INVITED PAPER

Efficient Biosignal Processing Using Hyperdimensional Computing: Network Templates for Combined Learning and Classification of ExG Signals

By Abbas Rahimi, *Member IEEE*, Pentti Kanerva, Luca Benini, *Fellow IEEE*, and Jan M. Rabaey, *Fellow IEEE*

ABSTRACT | Recognizing the very size of the brain's circuits, hyperdimensional (HD) computing can model neural activity patterns with points in a HD space, that is, with HD vectors. Key examined properties of HD computing include: a versatile set of arithmetic operations on HD vectors, generality, scalability, analyzability, one-shot learning, and energy efficiency. These make it a prime candidate for efficient biosignal processing where signals are noisy and nonstationary, training data sets are not huge, individual variability is significant, and energy-efficiency constraints are tight. Purely based on native HD computing operators, we describe a combined method for multiclass learning and classification of various ExG biosignals such as electromyography (EMG), electroencephalography (EEG), and electrocorticography (ECoG). We develop a full set of HD network templates that comprehensively encode body potentials and brain neural activity recorded from different electrodes into a single HD vector without requiring domain expert knowledge or *ad hoc* electrode selection process. Such encoded HD vector is processed as a single unit for fast one-shot learning, and robust classification. It can be interpreted to identify the most useful features as well. Compared to state-of-the-art counterparts, HD computing enables online, incremental, and fast learning as it demands less than a third as much training data as well as less preprocessing.

KEYWORDS | Biosignal classification; brain-inspired computing; brain–machine interface; ECoG; EEG; EMG; error-related potential; hyperdimensional (HD) computing; human–machine interface; interpretable machine learning; motor imagery; network architectures; one-shot learning; seizure detection; vector symbolic architectures

I. INTRODUCTION

Some of the most compelling application domains of the Internet of Things (IoT) relate to how humans interact

Manuscript received February 5, 2018; revised July 26, 2018; accepted September 1, 2018. This work was supported by the ETH Zurich Postdoctoral Fellowship program; the Marie Curie Actions for People COFUND Program; the EU H2020 project OPRECOMP under Grant 732631; the Intel Strategic Research Alliance program on Neuromorphic Architectures for Mainstream Computing; and the NSF 16-526: Energy- Efficient Computing: from Devices to Architectures (E2CDA), a Joint Initiative between NSF and SRC. *(Corresponding author: Abbas Rahimi.)*

A. Rahimi is with the Department Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA 94720 USA, and also with the Department of Information Technology and Electrical Engineering, ETH Zurich, 8092 Zürich, Switzerland (e-mail: abbas@eecs.berkeley.edu; abbas@ee.ethz.ch).

P. Kanerva is with the Helen Wills Neuroscience Institute, University of California at Berkeley, Berkeley, CA 94720 USA (e-mail: pkanerva@berkeley.edu).

L. Benini is with the Department of Information Technology and Electrical Engineering, ETH Zurich, 8092 Zürich, Switzerland, and also with the Department of Electrical, Electronic and Information Engineering, University of Bologna, 40136 Bologna, Italy (e-mail: luca.benini@iis.ee.ethz.ch; luca.benini@unibo.it).

J. M. Rabaey is with the Department of Electrical, Electronic and Information Engineering, University of Bologna, 40136 Bologna, Italy.

Digital Object Identifier 10.1109/JPROC.2018.2871163

with the world around them and with the cyberworld through “wearable” devices. The growing sophistication of these devices requires a continuous reduction in energy-per-operation. Unfortunately, with the slowdown of traditional semiconductor scaling, leakage and uncertainty [1] limit the amount of energy scaling that can be reached [2]. The only viable solution is to rethink functionality to cope with uncertainty by adopting computational approaches that are inherently robust to uncertainty [3]. Advances in learning-based computing for IoT increase energy efficiency toward TOPS/Watt [4], but further improvement requires a novel look at data representations, associated operations, circuits, and materials and substrates that enable them [5]. Monolithic 3-D integrated nanotechnologies [6], [7] combined with novel brain-inspired computational paradigms that support fast learning and fault tolerance could lead the way [5].

Emerging hyperdimensional (HD) computing [8] is based on the understanding that brains compute with patterns of neural activity that are not readily associated with scalar numbers. In fact, the brain’s ability to calculate with numbers is feeble. However, due to the very size of the brain’s circuits, we can model neural activity patterns with points of a HD space, that is, with HD vectors. When the dimensionality is in the thousands, operations on HD vectors create a computational behavior with unique features in terms of robustness and efficiency [9], [10].

HD computing brings into play the rich and subtle mathematics of HD spaces. It relates partly to the linear algebra and probabilities of artificial neural nets, and partly to the abstract algebra and geometry of HD spaces. Groups, rings, and fields over HD vectors become the underlying computing structures, with permutations, mappings, and inverses as primitive computing operations, and with randomness programmatically inscribed in the way new objects and entities are labeled. However, its performance depends on good design—instead of automated training—of a network architecture that consists entirely of the HD primitive operations [11]–[13].

In this paper, we first focus on the key properties of HD computing resulting from the application of a well-defined set of arithmetic operations on HD vectors. Key properties that are examined include: generality, scalability, analyzability, one-shot learning, energy efficiency, natural performance without domain expert knowledge and less preprocessing [5], [14]–[19]. These leading properties take HD computing beyond the typical text and language applications [20]–[26], and make it a prime candidate for a new category of applications: integrated learning-based wearable/implantable devices, for which biosignal training data sets are small, individual variability is significant, privacy, latency, and energy efficiency demands are tight [27]–[30].

More specifically, we design a full set of HD network templates to ease constructing an efficient and complete representational architecture for handling both learning and classification tasks in the domain of personalized devices. The proposed templates collectively handle various types of biosignals including electromyography (EMG), electroencephalography (EEG), and electrocorticography (ECoG)—collectively referred to as ExG. As concrete examples, we target multiclass learning and inference in 1) EMG-based hand gesture recognition for human–machine interfaces; 2) EEG-based brain–computer interfaces; and 3) ECoG-based seizure detection. Our network templates encode body potentials or brain neural activity recorded from various electrodes into a single HD vector capturing the temporal and spatial features of the signals, without requiring any *ad hoc* electrode selection process or domain expert knowledge. The encoded HD vector is used for fast learning and robust classification; besides, it can be exploited to identify the most useful features. Remarkably, the network templates are designed purely based on the native HD operations without involving any biologically implausible or inefficient optimization algorithm such as gradient descent and backpropagation. Such simplicity of networks enables efficient implementation of both learning and classification tasks with fully binary operations leading to significant energy saving, e.g., [16].

Further, HD learning follows the “one-shot” approach, that is object categories are learned from one or few examples and only in a single pass (i.e., one epoch) over the training data. We demonstrate the benefits of HD computing by comparing it with the state-of-the-art machine learning methods for biosignal processing including support vector machines (SVMs) [31]–[33], Gaussian classifiers [34], feedforward multilayer perceptron (MLP) [33], and convolutional neural networks (CNNs) [35]. Compared to the state-of-the-art counterparts, our experimental results show that HD computing enables online, incremental, and fast learning as it demands less than a third as much training data and preprocessing. While this paper focuses on EMG, EEG, and ECoG signals, other streaming multidimensional sensor data such as electrocardiography (ECG), speech, or smell could equally be applicable [36]–[38].

This paper is organized as follows. In Section II, we introduce HD computing (with concise description in the Appendix). In Section III, we discuss key properties of HD computing for designing efficient biosignal processing architectures. In Section IV, we present our main contributions by proposing a full set of HD network templates to efficiently learn and classify various types of biosignals. Our experimental results are described in Section V followed by discussion in Section VI. Section VII concludes the paper.

II. BACKGROUND IN HD COMPUTING

This section provides a background in HD computing. The brain’s circuits are massive in terms of numbers of neurons and synapses, suggesting that large circuits are fundamental to the brain’s function. HD computing explores this idea by looking at computing with HD vectors as ultrawide words. It is rooted in the observation that key aspects of human memory, perception, and cognition can be explained by the mathematical properties of HD spaces, and that a powerful system of computing can be built on the rich algebra of HD vectors. The difference between traditional computing and HD computing is apparent in the elements that we compute with. In traditional computing, the elements are Booleans, numbers, and memory pointers, whereas in HD computing they are HD vectors. HD vectors are d-dimensional (the number of dimensions is in the thousands) and (pseudo)random with independent and identically distributed (i.i.d.) components. They thus conform to a holographic or holistic representation: the encoded information is distributed equally over all the d components such that no component is more responsible to store any piece of information than another. Such representation maximizes robustness for the most efficient use of redundancy [8]. Other examples of such computing structures include holographic reduced representations [39], semantic pointer architecture [40], binary spatter codes [21], multiply–add–permute (MAP) coding [41], random indexing [20], and vector symbolic architectures (VSAs) [11], with a quick summary in [5].

The number of different, nearly orthogonal HD vectors is very large when the dimensionality is in the thousands [8], [9]. Two such HD vectors can now be combined into a new HD vector using simple vector-space operations, while preserving the information of the composing HD vectors with high probability. Computing with HD vectors begins with selecting a set of random HD vectors to represent basic objects. These HD vectors are also thought of as random labels. For example, in a language recognition application [23], [24], the letters of the alphabet as the inputs can be the basic objects, and they are assigned to random labels. In the same vein, in a biosignal processing application, each input electrode is assigned to a random label, independently of all the other labels. They serve as seed HD vectors, and they are used to make representations for more complex objects. To generate seed HD vectors, we use bipolar dense codes of equally probable $+1$s and -1s, i.e., $\{-1,+1\}^d$ where $d = 10\,000$; this dimensionality works particularly well for our applications, but it is essentially a hyperparameter that can be tuned [42]. In the following, we describe similarity measure and arithmetic operations using this code.

A. Similarity Measurement of HD Vectors

An essential operation in HD computing is the computation of the distance (or similarity) between two HD vectors. For dense bipolar HD vectors,[1] we use cosine similarity as the distance metric between two HD vectors by measuring the cosine of the angle between them using a dot product. It is defined as $\cos(A,B) = |A' * B'|$, where A' and B' are the length-normalized vectors of A and B, respectively, and $|C|$ denotes the sum of the elements in C. It is thus a measure of orientation and not magnitude: two HD vectors with the same orientation have a cosine similarity of 1, two orthogonal HD vectors have a similarity of 0, and two HD vectors diametrically opposed have a similarity of -1.

B. Arithmetic Operations on HD Vectors

HD computing builds upon a well-defined set of arithmetic operations with random HD vectors. These arithmetic operations are used for encoding and decoding patterns. The power and versatility of the arithmetic derives from the fact that the basic operations, namely addition and multiplication, form an algebraic structure resembling a field, to which permutations give further expressive power.

