parallel counter accumulates the bits in the shift register and uses the sum as an approximate value of the expected binary data. Based on the approximate value, a new rate-coded bitstream for *input* 1 is generated conditionally based on *input* 0.

5 EVALUATION

In this section, uBrain is evaluated against existing designs based on CPU or accelerators, with respect to application accuracy, hardware area, frequency, latency, power and energy.

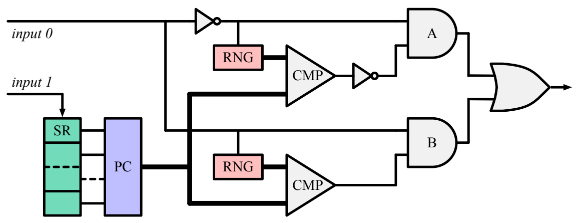


Figure 12: In-stream multiplier for rate coding. The shift register (SR) and parallel counter (PC) approximate one input to achieve approximate conditional bitstream generation.

5.1 Experimental Setup

5.1.1 DNN Training and Inference. We use dataset [21, 71] for motor imagery and [78] for seizure prediction, with sampling frequencies of 128Hz and 8Hz. The motor imagery dataset contains five categories, i.e., eye closed, both feet, both fists, left fist and right fist, with 468024 training and 156235 testing samples. The seizure prediction dataset contains two categories, i.e., onset or no onset, with 117265 training and 39263 testing samples. All samples have an equally sized window, including the mesh clips from 10 consecutive timestamps. The training and inference with post-training quantization are done using an open-source unary computing simulator, UnarySim [91, 92]. We use cross entropy loss and Adam optimizer with 0.00005 weight decay and 0.001 initial learning rate. The learning rate scheduler is CosineAnnealingWarmRestarts with 50 restart epochs out of 900 total epochs.

5.1.2 Evaluation Methodology. Given highly diverse BCI platforms, the evaluation requires fairness with respect to functionality, latency, and power, etc. We ensure that 1) all BCIs cover identical functionalities from signal acquisition to classification, 2) all BCIs reach a similar level of accuracy, even with sensor errors considered, 3) all BCIs meet the real-time requirement, indicated by the human reaction and movement time, and 4) all BCIs run at the frequency constrained by the real-time requirement under an identical technology node for each stage to avoid power consumption more-than-necessary. However, we believe that fairness is still a long-term goal for architecting BCIs, due to 1) inaccurate error modelling, and 2) inaccurate performance modelling, etc.

5.1.3 BCI Hardware. We evaluate the hardware sense, store and compute stages of uBrain against three baselines, including CPU, systolic array and stochastic computing BCIs. Among all stages, the sense stage results of all designs are analytically modelled from existing literature and scaled to 65nm technology. The sense stages in three baselines are based on neural ADCs [32], while the subunits in uBrain ATC are taken from prior works [30, 46, 85]. The results for the store and compute stages in the CPU and systolic baselines are reported by analyzing the DNN execution results with built-in tools [61] and by an open-source systolic array simulator [93], while those in the stochastic baseline and uBrain are both obtained by synthesizing the RTL using Synopsys Design Compiler with TSMC 32nm technology. The synthesis frequencies are as follows: 1) the frequency of the systolic baseline is 400MHz; 2) the frequency of FC3 in the stochastic baseline and uBrain is 33.6MHz. 3) the frequency of all rest layers in the stochastic baseline and uBrain is 4.2MHz. Those are selected to meet the latency requirement: 1) the systolic baseline does not process signals right after sensing and has significantly high frequency; 2) the stochastic baseline and uBrain both immediately process signals after sensing to utilize low frequency and low-leakage techniques, including high threshold voltage and low supply voltage, which are not applicable to the systolic array at 400MHz. The DRAM results in non-CPU designs are reported by CACTI [9] at 22nm technology.

CPU Baseline. The CPU baseline is the NVIDIA Jetson Nano Developer Kit [60], which has a quad-core ARM A57 [7] with a maximum frequency of 1479MHz and a 4GB LPDDR4 [62], as well as a GPU. The CPU and DRAM area numbers are scaled to 32nm and 22nm technology for fair comparison. And we enable only one out of four cores in ARM A57 and exclude the GPU to retrieve the power consumption without technology scaling. The real-time requirement is $T \leq W/f_s$, where T is the total runtime of the computing stage, W and f_s are the total timestamps in a window and the sampling frequency at the sense stage. The CPU running frequency is tuned to satisfy the real-time requirement, i.e., the computation for the current W timestamps is always completed before the data collection of the next W timestamps is done. For example, in Figure 1a, the computation of the first 10 timestamps starts at the tenth timestamp and must end before the twentieth timestamp, as the data from the next window are ready.

Systolic Baseline. The systolic baseline is a binary computing systolic array for edge computing from [12, 93]. It has 12-row-by-14-column of processing elements, 192KB on-chip SRAM and a 16MB off-chip DDR3 with 22nm technology. Its running frequency is also tuned to meet the real-time requirement in a similar manner to the CPU baseline.

Stochastic Baseline. This baseline is a stochastic computing version of uBrain, already benefiting from customized unary operations and immediate signal processing after sensing. All data apply rate coding and all multipliers follow that from uGEMM [91] in Figure 5a, identical to uBrain. However, as there exists no prior in-stream multiplier for rate coding, i.e., accurate unary multiplier with arbitrary input correlation, the stochastic baseline needs unary-binary interconversion followed by static multipliers to replace the in-stream multiplier in MGU4. The layerwise real-time requirement is given by Equation 5, where t_b, f_b, l_b, n_b and H_b are per block runtime, frequency, bitstream length, binary data resolution and the count of logic halving. And the resultant total runtime is $T = \sum t_b$.

$$t_b = l_b * 2^{H_b}/f_b = 2^{n_b} * 2^{H_b}/f_b = 2^{n_b+H_b}/f_b = 1/f_s \qquad (5)$$

Note for the MGU4 layer with data interconversion, its actual l_b, i.e., computing cycles, doubles compared to that in the uBrain MGU4 layer, thus doubling the running frequency compared to that in uBrain. Both inter- and intra-layer HTDM are applied to CNN layers. The Conv1 and Conv2 layers traverse all possible intra-layer HTDM to find different implementations. The FC3 layer is halved 8 times to improve on-chip hardware efficiency; otherwise, an extra on-chip SRAM of size 1.25MB is necessary to store the weight, introducing about 4.5mm^2 area and 340mW power overheads. And this holds for both the stochastic baseline and uBrain. We choose three different implementations for comparison, namely SC, SC-A and SC-P. SC is an implementation with no intra-layer HTDM for both Conv1 and Conv2, which exhibits the largest area and power. Then SC-A has the minimum area, i.e., the most aggressive intra-layer HTDM for both Conv1 and Conv2. This implementation always computes a single output at a time for both Conv1 and Conv2. Finally, SC-P exhibits the minimum power by individually selecting the configurations for Conv1 and Conv2 for minimized power. The configurations in SC-P do not necessarily correspond to those in SC-A, as though the SC-A has a smaller area, thus smaller leakage power, the running frequency increases to raise the dynamic power. Though the configuration differs from each other, those implementations have identical accuracy.

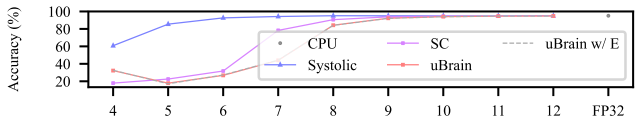
uBrain. For the proposed uBrain architecture, we also have three different implementations with identical accuracy, i.e., uBrain, uBrain-A and uBrain-P. Similar to the stochastic baseline, uBrain has no intra-layer HTDM in Conv1 and Conv2, while uBrain-A and uBrain-P have the best area and power by varying intra-layer HTDM. The layerwise real-time requirement is also given by Equation 5.

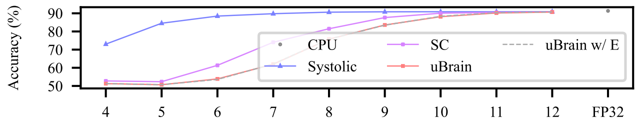
5.2 Accuracy

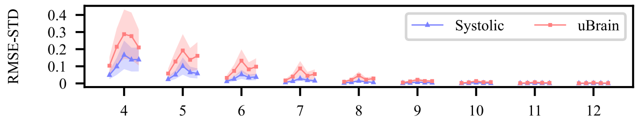
We examine both the overall classification accuracy for both motor imagery and seizure prediction and the layerwise numerical accuracy according to the layer type in Figure 13.

5.2.1 Task Accuracy. The trained DNNs for motor imagery and seizure prediction only have 0.002% and 0.001% weights beyond $[-1, 1]$ due to weight decay, achieving 95.1% and 91.3% FP32 accuracy, respectively, with the corresponding accuracy comparison shown in Figure 13a and Figure 13b. We observe that higher data resolution, i.e., larger bitwidth and bitstream length, yields higher accuracy for both tasks. When the binary data resolution is $N = 10$, the inference accuracy drop of uBrain, compared to FP32 for motor imagery and seizure prediction, is about 1.0% and 3.2%, demonstrating that uBrain provides high task capability in terms of both accuracy and diversity. Note that training the DNN will even mitigate such accuracy drop. Additionally, to model the inaccuracy of non-conventional ATC, we deliberately inject $0 \sim 5\%$ random uniform errors to each input bitstream, i.e., reaching down to 4.3-bit resolution, even though there exists extensive research on ATC-like circuits [52, 57] with up to 11-bit resolution [57]. Overlapped accuracy in Figure 13, i.e., no accuracy drop, indicates that uBrain is robust enough under modelled errors.

5.2.2 Accuracy of Conv and FC Layers. Figure 13c draws the accuracy of different layers in CNN and heads for the systolic baseline and uBrain. The SC result is ignored, as the stochastic baseline has identical accuracy to uBrain for those layers. The input to each layer is randomly generated but identical for different designs, and the resultant error is averaged across multiple runs. It is observed that uBrain layers have similar accuracy to the others at high data resolution. The systolic results show lower errors than uBrain, due to 1) the deterministic computation and 2) higher output resolution [93], i.e., two N-bit inputs produce a $2N$-bit product. Furthermore, as the bitstream length grows, the error of those layers gradually decreases, though the decrease rate becomes more trivial with a larger bitstream length.


(a) Classification accuracy of motor imagery task.


(b) Classification accuracy of seizure prediction task.


(c) RMSE and STD of the Conv and FC layers. For each curve, the data points from left to right represent Conv1, Conv2, FC3, FC5 and FC6.

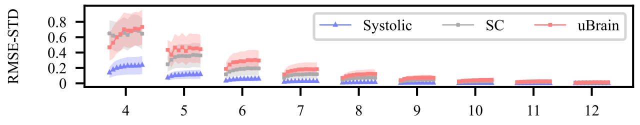
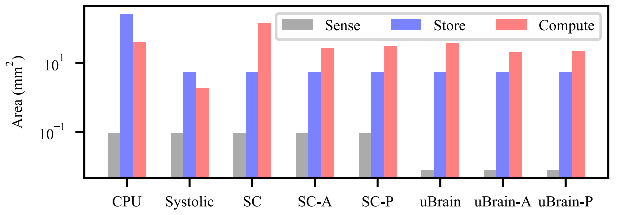

(d) RMSE and STD of the MGU4 layer. For each curve, the data points from left to right represent the output RMSE from timestamp 0 to 9.

Figure 13: Accuracy comparison. The accuracy of 32-bit floating-point CPU, fixed-point systolic array, stochastic computing baselines and uBrain are denoted as FP32, Systolic, SC and uBrain, respectively. The horizontal axis refers to binary data resolution. The binary data resolution, N, represents N-bit binary data for the systolic baseline and 2^N-bit unary bitstream for the systolic baseline and uBrain. The vertical axis refers to classification accuracy in (a) and (b), and root mean square error (RMSE) compared to FP32 in (c) and (d). The dashed lines in (a) and (b) are uBrain results with injected errors. The shadow in (c) and (d) is error standard deviation (STD).