We use a variant of the MAP coding described in [41]. The MAP operations on HD vectors are defined as follows. Pointwise multiplication of two HD vectors A and B is denoted by $A * B$, and pointwise addition is denoted by $A + B$. Multiplication[2] takes two vectors and yields a third, $A * B$, that is dissimilar (approximately orthogonal) to the two and is suited for variable binding; and addition, or bundling, takes several vectors and yields vector $[A + B + \ldots + X]$ that is maximally similar to them and is suited for representing sets. The brackets $[\cdots]$ mean that the sum vector is normalized to $\{+1,-1\}^d$ based on the sign, with ties broken at random. Finally, the third operation is permutation, ρ, that rotates the coordinates of HD vector. A simple way to implement this is as a cyclic right shift by one position. All these operations have a complexity of $\mathcal{O}(d)$ and produce a d-dimensional vector.

The usefulness of HD computing comes from the nature of the operations. Specifically, addition produces a vector that is similar to the argument vectors—the inputs—whereas multiplication and random permutation produce a dissimilar vector; multiplication and permutation are invertible, addition is approximately invertible; multiplication distributes over addition; permutation distributes over both multiplication and addition; multiplication and permutation preserve similarity, meaning that two similar vectors are mapped to equally similar vectors elsewhere in the space.

Operations on HD vectors can produce results that are approximate or "noisy" and need to be associated with the "exact" vectors. For that, a list of known (noise-free) seed HD vectors is maintained in a so-called "item" or "clean-up" memory. When presented with a noisy HD vector, the item memory (IM) outputs the HD vector that is most similar or closest. Making this work reliably requires high dimensionality. With 10 000-b HD vectors, 1/3 of the bits can be flipped at random and the resulting HD vector can still be identified with the originally stored one with very high probability.

[1] In this paper, we use only capitalized italic letters to indicate HD vectors; they may also appear with a subscript.

[2] By default, we refer to the pointwise multiplication ($*$) unless otherwise mentioned.

The operations make it possible to encode and manipulate sets, sequences, and lists—in essence, any data structure. A data record consists of a set of fields (keys, variables, or attributes) and their values (fillers). A data record consisting of fields x, y, z with values a, b, c can be encoded into an HD vector H as follows. First, random seed HD vectors are chosen for the fields and the values (X, Y, Z, A, B, C), and are stored in the IM. We then encode the record by binding the fields to their values with multiplication and by adding together the bound pairs

$$H = [(X * A) + (Y * B) + (Z * C)].$$

This resulting representation is holographic because the fields are superposed over each other—there are no spatially identifiable fields. Importantly, the value of x can be extracted from this holographic representation by multiplying H with the inverse of X, which for $*$ is X itself: $A' = X * H$. The resulting HD vector A' is given to the IM which returns A as the most similar stored HD vector. An analysis of this example would show how the properties of addition and multiplication come to play (see also the Appendix). A thing to note about the operations is that addition and multiplication approximate an algebraic structure called a field, to which permutation gives further expressive power.

The permutation is a reversible mapping that generates a dissimilar quasi-orthogonal HD vector of its input. In geometry sense, the permutation rotates the HD vector in the space. The rotated HD vector is uncorrelated with all the other HD vectors. The permutation can be used to encode a sequence of items, e.g., a sequence of three letters abc called a trigram. We make a trigram HD vector by permuting the first letter vector twice, the second letter vector once, and use the third letter vector as is, and then by multiplying the three HD vectors component by component as

$$\rho(\rho A * B) * C = \rho\rho A * \rho B * C.$$

This efficiently distinguishes the sequence abc from, e.g., acb or any other trigram that may share letters or differ only in letter order.

HD computing has been described above in terms of dense bipolar HD vectors. The representational system is closed under the aforementioned MAP operations. Throughout this paper, we refer to this representational bipolar space and the related MAP operations unless otherwise stated. Note that HD computing supports one more operation that is rarely used: scalar multiplication (weighting). The scalar multiplication of an HD vector A with a scalar value v is denoted by $v \cdot A$. This results in the scaled version of A since every component of the HD vector is multiplied with the same scalar value. If the scalar value belongs to real numbers, the result of scalar multiplication is in real space as well, and is often combined with addition. Sections IV-A3 and IV-B4 use the scalar multiplication operation.

III. KEY PROPERTIES OF HD COMPUTING FOR EFFICIENT BIOSIGNAL PROCESSING

In this section, we first present some compelling applications of biosignal processing and describe their challenges. We then highlight how key properties of HD computing can respond to these challenges.

To focus the discussion, this paper considers three types of biosignals: 1) EMG signals as recording of the electrical activity produced by the skeletal muscles; 2) EEG signals as recording of the electrical activity of the brain from the scalp; and 3) ECoG, or intracranial EEG (iEEG), as a type of recording that uses electrodes placed directly on the exposed surface of the brain. EMG is widely used in various directions to create a human–machine interface at the neuromuscular level [43]. We specifically target the processing task of the neuromuscular EMG signals; one possible outcome is the recognition of gestures that can serve as the primary commands to control a prosthetic arm [44], [45]. In contrast, brain–computer interfaces based on EEG signals aim to provide a communication and control channel between the human brain and external devices. We focus on two types of brain activity measured by the noninvasive EEG signals to recognize a user's intentions: error-related potentials and motor imagery. When a user recognizes an error during monitoring of an external agent, an error-related potential (ERP) can be measured in the EEG signal; recognition of the ERP can be utilized to correct and improve the behavior of the external agent [34], [46], [47]. Alternatively, in motor imagery (MI) brain–computer interface, a user is asked to imagine movements of different parts of the body that arises the brain activity of the motor cortical areas; this MI recording can be decoded to recognize the desired movement commands [30], [32], [48], [49]. Finally, we focus on a seizure detection task based on ECoG signals for patients with drug-resistant epilepsy [50].

Processing and classification of these biosignals pose a number of challenges including the following. Operating with a variety of biosignal acquisitions ranging from a pair of differential EMG electrodes in time-domain to complex acquisitions with more EEG electrodes in frequency-domain demands a versatile learning and inference (classification) method. Further, these biosignals are noisy and nonstationary—especially the brain signals change over time—with large individual variability among subjects that demand continuous recalibration and personalized (subject-specific) learning. Such personalized learning should be effective with small training data, and for on-chip operation with limited amount of resources and energy. The on-chip learning reduces privacy and security risks by limiting the attack surface to only the personalized device, rather than device, gateway, and cloud, which is aligned with the concept of federated learning [28]

Table 1 Overview of Various Biosignal Processing Applications and Their Related HD Computing Architecture

Task Complexity					Configured HD Network Template (HD Classifier)		
Recording	No. Elec.	No. Classes	No. Subjects	Classifier's Inputs	Mapping	Encoder	Associative Memory
EMG	4	5	5	Time series	IM+CIM (m=21, linear)	Spatial-temporal	5 prototypes
EEG ERP	64	2	6	Time series	IM+CIM (m=100, linear)	Temporal-spatial	2 prototypes
ECoG	36–100	2	16	Symbols (LBP)	IM	Spatial-histogram	2 prototypes
EEG MI	16	3	5	Features	Weighting	Spatial	3 prototypes
EEG MI	22	4	9	Features	Weighting	Spatial	4 prototypes
EEG MI	22	4	9	Features	IM+CIM (m=100, logarithmic)	Spatial	4 prototypes

based on the principle of focused collection or data minimization [29]. At the same time, the learning should be interpretable with the goal to understand the underlying features related to the classification task [51]. In the following, we describe how HD computing can address these challenges.

A. Scalable Computational Paradigm With Versatile Arithmetic Operations

HD computing offers a simple and complete computational paradigm based on learning, and builds upon a well-defined and versatile set of operations with random HD vectors. The MAP arithmetic operations can encode and decode patterns in a huge quasi-orthogonal hyperspace [41]. The encoding/decoding is scalable and versatile. HD computing has been initially used to operate with a single streaming input of characters to encode texts [23], [24]; we have extended the encoder to operate with simultaneous analog biosignal inputs [14]–[17], [19]. The encoder flexibly operates with various types of ExG biosignal acquisitions, and simply scales with different numbers of electrodes (see Table 1): ranging from 36 to 100 ECoG electrodes with the highest signal-to-noise ratio (SNR) [19], to EMG signals with relatively lower SNR using few patch electrodes [14] or denser flexible electrode array [15], and finally to 16–64 EEG electrodes with the lowest SNR [17]. Execution of HD computing on an 8-core parallel ultralow-power (PULP) accelerator shows that the EMG encoder can scale to process up to 256 electrodes while meeting the 10-ms classification constraint for real-time EMG tasks [16]. Besides versatile classification, which finds associations of a new item with a set of known items associated with a label, the arithmetic operations can be used for query processing as well, which answers desired questions about a particular stored item [22].

B. Learning Transparent Codes With Interpretable Features

Thanks to the well-defined set of arithmetic operations with inverses, HD computing produces transparent (i.e., analyzable) codes with interpretable features. For example, in the typical application of EEG ERP, a domain expert carefully determines a subset of relevant electrodes (e.g., two electrodes out of 64), depending upon the subject; this subset of selected electrodes is used for subsequent classification [34]. At first, HD computing does not require such domain expert knowledge for the electrode selection process, and hence operates naturally with all the 64 electrodes at negligible loss of accuracy. Besides, the learned HD vectors can be analyzed to identify what electrodes provide meaningful data for the classification. It has been shown that instead of asking for the domain expert knowledge, HD computing can identify the same subset of electrodes as relevant by measuring the relative distances between the learned prototype HD vectors [18]. Producing such transparent codes also enables verification of the learned model [52], and is in sharp contrast to blind application of conventional learning methods that produce a "black box."

C. Learning Is One Shot, Fast, and Computationally Balanced With Respect to Classification

In contrast to other neuro-inspired approaches in which learning is computationally much more demanding than subsequent classification, learning in HD computing is based on the same algorithms as classification. The learning algorithm works in "one shot," namely, object categories are learned from one or few examples, by using only a portion of training data, in a single pass without impacting the classification accuracy. For instance, state-of-the-art SVM [31] for the EMG classification task reaches to 97.8% accuracy by using the full set of training data, while the HD algorithm achieves the same level of accuracy by using only 1/3 of the training data in one pass [14]. Using a larger number of 64 EMG electrodes, HD computing demonstrates one-shot learning—in the true sense of the word—by training from a single gesture per class [15]. We also observe that HD computing quickly learns from one or two seizures and perfectly detects unseen seizures for the majority of patients (ten out of 16) [19]. Similar benefit is observed for the EEG ERP classification task [17], [18]: the HD algorithm learns $\approx 3\times$ faster by using only 34% of training trials while maintaining an average accuracy of 70.5%, which is higher than the state-of-the-art classifier using the full set of training trials.

In addition, since the same algorithm is used for learning and classification, the architecture is ideal for online and continuous learning. The new examples can be learned by incrementally updating the associative memory (AM) described in Section IV-C. This enables the fast learning to be executed in a real-time online fashion, and the classifier can be updated (i.e., partially retrained) with new samples to address the nonstationary nature of biosignals.