(a) Total BCI area breakdown in log scale. All baselines have the same sensing ADC, and non-CPU designs have the same off-chip DRAM.

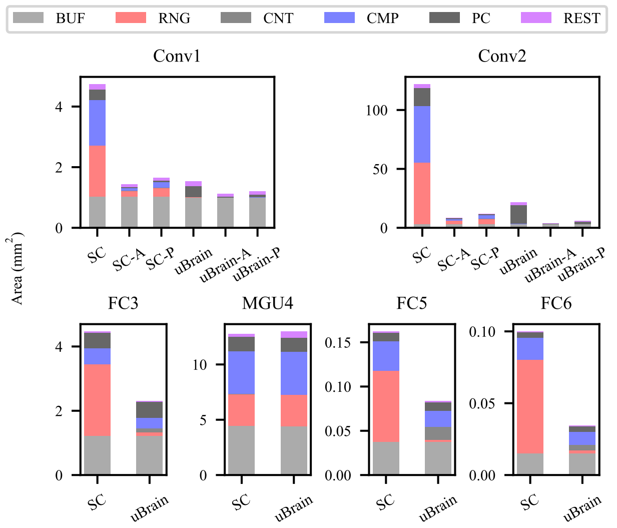

(b) Layerwise area of the stochastic baseline and uBrain. Decomposed subunits include input and weight buffer (BUF), random number generator (RNG), counter (CNT), comparator (CMP) and parallel counter for accumulation (PC), while the rest logic contains activation function units, AND and XNOR gates in unary multipliers, multiplexers and demultiplexers for intra-layer HTDM, etc. As FC3, MGU4, FC5 and FC6 have fixed intra-layer HTDM, only the SC and uBrain results for those are shown.

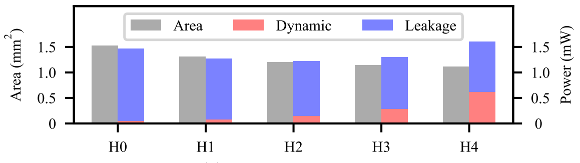
Figure 14: Area comparison.

5.2.3 Accuracy of MGU Layer. Due to the iterative computation in the MGU4 layer, i.e., current output is fed back to itself as input at the next timestamp, we examine the output accuracy after each iteration from total 10 iterations, corresponding to the window size of 10 in the adopted DNN. The accuracy results are shown in Figure 13d. With more iterations, more errors are accumulated in all designs. For this layer, uBrain has worse accuracy than both systolic and stochastic baselines, as the uBrain MGU4 layer have a fully streaming architecture internally. Then, identical to the Conv and FC layers, longer bitstreams lead to lower errors in the MGU4 layer.

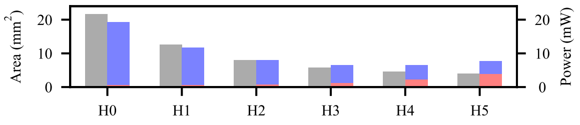
Above results demonstrate that *optimizing DNNs with customized unary operations, i.e., 1) constraining DNN data to legal unary data range and 2) matching operations to accurate unary computing units, guarantees simultaneous high task accuracy and diversity.* For the following hardware evaluation, we focus on motor imagery, as the insights are qualitatively applicable to seizure prediction. The evaluated systolic array has 8-bit data, while the stochastic baseline and uBrain have 10-bit data to provide similar accuracy as in Figure 13a and Figure 13b. Note that 7-bit systolic array is the smallest to offer higher accuracy than 10-bit uBrain, but introduces fractional accesses to byte-addressable memories with insignificant hardware savings; therefore, it is not selected as the baseline.

5.3 Area

5.3.1 Total Area. The total BCI area breakdown is given in Figure 14a, including sense, store and compute stages. For the sense stage, the ADC area in all baselines is identical, 12.3× larger than the ATC area in uBrain. For the store stage, CPU, systolic and stochastic baselines store inputs in off-chip DRAM, as well as DNN weights. On the contrary, uBrain only stores DNN weights in DRAM, as the input temporal-coded bitstreams from ATC directly participate in computation. As the FC3 weights actually occupies 95.1% of total DNN weights, we use identical DRAM size in systolic and stochastic baselines and uBrain to ensure identical capability on diverse tasks. For the compute stage, SC with no intra-layer HTDM for Conv1 and Conv2 has the largest area, and the CPU baseline ranks the second. SC-A has the best area among all stochastic baseline implementations, as it halves the computing kernel most aggressively, but



(a) The Conv1 layer.



(b) The Conv2 layer.

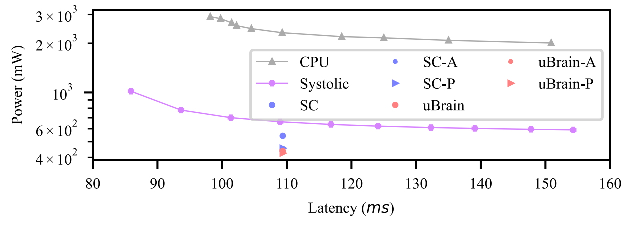
Figure 15: Impact of intra-layer HTDM on the area and power of the Conv1 and Conv2 layers in uBrain. Hx means the computing units are halved x times. $x_{max} = \log_2 c_o$, where c_o is the output count, e.g., 16 for Conv1 and 32 for Conv2.

this does not leads to the best power in SC-P. Similar comparison can be found among uBrain implementations. The systolic baseline has the lowest on-chip area, as it is designed for reconfigurable execution based on a partially parallel architecture, which comes at the cost of high running frequency. In terms of the on-chip area, uBrain-A exhibits 2.0×, 0.1× and 1.3× improvements over CPU, systolic and SC-A designs. Area is the main price uBrain needs to pay for performance, as each DNN layer has dedicated hardware.

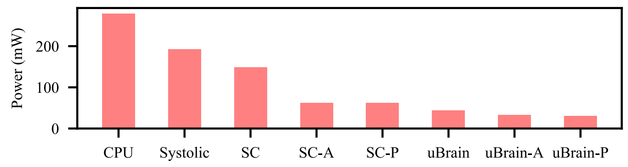
5.3.2 Layerwise Area. The detail layerwise area is in Figure 14b. Such layerwise comparison between the stochastic baseline and uBrain emphasizes the advantages of our multiplier innovation. First, we observe that stochastic implementations, i.e., SC, SC-A and SC-P, always have a higher area than the corresponding uBrain implementations. The largest area reduction across all implementations is 30.7×, occurring in Conv2 from SC to uBrain-A. The largest area reduction across all best-area implementations is 2.9× in FC6 from SC-A to uBrain-A. Second, in all layers, except MGU4, the area ratio of the bottleneck in existing designs, i.e., the bitstream generation logic with RNG and CMP, is reduced tremendously due to the aggressive sharing, effectively demonstrating the advantage of the proposed static multiplier for temporal coding in Figure 10 over the existing design in Figure 5a. The cost of buffers almost maintains constant in each layer, as intra-layer HTDM only reduces the number of the computing units. For the MGU4 layer, the proposed in-stream multiplier introduces negligible overheads to the overall area, as the MatMul in Figure 11 dominates the area. Figure 15 demonstrates that *intra-layer HTDM reduces the area significantly, especially when the data reuse is high*, e.g., Conv2 area ratio reduction is more significant than that in Conv1.

5.4 Frequency and Latency

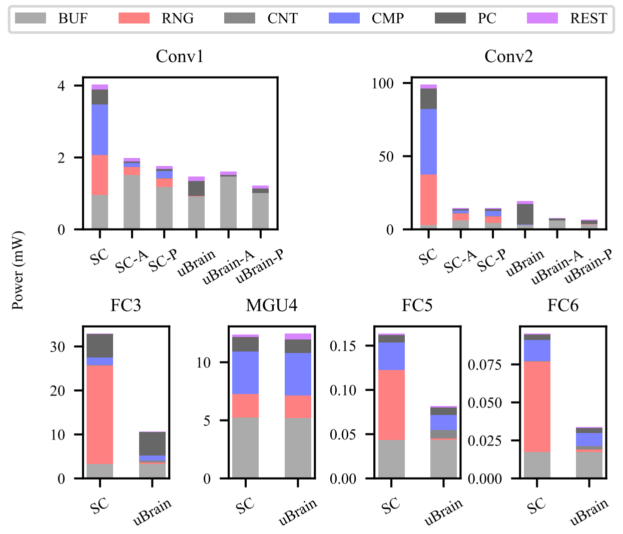
The real-time requirement needs the current data window to be processed before the next window is ready. With 128Hz sampling frequency, the CPU and systolic baselines have a tunable latency depending on the running frequency, but exhibit an upper bound of 20 timestamps, as in Figure 1a. Further considering the max frequency of the CPU and systolic baselines, the correspondent minimum latencies are 98.166ms at 1479MHz and 78.960ms at 400MHz, respectively. On the contrary, the stochastic baseline and uBrain (except FC3) run below 4.2MHz with a fixed latency of 14 timestamps, i.e., 109.375ms, as in Figure 8. All those latencies satisfy the requirement of the human reaction and movement time in Section 2.1.2, i.e., 250ms for total 32 timestamps. The iso-task-latency frequency decrease over CPU and systolic baselines are 219.4× and 18.6×, demonstrating that uBrain's *immediate signal processing after sensing lowers the running frequency significantly by exploiting the sense/store time for compute.*



(a) Total BCI power.



(b) On-chip BCI power. CPU and systolic baselines run at the best power.



(c) Layerwise power.

Figure 16: Power comparison.

5.5 Power

5.5.1 Total Power. The overall BCI power is presented in Figure 16a. uBrain-P outperforms all the rest, even uBrain-A, as more intra-layer HTDM decreases the area but increases the dynamic power due to higher frequency, e.g., in Figure 15a, H4 for Conv1 with the smallest area has higher dynamic power than H3. For the iso-task-latency total power, uBrain outperforms CPU and systolic baselines by 5.5× and 1.6×. For the iso-task-latency DRAM dynamic power, uBrain outperforms the systolic baseline by 2.8×, as uBrain's bandwidth, 0.14GB/s, is 2.1× lower. For the sense power, uBrain outperforms others by 2.6×. Above facts prove that uBrain *offers high power efficiency at all hardware stages.*

5.5.2 On-chip Power. The on-chip power is in Figure 16b. uBrain-P exhibits 9.0×, 6.2× and 2.0× higher power efficiency compared to CPU, systolic and SC-P designs. The fact that uBrain-P has a higher on-chip area but lower power than the systolic baseline confirms the key insight that *immediate signal processing after sensing achieves high power efficiency by exploiting low dynamic power under low running frequency and low-leakage techniques.*

5.5.3 Layerwise Power. The layerwise power is given in Figure 16c. The subunit power ratios of all layers are similar to their area ratios, except the RNG in SC FC3. This is because one input merely controls one weight here, i.e., no RNG sharing for conditional bitstream generation leads to higher dynamic power. The highest power reduction across all implementations and all best-power implementations is 15.2× in Conv2 from SC to uBrain-P and 3.1× in FC3 from SC to uBrain, proving that *intra-layer HTDM opens opportunities to explore desirable trade-offs between area and power.*

5.6 Energy

When running at the best power, all designs have an identical effective compute period of 10 timestamps, i.e., computations of the current window in baselines are done just before the next window is ready. Then, the comparison of energy, energy efficiency and energy delay product rolls back to comparing the best power, and uBrain-P ranks the first.

6 CONCLUSION

In this work, we recognize that the existing brain computer interfaces lack immediate signal processing after sensing, providing low task capability and suboptimal hardware efficiency at all hardware stages. To address this, we propose a unary computing brain computer interface, uBrain, with algorithm-hardware co-design. uBrain leverages emerging deep neural networks to offer high task accuracy and diversity with customized unary operations. Its hardware is optimized with immediate signal processing after sensing to boost the hardware efficiency. Overall, uBrain exhibits 9.0×, 6.2× and 2.0× higher on-chip power efficiency over the CPU, systolic array, and stochastic computing baselines.

ACKNOWLEDGMENTS

We thank our reviewers for their valuable feedback. This work is supported by the Wisconsin Alumni Research Foundation and NSF under award No. CNS-2045985, CNS-1845469 and CNS-2112562.