D. Less Preprocessing

Most preprocessing of the electrode signal can be eliminated in HD computing as it can operate with noisy inputs; it is also robust to electrodes that do not carry meaningful information. In the EMG-based gesture recognition task, HD computing can maintain its accuracy when the four patch electrodes are replaced by 64 flexible noisier electrodes with a lower SNR [15]. Similarly, HD computing can continue its natural operation with all 64 electrodes and less preprocessing by removing a common average reference (CAR) filter [53] from EEG signals that results in only a slight loss of accuracy (from 74.5% to 71.7%) [17]. Overall, we have observed that HD computing is a nice fit for fast and one-shot learning and classification of noisy ExG signals with minimal information about the task: e.g., in the absence of domain expert knowledge, and by training with much less data and preprocessing.

E. Energy Efficiency

At its very core, HD computing is about manipulating and comparing large patterns within the memory itself. The MAP operations allow a high degree of parallelism by needing to communicate with only a local component or its immediate neighbors. Distance computation can be performed in a distributed fashion; it is the only operator proportional to vector dimension. An architecture based on HD computing can be seen as an extremely wide dataflow processor with small instruction set of bit-level operations. Further, logic can be tightly integrated with the memory and all computations are fully distributed that can save energy [6], [7]. This forms a fundamental departure from the traditional von Neumann architectures where data have to be transported to the processing unit and back, creating the infamous memory wall.

Further, simplicity of HD computing is another important factor for energy efficiency. HD computing requires far fewer operations than other approaches such as SVMs, CNNs, k-nearest neighbors (KNNs), and MLP for the same functionality [14], [16], [19], [24]. For instance, in the EMG classification task, we use the SVM with fixed-point operations, instead of floating point, that leads to best performance preserving the accuracy [54]. HD computing achieves the same level of accuracy as the SVM on a commercial embedded ARM Cortex M4 using only 1/2 as much power [16]. This is due to the fact that HD computing mostly uses basic bitwise operations. This simplicity allows scalable execution on the embedded 8-core PULP accelerator with bitmanipulation instruction extensions that achieves $10\times$ higher energy efficiency than the ARM Cortex M4 [16].

The same is true about memory accesses. This compensates for the very wide words used in HD computing. The memory requirement for HD computing scales linearly: e.g., in the language recognition task, by moving from a trigram ($n = 3$) to pentagram ($n = 5$), HD computing requires two more extra HD vectors whereas the memory required by the baseline KNN grows exponentially with n. Using pentagrams of letters, the baseline requires $500\times$ larger memory than HD [24], or some hashing-based algorithm to manage the memory, yet requiring more of it than the HD-based approach. As another example, in the seizure detection task, the MLP demands $5\times$–$13\times$ larger memory than HD computing to store its weights, even if we optimistically assume that all its weights could be quantized to one bit [19]. In the following, we describe two more properties of HD computing that can further improve energy efficiency.

1) Robustness under low SNR. By its very nature, HD computing is extremely robust in the presence of failures, defects, variations, and noise of computing fabrics, all of which are synonymous to ultralow energy computation. It has been shown that HD computing degrades very gracefully in the presence of temporary and permanent faults compared to a baseline KNN classifier for the language recognition task: by injecting the intermittent hardware-induced errors in both classifiers, HD computing tolerates $8.8\times$ higher probability of failure per individual memory cells [24]; considering the permanent hard errors, HD computing tolerates $60\times$ higher probability of failures [7]. The robust operation under low SNR conditions and high variability perfectly matches with emerging nanotechnologies promising to deliver substantial energy savings [5]–[7].

   Such robustness of HD computing is achieved by its inspiration from brain's circuits: (pseudo)randomness, hyperdimensionality, and fully distributed holographic representation. Symbols represented with HD vectors begin with i.i.d. components and when combined with the MAP operations, the composite HD vectors also appear as identically distributed random vectors, and the independence of the individual components is mostly preserved. Specifically, the pointwise multiplication and addition are i.i.d. preserving; when the permutation is combined with the multiplications to encode n-grams, we end up with vectors whose components are identically distributed and nearly independent. This means that a failure in a component of HD vectors is not "contagious." At the same time, failures in a subset of components are compensated for by the holographic nature of the data representation, i.e., the error-free components can still provide a useful representation that is similar enough to the original HD vector. This inherent robustness also eliminates the need for asymmetric error protection in memory units. This type of robustness is absolutely unique, and enables both aggressive scaling of device dimensions and integration complexity as well as SNR levels.

2) Sparsity in time and space. Biologically plausible sparsity [55] is essential to the efficiency of the fully distributed computational paradigms offered by HD computing [56]–[60]. At any point only a fraction of

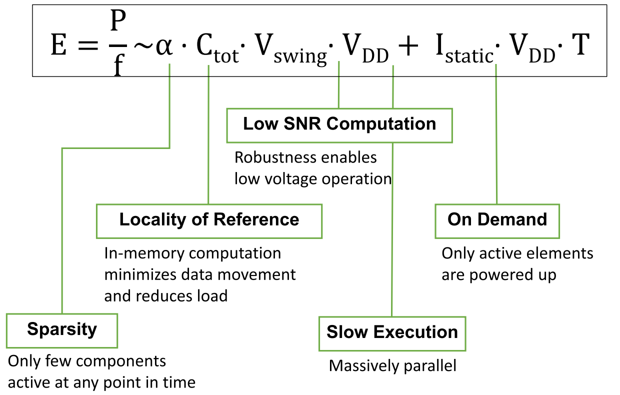

$E = \frac{P}{f} \sim \alpha \cdot C_{tot} \cdot V_{swing} \cdot V_{DD} + I_{static} \cdot V_{DD} \cdot T$
Low SNR Computation
Robustness enables low voltage operation
Locality of Reference
In-memory computation minimizes data movement and reduces load
On Demand
Only active elements are powered up
Sparsity
Only few components active at any point in time
Slow Execution
Massively parallel

Fig. 1. ***How properties of HD computing lead to ultralow-energy computing.***

the memory/logic fabric should be active leading to a "mostly dark operational" model. However, it requires both representations that are intrinsically sparse and new operations that preserve them along with asynchronous execution. For instance, a sparse binary representation—where the number of ones is significantly less than zeros—along with a componentwise context-dependent thinning operation can lower the switching activity and hence power consumption [59]. This is applied to various pattern recognition tasks including the EMG classification with detail discussions about choice of density, operations, and capacity in [59].

Overall, Fig. 1 provides a perspective on the aforementioned attributes of HD computing responsible for ultralow-energy computation. The equation for total energy consumption (E) consists of two major components, the dynamic and the static dissipations, where P is total power, f is frequency, α is switching activity, C_{tot} is total load and short-circuit capacitances, V_{swing} is voltage swing, V_{DD} is supply voltage, I_{static} is static and leakage current, and T is time period. Fig. 1 illustrates how these various terms of power consumption are impacted by the properties of HD computing. For instance, sparsity directly lowers the switching activity factor α of dynamic power. Targeting V_{DD}, robustness of HD computing perfectly copes with the uncertainty which is the largest hindrances to lower V_{DD}; besides, V_{DD} can be further lowered by slowing down the execution enabled by the massively parallel HD operations. These properties in combination can significantly reduce energy consumption.

IV. HD NETWORK TEMPLATES FOR COMBINED LEARNING AND CLASSIFICATION OF BIOSIGNALS

In this section, we present the main contributions of the paper. Note that HD computing offers a simple and complete computational paradigm that is easy to work with at the level of HD vector representation and related mathematical operations. Nevertheless, it is relatively harder to work at the level of complete representational architectures as mentioned by Gayler in his inspiring paper [11, p. 6]: "Typical connectionist architectures rely on training procedures to achieve their effectiveness. However, VSAs [Vector Symbolic Architectures, or HD computing architectures] provide no opportunity for training to substitute for architectural effectiveness. That is, good performance depends on good *design* rather than automated training, and this is a harder research task." To address this issue for biosignal processing, we *design* a full set of efficient network templates based on HD computing. These few HD network templates ease constructing a complete representational architecture by providing predefined *options* that can be *configured* to meet specific goals. Each HD network template generates HD vectors for different types of inputs—including time-domain or frequency-domain ExG—and chooses how to combine these symbolic level HD vectors to create more complex representations. More importantly, the templates are purely constructed based on the native operations of HD computing, and do not require any inefficient or biologically implausible algorithm, such as backpropagation for optimizations and weight tuning. This enables efficient implementation of the constructed architecture—to perform combined learning and classification tasks—with fully binary operations.

Fig. 2 illustrates a universal HD architecture, for solving supervised classification tasks, that is uniformly composed of three main modules: mapping, encoding, and AM. For each of these modules, we design the network templates providing predefined options and attributes for a variety of purposes. The three modules can be then configured and cascaded to essentially build an HD data flow processor. The mapping module first maps the input biosignals from the original representation to the HD vectors where they can be manipulated by means of the versatile arithmetic operations reside in the encoder module. The output of the encoder is another HD vector that encloses our event of interest for learning/classification in the HD space. Finally, the AM module turns the output of the encoder to a prototype HD vector representing a given class. During training, the AM stores and updates a set of prototype HD vectors; it finds the closest one to the output of the encoder during testing. Following is a detailed description of the three modules.

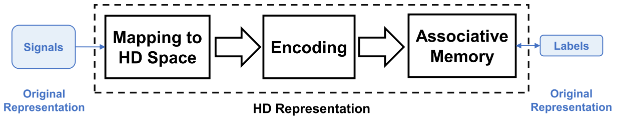


Fig. 2. ***A universal HD architecture for combined learning and classification that is composed of: mapping, encoding, and AM. Proposed HD network templates are shown in Fig. 4 for mapping and encoding, and in Fig. 5 for AM.***

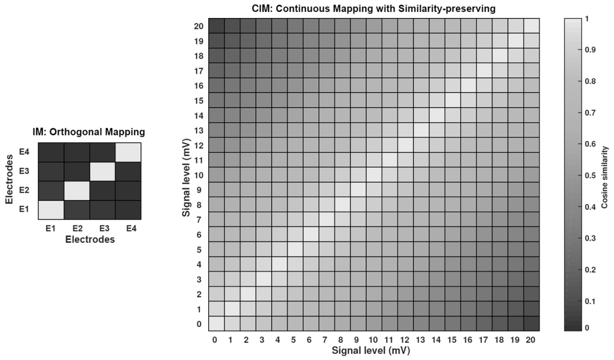


Fig. 3. ***Comparison between cosine similarity matrices of mapped items using: 1) orthogonal mapping of four electrode names via the IM, in the left; and 2) continuous mapping of quantized electrode signals with $m = 21$ levels via the CIM, in the right.***

A. Mapping to HD Space

The fist step is to map raw inputs or features from the original representation to the HD representation space. The HD representation is typically produced through a mapping also known as projection. In the following, we present three projection options that can be chosen based on the type of inputs or features; we later discuss about other options, including learning a projection in Section VI.