REFERENCES

[1] Sarah N. Abdulkader, Ayman Atia, and Mostafa Sami M. Mostafa. 2015. Brain Computer Interfacing: Applications and Challenges. *Egyptian Informatics Journal* 16, 2 (2015), 213–230.

[2] Abeer Al-Nafjan, Manar Hosny, Yousef Al-Ohali, and Areej Al-Wabil. 2017. Review and Classification of Emotion Recognition Based on EEG Brain-Computer Interface System Research: A Systematic Review. *Applied Sciences* 7, 12 (2017), 1239.

[3] A. Alaghi, C. Li, and J. P. Hayes. 2013. Stochastic Circuits for Real-Time Image-Processing Applications. In *Design Automation Conference.*

[4] Turky N. Alotaiby, Saleh A. Alshebeili, Faisal M. Alotaibi, and Saud R. Alrshoud. 2017. Epileptic Seizure Prediction Using CSP and LDA for Scalp EEG Signals. *Computational Intelligence and Neuroscience* (2017).

[5] Maria Karoline Andrade, Maíra Araújo de Santana, Giselle Moreno, Igor Oliveira, Jhonnatan Santos, Marcelo Cairrão Araújo Rodrigues, and Wellington Pinheiro dos Santos. 2020. An EEG Brain-Computer Interface to Classify Motor Imagery Signals. *Biomedical Signal Processing: Advances in Theory, Algorithms and Applications* (2020), 83–98.

[6] Gopala K. Anumanchipalli, Josh Chartier, and Edward F. Chang. 2019. Speech Synthesis from Neural Decoding of Spoken Sentences. *Nature* 568, 7753 (2019), 493–498.

[7] ARM. 2021. *Cortex-A57.* Retrieved 2021-11-16 from https://en.wikichip.org/wiki/arm_holdings/microarchitectures/cortex-a57

[8] J Aziz, R Genov, M Derchansky, B Bardakjian, and P Carlen. 2007. 256-Channel Neural Recording Microsystem with On-Chip 3D Electrodes. In *International Solid-State Circuits Conference.*

[9] Rajeev Balasubramonian, Andrew B. Kahng, Naveen Muralimanohar, Ali Shafiee, and Vaishnav Srinivas. 2017. CACTI 7: New Tools for Interconnect Exploration in Innovative Off-Chip Memories. *ACM Transactions on Architecture and Code Optimization* 14, 2 (2017), 1–25.

[10] Mojtaba Bandarabadi, César A. Teixeira, Jalil Rasekhi, and António Dourado. 2015. Epileptic Seizure Prediction Using Relative Spectral Power Features. *Clinical Neurophysiology* 126, 2 (2015), 237–248.

[11] Sebastien Barthelemy and Philippe Boulinguez. 2001. Manual Reaction Time Asymmetries in Human Subjects: The Role of Movement Planning and Attention. *Neuroscience Letters* 315, 1-2 (2001), 41–44.

[12] Yu Hsin Chen, Tushar Krishna, Joel S. Emer, and Vivienne Sze. 2017. Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks. *IEEE Journal of Solid-State Circuits* 52, 1 (2017), 127–138.

[13] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. 2015. BinaryConnect: Training Deep Neural Networks with Binary Weights during Propagations. In *International Conference on Neural Information Processing Systems.*

[14] John P Donoghue. 2002. Connecting Cortex to Machines: Recent Advances in Brain Interfaces. *Nature neuroscience* 5, 11 (2002), 1085–1088.

[15] Nelly Elsayed, Zaghloul Saad, and Magdy Bayoumi. 2017. Brain Computer Interface: EEG Signal Preprocessing Issues and Solutions. *International Journal of Computer Applications* 169, 3 (2017), 975–8887.

[16] Oliver Faust, Yuki Hagiwara, Tan Jen Hong, Oh Shu Lih, and U Rajendra Acharya. 2018. Deep Learning for Healthcare Applications Based on Physiological Signals: A Review. *Computer Methods and Programs in Biomedicine* 161 (2018), 1–13.

[17] David Fick, Gyouho Kim, Allan Wang, David Blaauw, and Dennis Sylvester. 2014. Mixed-Signal Stochastic Computation Demonstrated in An Image Sensor with Integrated 2D Edge Detection and Noise Filtering. In *Custom Integrated Circuits Conference.*

[18] Kosuke Fukumori, Hoang Thien Thu Nguyen, Noboru Yoshida, and Toshihisa Tanaka. 2019. Fully Data-driven Convolutional Filters with Deep Learning Models for Epileptic Spike Detection. In *International Conference on Acoustics, Speech and Signal Processing.*

[19] B. R. Gaines. 1969. Stochastic Computing Systems. *Advances in Information Systems Science* (1969), 37–172.

[20] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *Conference on Computer Vision and Pattern Recognition.*

[21] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. 2000. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. *Circulation* 101, 23 (2000), 215–220.

[22] Christoph Guger, Brendan Z. Allison, and Natalie Mrachacz-Kersting. 2017. Brain-Computer Interface Research: A State-of-the-Art Summary 7. In *SpringerBriefs in Electrical and Computer Engineering.*

[23] Hayrettin Gürkök, Bram van de Laar, Danny Plass-Oude Bos, Mannes Poel, and Anton Nijholt. 2014. Players' Opinions on Control and Playability of a BCI Game. In *International Conference on Universal Access in Human-Computer Interaction.*

[24] EEG Hacker. 2022. *Estimating OpenBCI Battery Life.* Retrieved 2022-02-14 from https://eeghacker.blogspot.com/2015/01/estimating-openbci-battery-life.html

[25] Kaining Han, Junchao Wang, Xingliang Xiong, Qiang Fang, and N. G. David. 2020. A Low Complexity SVM Classifier for EEG Based Gesture Recognition Using Stochastic Computing. In *International Symposium on Circuits and Systems.*

[26] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (Nov. 1997), 1735–1780.

[27] Arthur R. Jensen and Ella Munro. 1979. Reaction Time, Movement Time, and Intelligence. *Intelligence* 3, 2 (1979), 121–126.