1) Orthogonal Mapping: When an input can be described by a finite alphabet of independent symbols, its mapping to the HD vectors is simple. This can be done by a class of data-analysis methods that is referred to as symbolization [61]. Symbolization describes the process of transforming raw experimental measurements into a series of discrete symbols. Each symbol can be simply assigned to a unique HD vector that is chosen randomly. We can maintain all these HD vectors in the IM. The IM here acts as a symbol table or dictionary of all the HD vectors defined in the system. For instance, in the European language recognition task [23], [24], the discrete inputs (the 26 letters of the alphabet and the space) are the initial items, and they are assigned to random HD vectors with i.i.d. components. On the other hand, in a biosignal processing task, the electrodes with unique names are the primary inputs, e.g., four electrodes in the EMG task namely "e1," "e2," "e3," and "e4." Since the name of every electrode is a unique string, it can be easily mapped to an HD vector using the IM with four entries. The IM, shown in Fig. 3, represents the four basic electrodes by assigning a unique quasi-orthogonal HD vector to every electrode: $E1 \perp E2 \perp E3 \perp E4$. They stay fixed throughout the computation, and they serve as seeds from which further representations are made.

As another alternative, projection to the binary HD vectors can be implemented by means of a cellular automaton [62], [63]. The input features in the original representation can be first binarized and then passed through several steps of computation with a cellular automaton. A cellular automaton consists of a regular grid of cells each in one of the binary states. Every cell evolves in time according to a fixed rule with a chaotic behavior that can produce a sequence of (pseudo)random HD vectors. The state of a cell on the next computational step depends solely on its current state and the states of its neighbors. After several steps of computation, the time-space state of cellular automaton is the projection to HD space as described in [64].

One other option is to exploit the random process variations that are naturally present in any deeply scaled and low voltage nanotechnology process [6], [7], [38]. The application of the use of process variations in HD mapping is reported in [38], where groups of randomized delay lines are used to perform random indexing. Another approach is to make use of linear feedback shift registers (LFSRs) to produce sequences of random seeds with pseudo-i.i.d. behavior. One caveat is that a generated seed may be close to the permuted version of the previous seed (see Section II-B for the realization of permutation operation), as both rely on single bit shifts. We therefore need to use a second permutation that interferes minimally with circular shift.

2) Continuous Mapping That Preserves Similarity: The aforementioned orthogonal mapping is well matched to the output of symbolization, or to the input data and features in the form of discrete symbolic primitives (letters or words) that can be readily mapped to the HD vectors. However, in the biosignal processing applications, each electrode produces an analog time-varying signal where the signal level has an amplitude in real values. Hence, we decouple mapping of the name and the signal level of an electrode. The latter one demands a different mapping method to the HD space to preserve "similarity" between a range of real values in the original representation to their corresponding mapped HD vectors.

For this method of mapping, we limit our case to signal levels that are first quantized using a quantization step with a fixed number of levels (m). Accordingly, we have extended the notion of IM to a continuous item memory (CIM) that can map a range of quantized signal levels [14]. The CIM utilizes a method [65] of mapping quantities "continuously" to the HD vectors that is simpler than the method in [36]. In this continuous vector space, two orthogonal endpoint HD vectors are generated for the minimum and maximum levels in the range. HD vectors for intermediate levels are then generated by linear interpolation between these two endpoints so that the cosine similarity of HD vectors corresponds to the closeness of levels.

For example, the quantization with 21 levels $(m = 21)$ is suitable for electrodes with an amplitude of 0–20 mV in the EMG-based hand gesture recognition task. We choose a random HD vector for the minimum level $(V_{\min})$ and randomly flip $d/2/(m-1)$ of its bits for each successively higher level (once flipped, a bit will not be flipped back). The HD vectors for the minimum and maximum levels will then be $d/2$ bits apart or orthogonal to each other, i.e., $V_{\text{min}} \perp V_{\text{max}}$. These HD vectors are stored in the CIM for reuse. Fig. 3 (right) illustrates the cosine similarity between each pair of HD vectors in the CIM. As shown, by this mapping a linearly decreasing similarity is preserved between the HD vectors from V_{min} to V_{max}; however, it could be nonlinear based on the nature of input data or features as described in [59].

3) Mapping With Scalar Multiplication (Weighting): For those real-valued features that is not clear how the quantization should be done, we can use a method of mapping with weighting. The mapping with weighting directly projects a real-valued feature, with full range, to the HD space; this projection results in an HD vector whose components are real values. The projection can be done by a pair of IM and the scalar multiplication $(\cdot)$ as the basic operation in the linear algebra described in Section II-B. The IM first assigns a random bipolar HD vector to the entity of a feature $(F1)$. Then, by means of scalar multiplication $F1$ is multiplied by the scalar real value of the feature (v) that produces a real valued HD vector: $v \cdot F1$. This scalar multiplication, or weighting, scales the initially assigned HD vector by modulating magnitude of the vector components without changing its direction.

Although the method of mapping with weighting is simple, normalized, and seamlessly operational with any range of features, it requires energy-hungry floating-point operations and storage. Hence, this style of operation should be avoided as much as possible. During the early design phase, we can initially use this mapping option when we have no information about how the quantization and mapping should be performed. Next, we can replace it by a CIM that is able to reflect well the real-valued features in the HD space; the CIM can be evaluated by different techniques that linearly or nonlinearly change the similarity between the mapped bipolar HD vectors. By providing an example in Section V-D1, we show the tradeoff between these two methods of mapping.

B. Encoding

After projection to the HD space, further progressive representations should be formulated to encode the event of interest for learning and classification. The events of interest in biosignals processing, e.g., the hand gestures or the mental commands, have typically spatial and temporal components to be captured. Following is a detailed description of such encoding options that can be chosen as appropriate.

1) Spatial Encoder: As we described in Section IV-A, an electrode maps its name to an HD vector (e.g., $E1$) via the IM; it separately maps its signal level at a time point t to $V1_t$ via the CIM. This mapping is illustrated in Fig. 4(a). The purpose of a spatial encoder is to combine these mapped HD vectors across all the electrodes at a given time-aligned sample (t), and represent them in a single HD vector. To do so, we draw an analogy from [22] to generate a holistic HD vector representing data from all the electrodes by using a set of field–value pair. The electrode name corresponds to a field of a traditional data record, and its signal level corresponds to the value for the field. As shown in Section II-B, the field and the value can be bound by the multiplication operation. With this, for example for the first electrode, we can jointly project its name and its signal value to a bipolar bound HD vector: $E1 * V1_t$. To complete the holistic record, we bundle (via the addition) all such bound HD vectors to construct a single spatial HD vector as shown in Fig. 4(a):

$$S_t = [(E1 * V1_t) + (E2 * V2_t) + (E3 * V3_t) + (E4 * V4_t)].$$

The aforementioned spatial encoder outputs a bipolar HD vector, and works well when the number of electrodes is odd. However, when the number of electrodes is even (as well as small), pointwise addition of the bipolar HD vectors may produce 0s, so we end up with a ternary system unless we break the ties. The ties should be broken randomly and reproducibly. It can be done for example by adding an additional random HD vector to the record; however, it makes the encoder noncausal: two equal sets of input data in the original space become slightly dissimilar in the projected HD space [63]. Alternatively, using a constant HD vector would lead to all output HD vectors being slightly similar to each other even if they are supposed to be orthogonal. To address this issue, instead of choosing a random/constant HD vector, we compute an augmented HD vector [63] that is reproducible with the same set of input data, e.g., by further binding two already bound HD vectors: $(E1 * V1_t) * (E2 * V2_t)$. We add this augmented HD vector to the record, for example

$$\begin{aligned} S_t = [&(E1 * V1_t) + (E2 * V2_t) + (E3 * V3_t) + (E4 * V4_t) \\ &+ ((E1 * V1_t) * (E2 * V2_t))]. \end{aligned}$$

2) Temporal Encoder: The spatial encoder captures a vertical slicing of signals among all the electrodes at a given time. However, the events of interest for learning and classification have time-dependent components, e.g., a series of samples over time. We can temporally encode a sequence of symbols by using the permutation operation ρ. As described in Section II-B, the permutation can encode a sequence of n letters to form an n-gram HD vector. By analogy, a sequence of three spatial HD vectors with consecutive time stamps (S_{t-2}, S_{t-1}, and S_t) is encoded as follows: the first HD vector S_{t-2} is permuted twice $\rho^2 S_{t-2}$, the second HD vector S_{t-1} is permuted once ρS_{t-1}, and finally there is no permutation for the last HD vector S_t. These three new HD vectors are then combined with the pointwise multiplication into a trigram HD vector:

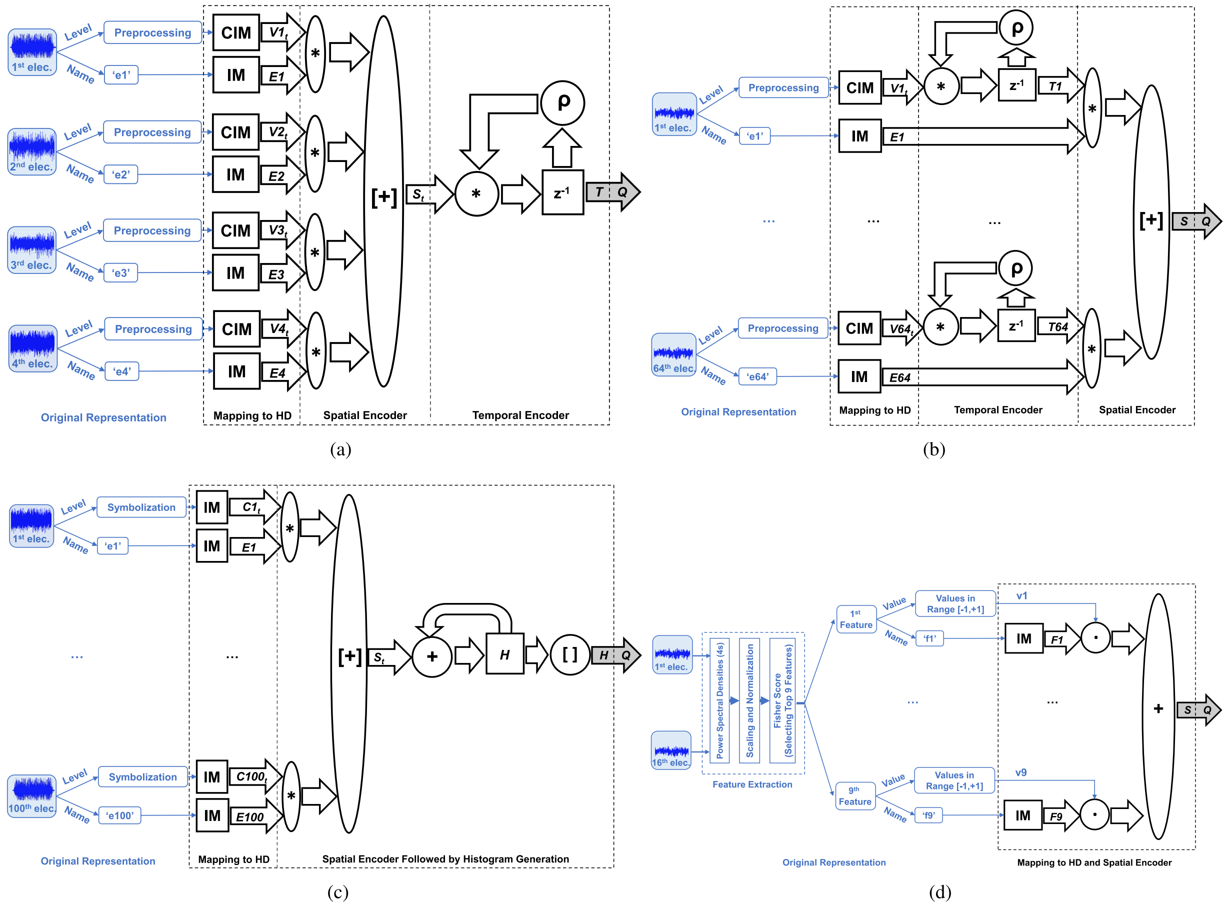


Fig. 4. *HD network templates to encode time-domain (a) EMG, (b) ERP EEG, and (c) ECoG signals: 1) preprocessing or symbolization in the original representation space; 2) mapping to the HD space; and 3) spatial (S), temporal (T), and histogram (H) encoders. HD network template to encode frequency-domain features for the (d) MI EEG task: 1) preprocessing and feature extraction in frequency domain of original representation; 2) mapping selected features to the HD space with weighting methods; and 3) spatial encoder. The output of encoding (Q) is used in the AM (Fig. 5) for learning and inference. (a) Template for the EMG task. (b) Template for the ERP EEG task. (c) Template for the ECoG task. (d) Template for the MI EEG task.*