[28] B Kamousi, Zhongming Liu, and Bin He. 2005. Classification of Motor Imagery Tasks for Brain-Computer Interface Applications by Means of Two Equivalent

Dipoles Analysis. *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 13, 2 (2005), 166–171.

[29] I. Karageorgos, K. Sriram, J. Veselý, M. Wu, M. Powell, D. Borton, R. Manohar, and A. Bhattacharjee. 2020. Hardware-Software Co-Design for Brain-Computer Interfaces. In *International Symposium on Computer Architecture.*

[30] S. Karen Khatamifard, M. Hassan Najafi, Ali Ghoreyshi, Ulya R. Karpuzcu, and David J. Lilja. 2018. On Memory System Design for Stochastic Computing. *IEEE Computer Architecture Letters* 17, 2 (2018), 117–121.

[31] Haidar Khan, Lara Marcuse, Madeline Fields, Kalina Swann, and Bülent Yener. 2018. Focal Onset Seizure Prediction Using Convolutional Networks. *IEEE Transactions on Biomedical Engineering* 65, 9 (2018), 2109–2118.

[32] Chul Kim, Siddharth Joshi, Hristos Courellis, Jun Wang, Cory Miller, and Gert Cauwenberghs. 2018. Sub-μ V_{rms}-Noise Sub- μ W/Channel ADC-Direct Neural Recording With 200-mV/ms Transient Recovery Through Predictive Digital Autoranging. *Journal of Solid-State Circuits* 53, 11 (2018), 3101–3110.

[33] Stuart T. Klapp. 2003. Reaction Time Analysis of Two Types of Motor Preparation for Speech Articulation: Action as A Sequence of Chunks. *Journal of Motor Behavior* 35, 2 (2003), 135–150.

[34] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *International Conference on Neural Information Processing Systems.*

[35] Anders Krogh and John A. Hertz. 1991. A Simple Weight Decay Can Improve Generalization. In *International Conference on Neural Information Processing Systems.*

[36] Vernon J. Lawhern, Amelia J. Solon, Nicholas R. Waytowich, Stephen M. Gordon, Chou P. Hung, and Brent J. Lance. 2018. EEGNet: A Compact Convolutional Neural Network for EEG-Based Brain-Computer Interfaces. *Journal of Neural Engineering* 15, 5 (2018), 056013.

[37] J.E. Le Douget, A. Fouad, M. Maskani Filali, J. Pyrzowski, and M. Le Van Quyen. 2017. Surface and Intracranial EEG Spike Detection Based on Discrete Wavelet Decomposition and Random Forest Classification. In *International Conference of the IEEE Engineering in Medicine and Biology Society.*

[38] Vincent T. Lee, Armin Alaghi, John P. Hayes, Visvesh Sathe, and Luis Ceze. 2017. Energy-Efficient Hybrid Stochastic-Binary Neural Networks for Near-Sensor Computing. In *Design, Automation & Test in Europe Conference & Exhibition.*

[39] Eric C. Leuthardt, Gerwin Schalk, Jonathan R. Wolpaw, Jeffrey G. Ojemann, and Daniel W. Moran. 2004. A Brain-Computer Interface Using Electrocorticographic Signals in Humans. *Journal of Neural Engineering* 1, 2 (2004), 63–71.

[40] Qi Lian, Yu Qi, Gang Pan, and Yueming Wang. 2020. Learning Graph in Graph Convolutional Neural Networks for Robust Seizure Prediction. *Journal of Neural Engineering* 17, 3 (2020), 035004.

[41] Chin Teng Lin, Che Jui Chang, Bor Shyh Lin, Shao Hang Hung, Chih Feng Chao, and I Jan Wang. 2010. A Real-Time Wireless Brain–Computer Interface System for Drowsiness Detection. *IEEE Transactions on Biomedical Circuits and Systems* 4, 4 (2010), 214–222.

[42] Pengfei Liu, Xipeng Qiu, and Huang Xuanjing. 2016. Recurrent Neural Network for Text Classification with Multi-Task Learning. In *International Joint Conference on Artificial Intelligence.*

[43] Chi Chun Lo, Tsung Yi Chien, Yu Chun Chen, Shang Ho Tsai, Wai Chi Fang, and Bor Shyh Lin. 2016. A Wearable Channel Selection-Based Brain-Computer Interface for Motor Imagery Detection. *Sensors* 16, 2 (2016), 213.

[44] Qian Lou, Wenyang Liu, Weichen Liu, Feng Guo, and Lei Jiang. 2020. MindReading: An Ultra-Low-Power Photonic Accelerator for EEG-Based Human Intention Recognition. In *Asia and South Pacific Design Automation Conference.*

[45] A. Madhavan, T. Sherwood, and D. Strukov. 2014. Race Logic: A Hardware Acceleration for Dynamic Programming Algorithms. In *International Symposium on Computer Architecture.*

[46] Alak Majumder, Monalisa Das, Bipasha Nath, Abir J Mondal, and Bidyut K Bhattacharyya. 2016. Design of Low Noise High Speed Novel Dynamic Analog Comparator in 65nm Technology. In *International Conference Radioelektronika.*

[47] Joseph N Mak and Jonathan R Wolpaw. 2009. Clinical Applications of Brain-Computer Interfaces: Current State and Future Prospects. *IEEE Reviews in Biomedical Engineering* 2 (2009), 187–199.

[48] Vladimir A. Maksimenko, Sabrina Van Heukelum, Vladimir V. Makarov, Janita Kelderhuis, Annika Lüttjohann, Alexey A. Koronovskii, Alexander E. Hramov, and Gilles Van Luijtelaar. 2017. Absence Seizure Control by a Brain Computer Interface. *Scientific Reports* 7, 1 (2017), 1–8.

[49] Malik M.Naeem Mannan, Shinjung Kim, Myung Yung Jeong, and M. Ahmad Kamran. 2016. Hybrid EEG–Eye Tracker: Automatic Identification and Removal of Eye Movement and Blink Artifacts from Electroencephalographic Signal. *Sensors* 16, 2 (2016), 241.

[50] G. Maor, X. Zeng, Z. Wang, and Y. Hu. 2019. An FPGA Implementation of Stochastic Computing-Based LSTM. In *International Conference on Computer Design.* 38–46.