$T = \rho^2 S_{t-2} * \rho S_{t-1} * S_t$. For n-grams at large, this becomes

$$T = \prod_{i=0}^{n-1} \rho^i S_{t-i}.$$

Fig. 4(a) shows the temporal encoder that computes the n-gram recursively where a single sample delay is denoted by z^{-1}. This eases the implementation of the temporal encoder by using the distributivity of the permutation over the multiplication as described in Section II-B. The temporal encoder is applied in cascade after the spatial encoder. Hence, HD vector T is the output of spatial–temporal encoding for representing the EMG hand gestures. T can be seen as the outcome of encoding module for the AM (referring to Fig. 2).

With the temporal encoding, one important step is to determine the proper size of an n-gram to be able to capture the entire event of interest. It has been done by downsampling the signal and statistically measuring the number of downsamples available in a hand gesture, or in a mental command. For instance, the EMG hand gestures can be represented by n-grams where $n \in \{3, 4, 5\}$ [14] whereas the ERP EEG decoding requires larger n-gram sizes where $n \in \{16, \ldots, 29\}$. With this larger n-gram size, we choose to change the order of encoders for the ERP EEG task as shown in Fig. 4(b): first doing the temporal encoding of every electrode, and then doing the addition to compute the spatial HD vector (S) as the output of encoding. First doing the temporal encoding allows us to analyze the n-gram HD vector produced from each electrode to distinguish meaningful electrodes from irrelevant electrodes in Section V-B3.

3) Spatial Encoder and Histogram Generation: Here, we describe a version of spatial encoder that is followed by a histogram generation to reflect the distribution of symbols over a specific window of time. This encoding is useful for ECoG signals that are directly transformed to symbols via symbolization [see Fig. 4(c)]. Symbolization may be efficiently achieved by mapping a sequence of ECoG samples into an l-bit code, i.e., a 1-D local binary pattern (LBP) [66]. An LBP code reflects the relational aspects between consecutive values of the ECoG signals, i.e., whether their amplitudes increase or decrease. Our symbolization considers six consecutive ECoG samples to compute a 6-b ($l = 6$) LBP code, and moves by one sample [19]. These LBP codes generate 2^l different symbols that are fed into the IM for mapping to the HD space. The IM assigns a quasi-orthogonal HD vector to every LBP code (in total, 64 different LBP codes). To combine these HD vectors across all the electrodes, the encoder generates a spatial record (S), in which an electrode name is treated as a field, and its LBP code as the value of this field. Hence, the IM also maps the name of electrodes to quasi-orthogonal HD vectors $E1 \perp E2 \ldots \perp E100$, for a patient with the maximum number of 100 electrodes [see Fig. 4(c)]. This allows, for example, to bind the name of the first electrode ($E1$) to its corresponding LBP code at time t ($C1_t$). This binding ($E1 * C1_t$) generates a new set of quasi-orthogonal HD vectors to represent LBP codes per electrode that effectively reduces the size of IM from 64×100 HD vectors to $64 + 100$ HD vectors. The spatial record (S) is then constructed by bundling the bound HD vectors of all electrodes

$$S_t = [E1 * C1_t + E2 * C2_t + \ldots + E100 * C100_t].$$

The HD vector S_t is computed for every new sample, and holographically represents the spatial information about the LBP codes of all electrodes. The next step is to compute the histogram of LBP codes inside a moving window that should be wide enough to theoretically permit at least a single occurrence of all possible LBP codes [67]. Considering a sampling frequency of 512 Hz, a window of 0.5 s contains 256 LBP codes that provides a high probability for every code to occur inside this window because $256 > 2^{l+1}$. The histogram computed from this window can be used as a signature for seizures: interictal (between seizures) and ictal (during seizures) states show different distributions of LBP codes [66], [67]. This shows that the distribution of LBP codes, not necessarily their sequence, is an important indicator to distinguish between ictal and interictal state. To estimate the histogram of LBP codes inside the window, a multiset of temporally generated S_t vectors is computed as

$$H = [S_1 + S_2 + \ldots + S_{256}].$$

The bundling is applied in the temporal domain through accumulation of S_t vectors $t \in \{1, \ldots, 256\}$ that are produced within the window and then thresholding at half (i.e., normalization).

4) Spatial Encoder With Weighting: When we are given a set of extracted features that are not from the time domain, and collectively capture the entire event of interest, we can use the spatial encoder to combine all of them into a spatial HD vector. Since these features are often complicated and mixed (e.g., multiscale), the weighting method (in Section IV-A3) can be used as an option to map them. Hence, we propose to construct a spatial encoder with weighting that is well suited to holistically map a feature set without quantization. Examples include the frequency-domain features for the MI EEG task, in which we extract power spectral density (PSD) for different frequency bands, and finally select nine top features among all the electrodes as shown in Fig. 4(d).

The mapping requires an IM to assign a unique set of orthogonal HD vectors to the feature set, i.e., $F1 \perp F2 \ldots \perp F9$. The extracted features have scalar values, e.g., $v1$ for the first feature. To represent this feature in the HD space, we perform a scalar multiplication between the value of feature and its corresponding HD vector: $v1 \cdot F1$. These scaled HD vectors are added across all the features to compute the real-valued spatial HD vector

$$S = v1 \cdot F1 + v2 \cdot F2 + \ldots + v9 \cdot F9.$$

This new spatial encoder computes the pointwise sum of the feature HD vectors weighted by the scalers. This encoder is a perfect match to automatically map any given feature set when there is no scheme for the feature quantization and mapping. This is done by working with real instead of bipolar vector components. The cost of this is so large that it should be used only when necessary (see Section VI).

C. Associative Memory

In the proposed HD architecture (see Fig. 2), the last module is the AM that directly operates with the output of encoding (Q). This output HD vector can come from any previously proposed encoders, for instance, from the spatial–temporal encoding [Fig. 4(a)] that makes $Q = T$, likewise from the temporal–spatial encoding [Fig. 4(b)] and the spatial encoding with weighting [Fig. 4(d)] by $Q = S$. The AM completes the supervised learning method by assigning a label to the output of encoding module. The method is based on the notion of a class prototype. The class prototype is an HD vector (P) representing all items from the entire class aligned with the notion of prototypical networks [68].

The AM initially allocates a set of class prototypes whose number (k) is equal to the number of classes in the task. As shown in Fig. 5(a), during training, for every trial, the AM selects a related class prototype HD vector based on the provided label, and updates it by adding the HD vector produced from the output of encoding (Q). For learning from the current training trial with a label of, e.g., "Label$_1$," the AM selects the corresponding class prototype HD vector ($P1$) and bundles it via the addition operation to the output of encoding: $P1 \mathrel{+}= Q$. This ensures that a single prototype representation emerges for each class. Such accumulative updates continue until the end of training. Simplicity of this update operation enables incremental learning from different examples during the

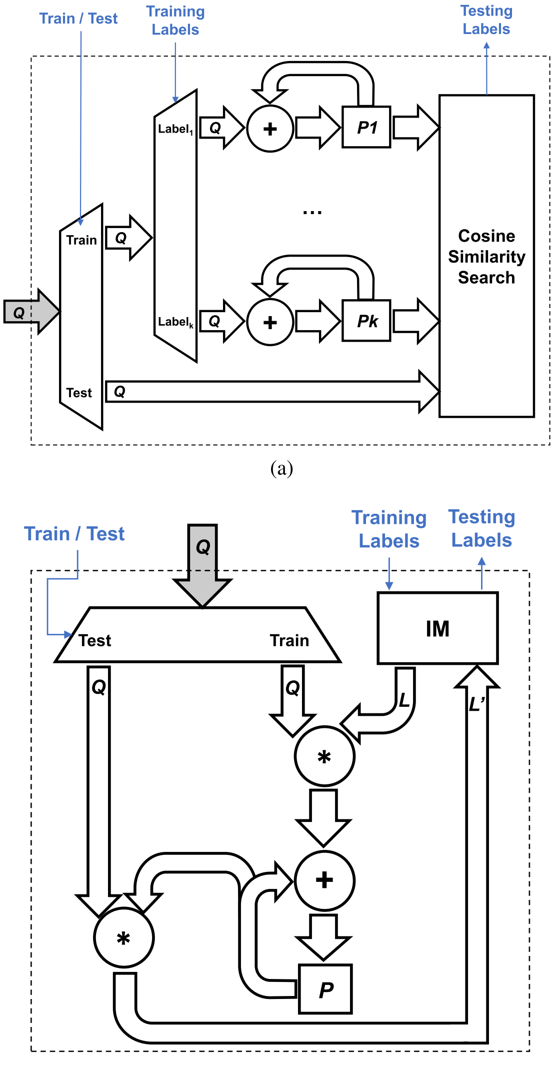


Fig. 5. *Two compatible AM architectures. Both support a different number of classes and two modes of operations: train and test. (a) AM with multiple prototypes. (b) AM with unified prototype.*

course of online functioning. By the end of training, the AM contains all the class prototype HD vectors—as the learned distributed patterns—that are organized based on their labels. The class prototype vectors can be normalized to $\{+1, -1\}^d$ based on the sign of components. Note the difference between the AM and the IM: the IM holds seed HD vectors that are assigned constants and stand for electrodes/letters/signal levels, while the AM holds prototype HD vectors that are learned and stand for classes.

The same mapping and encoding are used for both learning (training) and classification (inference, or testing); however, the AM has a train versus test mode. When testing, we call the output of the encoder a query HD vector since its class label is unknown. The query HD vector of the test trial is then sent to the AM to identify its source class. The AM in the test mode determines the class of the test trial by comparing its query HD vector to all the learned prototype HD vectors using the cosine similarity. The cosine similarity search computes k similarity scores among which the AM selects the highest one and returns its associated label as the class that the query HD vector has been generated from. Efficient solutions are required to search through a large AM [69]. The initial implementation of the AM on the PULP accelerator with eight cores and specialized bitwise instructions shows $10\times$ faster execution compared to a single core without optimized bitwise instructions [16].

1) Associative Memory With a Unified Prototype: Fig. 5(b) illustrates another version of the AM that requires only one unified prototype HD vector (P). This AM, instead of storing the prototype HD vectors separately per class, computes a single prototype HD vector as a record where the "fields" are the prototype HD vectors and the values are their mapped class labels in the HD space. To map the class labels to the HD space, we pair the AM to the IM that assigns a set of orthogonal HD vectors to the label set. For every training trial, its associated label is mapped to an HD vector (L) which is bound to the output of encoding $L * Q$; this bound pair is added to the unified prototype HD vector $P \mathrel{+}= L * Q$. During testing, we retrieve the label of test trial by unbinding the query HD vector from the unified prototype HD vector $L' = Q * P$, which results in the noisy HD vector of label (L'). To do the cleanup, we use the IM that returns L based on similarity search.

V. EXPERIMENTAL RESULTS

In this section, we present our experimental results for the proposed HD network templates developed in Matlab.[3] We show how they can be configured to be used in different biosignal processing applications, compare them with the state-of-the-art counterparts, and highlight their benefits. Table 1 gives an overview of these configured HD network templates—simply referred to as HD classifiers from here on—and their assigned biosignal processing tasks. We start from the simple task of multiclass EMG hand gesture recognition from few time-domain inputs, and move, step by step, to other tasks with increased complexity. Next, we consider the EEG ERP binary classification task with 64 time-domain inputs. We then consider the ECoG-based seizure detection task that demands a universal encoder to operate with different patients having 36–100 electrodes implanted. Finally, we consider two challenging tasks for the MI EEG: 1) a task of classifying three classes with frequency-domain features extracted from 16 electrodes; and 2) a task of four-class classification from a large number of multiscales features extracted from 22 electrodes. In the following sections, we describe each of these tasks in detail and present our findings. The accuracy term throughout this paper is referred as macroaveraging test accuracy that computes a simple average over classes for the test set.

[3] A collection of projects and codes based on HD computing is available at:https://github.com/HyperdimensionalComputing/collection

A. EMG-Based Hand Gesture Recognition

The EMG data acquisition is based on four sensors that cover the muscles involved in the hand movement from a physiological point of view. The data set [14] for five subjects is based on the recording of the EMG signals of the common hand gestures in a daily life. The selected gestures are: closed hand, open hand, two-finger pinch, point index, and rest position, forming five classes. For every gesture, the recording is composed of ten repetitions of the gesture, each with 3 s of the muscular contraction. Every contraction is followed by 3-s rest position.

The gestures are sampled at 500 Hz, and for the preprocessing a low-pass filter extracts the envelope of the signal, and a notch filter removes the residual power-line interference. The preprocessed signals from the four electrodes are downsampled by 250. These four preprocessed and downsampled values are used as the input features. The SVM, as the state-of-the-art method [31], learns and classifies with these four time-aligned features. However, a gesture is spanned over time for 3 s, and generates up to six sequences of such time-aligned features that makes a linear growth in the number of features from 4 to 24. The SVM cannot efficiently classify with this linearly increased number of features and its accuracy drops significantly [14]. On the other hand, the configured HD classifier for the EMG task, shown in the first row of Table 1 and Fig. 4(a), can capture the temporal component of gestures. This is accomplished by the spatial–temporal encoding that uses an n-gram where $n \in \{3, 4, 5\}$; the size of n-gram is set per subject.

Fig. 6 compares the learning curves of HD classifier and the SVM: plotting the classification versus the number of training trials. The bars show the average accuracy, and the errors are the standard deviation across five subjects. The HD classifier shows an average accuracy of 86.8% (7% higher than the SVM) when only 10% of the total data set is used for training. Although the classification accuracy is improved by increasing the training trials for both of them, the learning slope of HD is superior to the SVM. By increasing the training trials to 25%, the HD classifier reaches to 97.8% which is 8.1% higher than the SVM. After this learning point, increasing the number of training trials is not useful for the HD classifier as it has already learned and is able to generalize very well. However, this is not the case for the SVM since it requires $3.2\times$ as much training data (i.e., 80% of total trials) to reach the level of accuracy as the HD classifier with 25% of trials.

HD classifier shows further advantages using 64 high-density flexible EMG electrodes [15]: It achieves an average classification accuracy of 96.64% for five gestures, with only 7% degradation when training and testing across different days—a large improvement over degradations of more than 30% using the SVM. Moreover, HD maintains this accuracy when trained with only three trials of gestures; it also demonstrates comparable accuracy with the SVM when trained with one trial per gesture—one-shot learning.

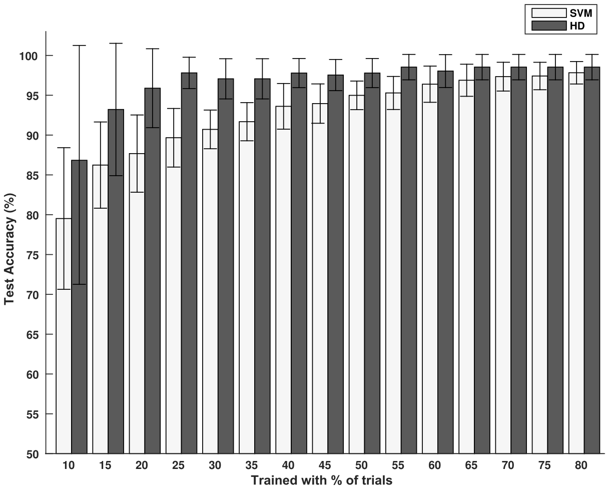


Fig. 6. ***Learning curve in the EMG task: the HD classifier learns*** $3.2\times$ ***faster than the SVM to reach to the maximum accuracy of 97.8%.***

B. Single-Trial Binary Classification of EEG ERPs

We consider a data set of EEG ERPs for six subjects [70]. The subjects are seated in front of a computer screen where a cursor moves horizontally (to left or right) in order to reach a target. The subject has no control over the cursor's movement and is asked only to observe the performance of an autonomous agent that controls the cursor, knowing that the goal is to reach the target. To study the EEG ERPs generated by observing an erroneous movement of the cursor, there is a probability of ≈0.20 in every trial for the cursor to move in the wrong direction (i.e., opposite to the target location). A trial is labeled as "correct" if the cursor moves toward the target; otherwise it is labeled as "error." Trials have an approximate duration of 2 s. There are two recording sessions, the first one for training and the second for testing. Each experimental session consists of ≈640 trials. Full details of the experimental protocol are provided in [34]. In the following, we explain their method for the EEG signal acquisition, preprocessing, and classification. We refer to it as the baseline for comparing with our HD classifier.

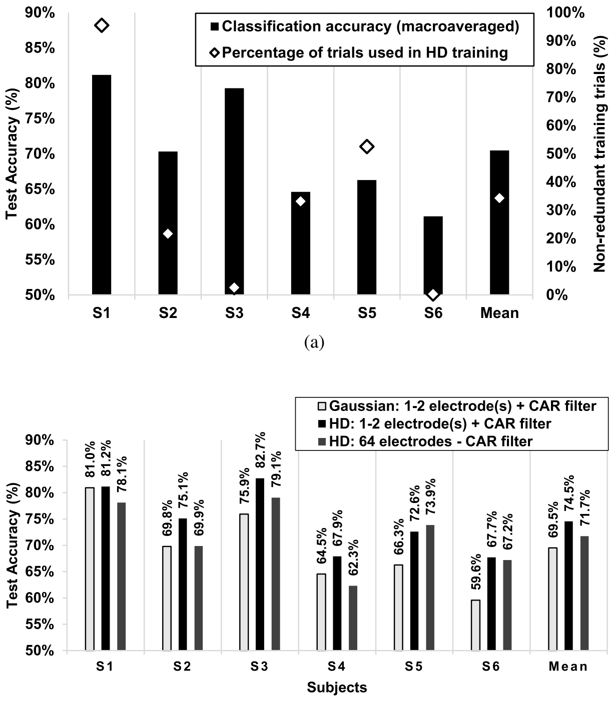
The EEG signals are recorded at a sampling rate of 512 Hz using 64 electrodes according to the standard 10/20 international system. For the preprocessing, the signals are spatially filtered using common average reference (CAR) [53]. By applying the CAR filter to an electrode, the average signal level of the entire electrode array is subtracted from that of the electrode of interest. If the entire head is covered by equally spaced electrodes and the potential on the head is generated by point sources, the CAR results in a spatial voltage distribution with a mean of zero [71]. We will demonstrate later that this spatial filter can be eliminated from the preprocessing with negligible effect on our classification accuracy as the HD classifier can work on raw data. Then, a 1–10-Hz band-pass filter (BPF) is applied to remove the unwanted frequency components. For every subject, a time window corresponding to the erroneous and the correct cursor movements is extracted for further analysis and classification.

As the state of the art, a Gaussian statistical classifier is used for binary classification of a single trial [34]. The Gaussian classifier estimates the posterior probability of a given trial corresponding to one of the two classes. Following domain expert knowledge [47], specific electrodes (FCz, Cz, or both, based on the sensitivity of subjects) are chosen to be used as the inputs to the classifier. The classifier parameters are then tuned using a stochastic gradient descent on the mean square error [46]. Our aim is to replace the aforementioned baseline preprocessing and classification by an efficient and fast HD classifier that enables a natural operation with all the 64 electrodes, and with less training and preprocessed data. For this task, the HD classifier is configured with the IM and the CIM for mapping, the temporal–spatial encoder [Fig 4(b)], and two prototypes in the AM for the two classes ($P1$ for the correct and $P2$ for the error) as summarized in Table 1.

1) Fast Learning: We assess how fast the training of HD classifier can be done while maintaining a classification accuracy as high as the baseline. We have observed that only some of the training trials can produce a nonredundant HD vector to be added to the class prototype [18]. Hence, during the training session, every time a new nonredundant trial is encountered, the AM is updated and the classification accuracy is measured for the entire test set. For the very first trials, the AM is almost empty, but as new trials are encountered it will be lightly populated leading to an increase in the accuracy.

We target the classification accuracy of the baseline that is achieved by using all available trials in the training session, working with the one or two selected electrode(s), and with the CAR preprocessing method. We provide the same setup for the HD classifier, but with fewer training trials, to assess how fast the target baseline accuracy can be reached. As shown in Fig. 7(a), the HD classifier is able to learn faster with some variation across subjects reflecting the significant individual variability: it requires only 0.3% of the nonredundant training trials for S6, and up to 96% for S1. On average, across all the subjects, the HD classifier reaches the target baseline classification accuracy of 70.5% when trained with only 34% of nonredundant training trials. This translates directly to ≈ 3× faster learning.