[51] Dennis J. McFarland and Jonathan R. Wolpaw. 2011. Brain-Computer Interfaces for Communication and Control. *Commun. ACM* 54, 5 (2011), 767–791.

[52] Hassan Mostafa and Yehea I. Ismail. 2013. Highly-Linear Voltage-to-Time Converter (VTC) Circuit for Time-Based Analog-to-Digital Converters (T-ADCs). In *International Conference on Electronics, Circuits, and Systems.*

[53] Angel Mur, Raquel Dormido, Jesús Vega, Natividad Duro, and Sebastian Dormido-Canto. 2016. Unsupervised Event Characterization and Detection in Multichannel Signals: An EEG Application. *Sensors* 16, 4 (2016), 590.

[54] Elon Musk. 2019. An Integrated Brain-Machine Interface Platform With Thousands of Channels. *Journal of Medical Internet Research* 21, 10 (2019), e16194.

[55] Vinod Nair and Geoffrey E Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *International Conference on International Conference on Machine Learning.*

[56] Sho Nakagome, Trieu Phat Luu, Yongtian He, Akshay Sujatha Ravindran, and Jose L. Contreras-Vidal. 2020. An Empirical Comparison of Neural Networks and Machine Learning Algorithms for EEG Gait Decoding. *Scientific Reports* 10, 1 (2020), 1–17.

[57] Shahrzad Naraghi. 2009. *Time-Based Analog to Digital Converters.* Ph. D. Dissertation. University of Michigan.

[58] NeuroBB. 2022. *Muse battery replacement.* Retrieved 2022-02-14 from https://neurobb.com/t/muse-battery-replacement/665

[59] NeuroSky. 2022. *TGAT1/TGAM1.* Retrieved 2022-02-14 from http://neurosky.com/biosensors/eeg-sensor/

[60] Nvidia. 2021. *Jetson Nano Developer Kit.* Retrieved 2021-10-14 from https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[61] Nvidia. 2021. *NVIDIA Jetson Linux Driver Package Software Features.* Retrieved 2021-10-14 from https://docs.nvidia.com/jetson/l4t/index.html

[62] Tae Young Oh, Hoeju Chung, Jun Young Park, Ki Won Lee, Seunghoon Oh, Su Yeon Doo, Hyoung Joo Kim, ChangYong Lee, Hye Ran Kim, Jong Ho Lee, Jin Il Lee, Kyung Soo Ha, YoungRyeol Choi, Young Chul Cho, Yong Cheol Bae, Taeseong Jang, Chulsung Park, Kwangil Park, SeongJin Jang, and Joo Sun Choi. 2015. A 3.2 Gbps/pin 8 Gbit 1.0 V LPDDR4 SDRAM With Integrated ECC Engine for Sub-1 V DRAM Core Operation. *IEEE Journal of Solid-State Circuits* 50, 1 (2015), 178–190.

[63] N Onizawa, D Katagiri, W J Gross, and T Hanyu. 2014. Analog-to-Stochastic Converter Using Magnetic-Tunnel Junction Devices. In *International Symposium on Nanoscale Architectures.*

[64] Ahmet Remzi Ozcan and Sarp Erturk. 2019. Seizure Prediction in Scalp EEG Using 3D Convolutional Neural Networks with an Image-Based Approach. *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 27, 11 (2019), 2284–2293.

[65] Natasha Padfield, Jaime Zabalza, Huimin Zhao, Valentin Masero, and Jinchang Ren. 2019. EEG-Based Brain-Computer Interfaces Using Motor-Imagery: Techniques and Challenges. *Sensors* 19, 6 (2019), 1423.

[66] Giuseppe Placidi, Danilo Avola, Andrea Petracca, Fiorella Sgallari, and Matteo Spezialetti. 2015. Basis for the Implementation of an EEG-Based Single-Trial Binary Brain Computer Interface through the Disgust Produced by Remembering Unpleasant Odors. *Neurocomputing* 160 (2015), 308–318.

[67] Rabie A. Ramadan and Athanasios V. Vasilakos. 2017. Brain Computer Interface: Control Signals Review. *Neurocomputing* 223 (2017), 26–44.

[68] Mamunur Rashid, Norizam Sulaiman, Anwar P. P. Abdul Majeed, Rabiu Muazu Musa, Ahmad Fakhri Ahmad, Bifta Sama Bari, and Sabira Khatun. 2020. Current Status, Challenges, and Possible Solutions of EEG-Based Brain-Computer Interface: A Comprehensive Review. *Frontiers in Neurorobotics* 14 (2020), 25.

[69] Seyed Navid Resalat and Valiallah Saba. 2016. A Study of Various Feature Extraction Methods on a Motor Imagery Based Brain Computer Interface System. *Basic and Clinical Neuroscience* 7, 1 (2016), 13.

[70] Yannick Roy, Hubert Banville, Isabela Albuquerque, Alexandre Gramfort, Tiago H. Falk, and Jocelyn Faubert. 2019. Deep Learning-Based Electroencephalography Analysis: A Systematic Review. *Journal of Neural Engineering* 16, 5 (2019), 051001.

[71] G. Schalk, D.J. McFarland, T. Hinterberger, N. Birbaumer, and J.R. Wolpaw. 2004. BCI2000: A General-Purpose Brain-Computer Interface (BCI) System. *IEEE Transactions on Biomedical Engineering* 51, 6 (2004), 1034–1043.

[72] M. Schönauer, S. Alizadeh, H. Jamalabadi, A. Abraham, A. Pawlizki, and S. Gais. 2017. Decoding Material-Specific Memory Reprocessing During Sleep in Humans. *Nature Communications* 8 (2017), 1–9.

[73] Andrew B Schwartz. 2004. Cortical Neural Prosthetics. *Annual Review of Neuroscience* 27 (2004), 487–507.