2) No Electrode Selection and Less Preprocessing: We assess the ability of the HD classifier to operate with noisy inputs and electrodes that do not carry meaningful information. Fig 7(b) compares the classification accuracy of the baseline method with two instances of our HD classifier. The first one has a setup equivalent of the baseline as aforesaid: uses one or two electrode(s) depending on the subjects, applies the CAR preprocessing filter on every electrode before the BPF step, and uses all training trials. As shown in Fig 7(b), this instance of the HD classifier surpasses the baseline accuracy across the six subjects. The HD classifier exhibits 67.7%–82.7% classification accuracy, with an average of 74.5%, which is 5% higher than the baseline with the same conditions.



(a)

(b)

Fig. 7. ***Comparison of the HD classifier with the baseline Gaussian classifier in the EEG ERP task. (a) The HD classifier on average learns ≈3x faster while meeting the target accuracy of the baseline Gaussian classifier (70.5%). (b) Accuracy of the baseline Gaussian classifier vs two instances of the HD classifier: (1) Using the selected electrodes and the CAR preprocessing as in the baseline; (2) Using all the 64 electrodes without the CAR preprocessing.***

The second instance of the HD classifier operates with all the 64 electrodes and without the CAR preprocessing filter. There is no CAR filter in the chain of preprocessing: every electrode signal is immediately passed through a BPF followed by the scaling and quantization step before mapping to the HD space by the CIM. Note that the simple BPF cannot be removed since the EEG ERPs are in the frequency range of 1–10 Hz.

Despite using the 64 electrodes without any electrode selection and no CAR filtering, the HD classifier maintains almost the same range of classification accuracy (i.e., 62.3%–79.1%) across the six subjects as shown in Fig. 7(b). This HD classifier shows on average 2.2% higher classification accuracy compared to the baseline. Note that the HD classifier achieves this by naturally using largely meaningless electrodes regardless of the subjects, while the baseline carefully selects a subset of electrodes per individual subject that can provide meaningful information for the Gaussian classifier. This also confirms the amenability of HD classifier to operate with less preprocessed data.

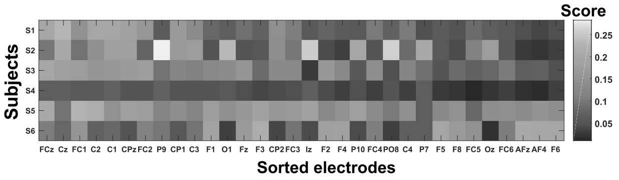


Fig. 8. ***Analyzability of the learned HD code that identifies the most useful electrodes (FCz and Cz). The x-axis shows the sorted electrodes (the top 32 out of 64) based on the average score across the six subjects.***

In HD computing, the input data are naturally clustered in the HD space, and the noise generated by meaningless electrodes tends to cancel out. This desirable property makes it possible to apply HD computing for clustering data with minimal knowledge about the nature of the data.

3) Learning Transparent Codes With Interpretable Features: Apart from the excellent performance of HD classifier with all the 64 electrodes, its learned code is transparent and can be analyzed to find out the important features related to the ERP task. More specifically, rather than asking for information from the domain expert, the learned HD vectors can be used to identify what electrodes provide meaningful data for the classification. Using the domain expert knowledge, Chavarriaga and Millán [34] identify FCz and Cz, or both electrodes as the most useful electrodes for their baseline classifier. We observe that the same subset of electrodes can be identified as useful by the following HD algorithm.

The algorithm is inspired by the distribution of distances in the HD space. For each electrode, we compute a score that measures the distance between two class prototypes that are generated solely by the electrode. This is supplied by first doing the temporal encoding in Fig. 4(b); note that in this encoder if two electrodes i and j receive an identical input stimuli, their encoded n-gram HD vectors become identical ($T_i = T_j$). Before computing the scores for electrodes, we need to compute a set of HD vectors $P1_i$ where $i \in \{1, \ldots, 64\}$. $P1_i$ is computed for every electrode i by adding its related n-gram HD vector (T_i) over all the trials belonging to the correct class. Similarly, $P2_i$ HD vectors are computed from the trials related to the error class

$$P1_i \mathrel{+}= T_i \quad \forall \text{ Correct trial}$$
$$P2_i \mathrel{+}= T_i \quad \forall \text{ Error trial.}$$

For every electrode i, we can then assign a score by measuring the distance between their $P1_i$ and $P2_i$

$$\text{score}_i = 1 - \cos(P1_i, P2_i).$$

In other words, this score reflects how well a given electrode discriminates between the two class prototypes: the larger, the better.

Fig. 8 shows the computed scores for each electrode and across the subjects. The electrodes are sorted in the x-axis according to their average score over the subjects; only the top 32 electrodes out of 64 are shown. As shown, the FCz and Cz electrodes are on top of the sorted list and have the highest discriminative scores for the six subjects, on average. However, all subjects do not exhibit the same sensitivity to these two electrodes. For example, S4 does not show a clear distinction between electrodes.

C. ECoG-Based Seizure Detection

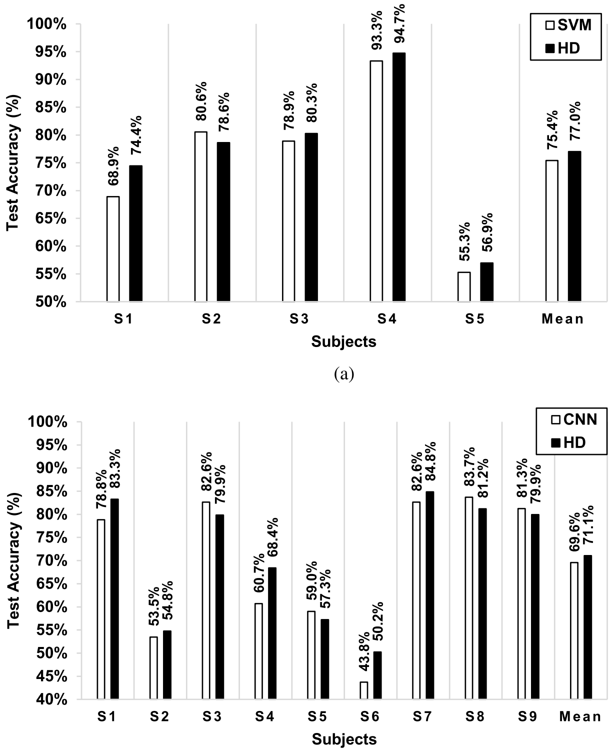
We consider an anonymized data set of 16 patients of the epilepsy surgery program of the Inselspital Bern for a total of 99 recordings. Each recording consists of 3 min of interictal segments (immediately preceding the seizure), and the ictal segment (ranging from 10 to 1002 s), followed by 3 min of postictal time; see [19] for more details. Two recent state-of-the-art methods use local pattern transformation [33] for seizure detection: 1) a method uses histograms of LBPs (2^l integer features per electrode) that performs best with a linear SVM classifier; and 2) akin to LBP, a local gradient pattern (LGP) is further proposed that with an MLP neural network outperforms LBP+SVM. We compare the performance of our HD classifier [see Fig. 4(c) and Table 1] with the LBP + SVM and the LGP + MLP methods by measuring specificity and sensitivity using a few seizures for training.

For the majority of the patients (ten out of 16), our HD classifier quickly learns from one or two seizures, and achieves perfect (100%) specificity and sensitivity with k-fold cross validation, where k is the total number of seizures minus the number of trained seizures. For the remaining minority of six patients, our HD classifier requires more seizures (three to six) for training. For these patients, we use 22 seizures for training and test with the remaining unseen 38 seizures. The HD classifier almost maintains its top performance with 100% sensitivity for five of six patients. In an identical setup, our HD classifier, on average, achieves higher specificity and sensitivity than the other methods. Moreover, the low specificity of LBP+SVM and LGB+MLP clearly limits their usefulness for long-time recordings [19].

D. Multiclass Classification of MI EEG

To increase the complexity of classification task, we move to multiclass brain–computer interfaces based on the MI recordings. The classifiers for the MI-based recordings face particular challenges to operate with complicated frequency-domain features, and with few training trials (≈15 per class per run) since the subjects quickly become exhausted. We first consider a data set [49] with five subjects that are asked to imagine three tasks: imagination of left hand, or right hand, or feet movements. Every subject participates in four runs, each with 45 trials. The MI-based brain–computer interfaces use the power of EEG oscillations in different frequency bands to decode the subject's intention. Hence, the power spectral densities (PSDs) are extracted as features for the classification. They are extracted for the frequency bands of 4–48 Hz from 4 s of the MI command recorded from 16 electrodes. After a normalization and scaling step, the real-valued features are sorted based on a Fisher scoring algorithm as shown in Fig. 4(d). Fisher score, as a filter-based approach, assesses the correlations between features and the class labels to find out features that are efficient for discrimination [72]. It assigns the highest score to the feature on which the data points of different classes are far from each other while requiring data points of the same class to be close to each other. The nine highest ranking features are selected to serve as inputs to the classifier.

For the baseline classification, we first use a Gaussian classifier [49]. However, the Gaussian classifier fails to achieve high accuracy with simultaneous classification into three classes, hence we instead choose the SVM [32] with parameters optimized for larger margin and regularization. We reuse the same feature extractor for the HD classifier which is configured with the weighting method for mapping the nine features, the spatial encoder [Fig. 4(d)], and three prototypes in the AM (one for each class), as summarized in Table 1. To evaluate the performance with fewer training trials, one run (out of four) is used for the training, one for the evaluation (i.e., model selection), and two for the testing. Fig. 9(a) compares the average test accuracy of the HD classifier versus the SVM measured through fourfold cross validation with two folds for testing. The HD classifier shows 53%–98% accuracy across all the subjects (77% on average). Compared to the optimized SVM, the HD classifier improves the minimum, maximum, and average accuracy by 6%, 4%, and 2%, respectively. These accuracy benefits are achieved by a simpler algorithm that is trained in a single pass over the training data.



(a)

(b)

Fig. 9. *Comparison of the HD classifier with the SVM and CNN in two MI tasks. (a) The MI EEG task with three classes. (b) The MI EEG task with four classes.*

1) MI Classification With Four Classes: Finally, we consider another data set for the MI-based brain–computer interfaces: the BCI competition IV-2a [73]. This challenging data set contains nine subjects, with four classes (right hand, left hand, feet, and tongue imaginations) recorded from 22 EEG electrodes. It has two separate sessions for train and test each with 48 trials. For the preprocessing and feature extraction, a filter bank (nine filters from 4–40 Hz) with common spatial pattern (CSP) is used [48]. The CSP is a linear transformation that projects the data into a space where data variance is maximized for one class relatively to another one. In addition to these static energy features, dynamic energy features are computed along with a CNN to improve classification accuracy [35]. The CNN surpasses the SVM and achieves an average classification accuracy of 69.6% across the nine subjects as the state of the art for this data set [35].