[74] Mohammed Shoaib, Niraj Jha, and Naveen Verma. 2011. A Low-Energy Computation Platform for Data-Driven Biomedical Monitoring Algorithms. In *Design Automation Conference.*

[75] Hyeonuk Sim and Jongeun Lee. 2017. A New Stochastic Computing Multiplier with Application to Deep Convolutional Neural Networks. In *Design Automation Conference.*

[76] Ranganatha Sitaram, Andrea Caria, and Niels Birbaumer. 2009. Hemodynamic Brain–Computer Interfaces for Communication and Rehabilitation. *Neural Networks* 22, 9 (2009), 1320 – 1328.

[77] S. R. Sreeja and Debasis Samanta. 2019. Classification of Multiclass Motor Imagery EEG Signal Using Sparsity Approach. *Neurocomputing* 368 (2019), 133–145.

[78] N. J. Stevenson, K. Tapani, L. Lauronen, and S. Vanhatalo. 2019. A Dataset of Neonatal EEG Recordings with Seizure Annotations. *Scientific Data* 6 (2019), 1–8.

[79] James L Stone and John R Hughes. 2013. Early History of Electroencephalography and Establishment of the American Clinical Neurophysiology Society. *Journal of Clinical Neurophysiology* 30, 1 (2013), 191–212.

[80] Akara Supratak, Hao Dong, Chao Wu, and Yike Guo. 2017. DeepSleepNet: A Model for Automatic Sleep Stage Scoring Based on Raw Single-Channel EEG. *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 25, 11 (2017), 81–91.

[81] Arwa M. Taqi, Fadwa Al-Azzo, M. Mariofanna, and Jassim M. Al-Saadi. 2017. Classification and Discrimination of Focal and Non-Focal EEG Signals Based on Deep Neural Network. In *International Conference on Current Research in Computer Science and Information Technology.*

[82] S Sharifi Tehrani, Warren J Gross, and Shie Mannor. 2006. Stochastic Decoding of LDPC Codes. *IEEE Communications Letters* 10, 10 (2006), 716–718.

[83] Mario Tudor, Lorainne Tudor, and Katarina Ivana Tudor. 2005. Hans Berger (1873-1941)–the History of Electroencephalography. *Acta Medica Croatica : Casopis Hravatske Akademije Medicinskih Znanosti* 59, 4 (2005), 307–313.

[84] Aosen Wang, Zhanpeng Jin, Chen Song, and Wenyao Xu. 2015. Adaptive Compressed Sensing Architecture in Wireless Brain-Computer Interface. In *Design Automation Conference.*

[85] Jing Wang, E. Sanchez-Sinencio, and F. Maloberti. 2000. Very Linear Ramp-Generators for High Resolution ADC BIST and Calibration. In *Midwest Symposium on Circuits and Systems.*

[86] Xiao Wei Wang, Dan Nie, and Bao Liang Lu. 2014. Emotional State Classification from EEG Data Using Machine Learning Approach. *Neurocomputing* 129 (2014), 94–106.

[87] Marco Weiergräber, Anna Papazoglou, Karl Broich, and Ralf Müller. 2016. Sampling Rate, Signal Bandwidth and Related Pitfalls in EEG Analysis. *Journal of Neuroscience Methods* 268 (2016), 53–55.

[88] Francis R. Willett, Donald T. Avansino, Leigh R. Hochberg, Jaimie M. Henderson, and Krishna V. Shenoy. 2021. High-Performance Brain-to-Text Communication via Handwriting. *Nature* 593, 7858 (2021), 249–254.

[89] Thomas A. Wozny, Witold J. Lipski, Ahmad Alhourani, Efstathios D. Kondylis, Arun Antony, and R. Mark Richardson. 2017. Effects of Hippocampal Low-Frequency Stimulation in Idiopathic Non-Human Primate Epilepsy Assessed via A Remote-Sensing-Enabled Neurostimulator. *Experimental Neurology* 294 (2017), 68–77.

[90] Di Wu, Yun Chen, Qichen Zhang, Yeong-Luh Ueng, and Xiaoyang Zeng. 2016. Strategies for Reducing Decoding Cycles in Stochastic LDPC Decoders. *IEEE Transactions on Circuits and Systems II: Express Briefs* 63, 9 (2016), 873–877.

[91] Di Wu, Jingjie Li, Ruokai Yin, Hsuan Hsiao, Younghyun Kim, and Joshua San Miguel. 2020. uGEMM: Unary Computing Architecture for GEMM Applications. In *International Symposium on Computer Architecture.*

[92] Di Wu, Jingjie Li, Ruokai Yin, Hsuan Hsiao, Younghyun Kim, and Joshua San Miguel. 2021. uGEMM: Unary Computing for GEMM Applications. *IEEE Micro* 41, 3 (2021), 50–56.

[93] Di Wu and Joshua San Miguel. 2022. uSystolic: Byte-Crawling Unary Systolic Array. In *International Symposium on High-Performance Computer Architecture.*

[94] Di Wu and Joshua San Miguel. 2019. In-Stream Stochastic Division and Square Root via Correlation. In *Design Automation Conference.*

[95] Di Wu, Ruokai Yin, and Joshua San Miguel. 2021. In-Stream Correlation-Based Division and Bit-Inserting Square Root in Stochastic Computing. *IEEE Design & Test* 38, 6 (2021), 53–59.

[96] Il Min Yi, Naoki Miura, and Hideyuki Nosaka. 2021. A 4-GS/s 11.3-mW 7-bit Time-Based ADC With Folding Voltage-to-Time Converter and Pipelined TDC in 65-nm CMOS. *IEEE Journal of Solid-State Circuits* 56, 2 (2021), 465–475.

[97] Dalin Zhang, Lina Yao, Xiang Zhang, Sen Wang, Weitong Chen, and Robert Boots. 2018. Cascade and Parallel Convolutional Recurrent Neural Networks on EEG-Based Intention Recognition for Brain Computer Interface. In *Conference on Artificial Intelligence.*

[98] Guo Bing Zhou, Jianxin Wu, Chen Lin Zhang, and Zhi Hua Zhou. 2016. Minimal Gated Unit for Recurrent Neural Networks. *International Journal of Automation and Computing* 13, 3 (2016), 226–234.

[99] Junwei Zhou and Andrew Mason. 2002. Communication Buses and Protocols for Sensor Networks. *Sensors* 2, 7 (2002), 244–257.