For the HD classifier, we scale the same architecture used in the previous MI task [Fig. 4(d)], by increasing the number of input features and the class prototypes (Table 1, the forth row). As shown in Fig. 9(b), the HD classifier achieves a higher classification accuracy compared to the CNN (71.1% versus 69.6%). This confirms the superiority of the proposed HD templates and their scalability to handle complicated tasks with a larger number of features and classes. Nevertheless, this HD classifier uses the spatial encoder with the weighing method that generates an HD vector with real-valued components requiring floating-point hardware and storage. To reduce the hardware complexity of the classifier, we substitute its weighting method with a pair of IM and CIM similar to the spatial encoder in Section IV-B1. We evaluate the linear and nonlinear quantization and similarity-preserving techniques in the CIM, and find out that a CIM hard coded by 100 levels $(m = 100)$ with logarithmically changing the similarity is a good match with the features. The configuration of the HD classifier is shown in the last row in Table 1. The HD classifier maps the real-valued input features to the bipolar HD vectors, as opposed to the HD vectors with real components, and performs the classification with 1% average accuracy loss (70.1% versus 71.1%).

VI. DISCUSSION

There are good reasons to prefer to use vectors as a means of representing items in memory: vector representation allows items to be treated as complex entities, and also allows for fuzzy composites of items to be constructed; furthermore, vectors are amenable to implementation in neural models [8], [11], [13], [40]. In HD computing, or VSAs, the fixed-size HD vectors represent symbolic information. These symbols, or HD vectors, can be combined using a small set of arithmetic operations (e.g., MAP). At the symbolic level, we should choose how to map items to the HD vectors, and how to combine them to create more complex representations. These choices greatly influence the performance. Text and language applications are well matched to this computing framework because the data already come in the form of symbolic primitives (letters or words), which are readily mapped to HD vectors. However, it is challenging for other types of data such as time series from multiple sensors.

To address this issue, we provide a small set of network templates that easily map analog multisensory inputs to HD vectors and construct a complete representational architecture by careful use of MAP operations. These initial solutions can be improved in different aspects. Examples includes the use of thermometer codes (see part I of the Appendix), or locality-sensitive hashing (LSH) for mapping continuous quantities into the HD vectors. The distance-preserving bit sampling LSH can convert $\ell1$-norm to Hamming distance with successful applications in arterial blood pressure time series [74], [75]. All these methods—without learning—come under umbrella of random projection (see [76] for a review). Nevertheless, a hybrid approach combining deep learning and HD computing can be taken where deep learning (either supervised or unsupervised) is used to learn natural features of the data that allow for its mapping into an HD vector. These vectors may then be combined and manipulated within the HD framework for high-level reasoning tasks.

VII. CONCLUSION AND FUTURE WORK

In this paper, we propose a full set of HD network templates for multiclass learning and inference in various biosignal processing applications. These templates facilitate designing versatile, fast, robust, and extremely energy-efficient classifiers by solely using simple native operations of brain-inspired HD computing without involving any biologically implausible or inefficient algorithms. Experimental results with the EMG-based hand gesture recognition, the EEG-based brain–computer interfaces (both ERP and MI), and the ECoG-based seizure detection demonstrate that the HD classifier usually reaches higher classification accuracy (or, at least equal) compared to the state-of-the-art counterpart. More importantly, this is accomplished by little or no prior knowledge about the task: 1) the HD classifier demands much less training data thanks to its simple and one-shot learning; 2) it also naturally operates with noisy and less preprocessed inputs; and 3) there is no need for domain expert knowledge or electrode selection process. Last but not least, the produced HD code is analyzable and interpretable.

Future work is focused on efficient hardware implementation of HD computing for brain–computer interfaces, epileptic seizure onset detection, and identification of ictogenic brain regions. ■

APPENDIX
HYPERDIMENSIONAL COMPUTING CONCEPTS

A. Hyperdimensional Space

Dimensionality $d = 10\,000$ is high dimensional; 10 or 100 are not. Small demonstrations can be made with $d = 1000$, and even very large tasks (e.g., modeling of networks with billions of nodes) can be managed with d less than 100 000.

High-dimensionality together with operations of the right kind are more important than the nature of the dimensions. Operations and properties that have proven useful are listed below.

B. Elements/Points of the Space

- HD vectors or, more generally, the elements of a space of (vector-like) points.
- Similarity metric: based on distance, dot product, cosine, correlation.
- Orthogonality: Randomly chosen vectors are dissimilar, unrelated, uncorrelated, quasi-orthogonal. Most of the space is dissimilar—nearly orthogonal—to any given point. The number of mutually dissimilar vectors far exceeds dimensionality, and finding one more such vector is easy.

C. Operations

Note: The terms "addition," "multiplication," and "permutation" are meant to be understood in a more general (modern/abstract algebra) sense.

- Addition is an operation on two or more vectors that yields a vector.
- Multiplication is an operation on two or more vectors that yields a vector.
- Permutation is a unary operation on a vector that yields a vector.
  - — The number of possible permutations is very large $(d!)$, but permutations themselves are not elements of the space of representations.
  - — Permutation is an example of a more general unary operation on vectors, namely, multiplication by a matrix. However, reducing it to a circular shift operation reduces the complexity of permutation as well as its inverse to $\mathcal{O}(d)$ rather than $\mathcal{O}(d^2)$.
- Normalization converts an intermediate results of an operation into an element of the space over which the operations are defined. For example, if the elements of the space are binary vectors, the arithmetic sum–vector of two or more vectors has to be normalized by a threshold function to make it binary.
- Scalar product or dot product provides a measure of similarity between two vectors.

D. Properties

Note: The properties refer to (pseudo)random vectors with i.i.d. components. Thanks to high dimensionality, the conditions listed below need only be satisfied approximately or with high probability. Note also that the algebra of addition and multiplication approximates a field over the vector space.

- Multiplication and permutation are invertible.
- Multiplication distributes over addition.
- Permutation distributes over both addition and multiplication.
- The sum vector is similar to each of its argument vectors.
- The product vector is dissimilar to each of its argument vectors.
- The result of a (random) permutation is dissimilar to the argument vector.
- Multiplication and permutation are "randomizing" operations that preserve similarity.
- Addition and multiplication are associative.
- Addition is commutative.

By the Law of Large Numbers, the reliability/predictability of the computations is directly related to vector dimensionality.

E. Examples of HD Spaces and Operations

- Real vectors: Holographic reduced representation (HRR) was the first among these systems. It uses d-dimensional real vectors whose components are i.i.d. normal with zero mean and $1/d$ variance. Addition is by normalized vector sum, and multiplication is by circular convolution.
- Complex vectors: Vector components are random phase angles, addition is by componentwise complex addition followed by normalization, and multiplication is by componentwise complex multiplication (addition of phase angles).
- 50–50 binary vectors: Addition is by componentwise majority rule followed by tie breaking, and multiplication is by componentwise XOR.
- Bipolar (± 1) vectors: The MAP architecture uses componentwise addition and multiplication, followed by normalization and is equivalent to the binary.

An operation can have a property that is useful in some contexts but needs a workaround in others. An example of such is the self-inverse property of multiplication of binary vectors with componentwise XOR. It may work well for undirected graphs but poorly for directed graphs.

F. Mapping With Vectors

A key notion of HD computing is that a vector can represent a mapping and that a mapping vector can be computed from examples in a single pass using the vector operations. Similarly, vectors for composed entities such as a network, are computed from vectors for the constituents (i.e., the nodes) with the HD operations in a single pass. This is much like traditional computing and very different from standard neural nets that compute mappings with gradient descent (backpropagation) in multiple passes over a set of examples. To apply a vector map to another vector, we simply multiply with the mapping vector and possibly follow it with a memory retrieval.

G. HD Memory

The long-term memory function of standard neural nets is encoded into—and is confounded with—the same set of weights that perform mappings between vectors. In HD computing, the two are separate. The memory corresponds to a computer RAM and it stores vectors made with the HD vector operations. Memory retrieval means finding the best matching vector (or vectors, i.e., nearest neighbors) in the set of vectors stored in the memory, which also yields a measure of confidence in the retrieved vectors.

H. Generality

The HD operations and memory are sufficient for general computing. For example, the Lisp programming language could be implemented with them.

Some computing operations that are native to HD have no simple counterpart in traditional computing. They include vectors as mappings and as semantic pointers. However, operating with numbers is awkward and inefficient. Thus traditional computing can be deemed quantitative and HD computing qualitative.

I. Theory on Coding Continuous Values in HD Vectors

Using the 50–50 binary vectors (binary spatter codes) we describe a method to encode graded values (quantities) into binary HD vectors via the thermometer code (unary coding). A thermometer code has a number of 1s to represent the quantity, followed by 0s:

$$111\ldots111000\ldots000.$$

Assume that x in the range $[0,1]$ is represented by a d-bit thermometer code, with the number of 1s proportional to x. If we want the ends of the range to have orthogonal HD vectors, the thermometer code for 0 has no 1s and the thermometer code for 1 has $d/2$ 1s. For example, 1000 1s followed by 9000 0s represents 0.2.

Take the thermometer code for x, $T(x)$, multiply (XOR) it with a random label L, and permute the result with a random permutation (not rotate) ρ. Then, the value x is encoded by the HD vector

$$X = \rho(L * T(x)).$$

Multiplying by L makes temperature look random, and permuting with ρ scrambles the coordinates. We can then read the thermometer by counting the number of 1s in $L * (qX)$, where q is the inverse permutation of ρ and L is its own inverse. We can do the same for the y coordinate but need a random label L' and a random permutation ρ' that are unique to the y-axis. The position (x, y) can then be labeled with $X * Y$. If we are given x and the vector $X * Y$, we can compute y.

The coding maintains temperature differences because neither XOR nor permutation affects Hamming distance h (both operations preserve similarity)

$$\begin{aligned} h(X1, X2) &= h(\rho(L * T(x1)), \rho(L * T(x2))) \\ &= h(L * T(x1), L * T(x2)) \\ &= h(T(x1), T(x2)). \end{aligned}$$

REFERENCES

[1] A. Rahimi, L. Benini, and R. K. Gupta, "Variability mitigation in nanometer CMOS integrated systems: A survey of techniques from circuits to software," *Proc. IEEE*, vol. 104, no. 7, pp. 1410–1448, Jul. 2016.

[2] J. M. Rabaey, "A roadmap to lower supply voltages—A system perspective," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2015.

[3] A. Rahimi, L. Benini, and R. K. Gupta, *From Variability Tolerance to Approximate Computing in Parallel Integrated Architectures and Accelerators*. New York, NY, USA: Springer-Verlag, 2017.

[4] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "Envision: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28 nm FDSOI," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2017, pp. 246–247.

[5] A. Rahimi *et al.*, "High-dimensional computing as a nanoscalable paradigm," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 9, pp. 2508–2521, Sep. 2017.

[6] T. F. Wu *et al.*, "Brain-inspired computing exploiting carbon nanotube FETs and resistive RAM: Hyperdimensional computing case study," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 492–494.

[7] H. Li *et al.*, "Hyperdimensional computing with 3D VRRAM in-memory kernels: Device-architecture co-design for energy-efficient, error-resilient language recognition," in *IEDM Tech. Dig.*, Dec. 2016, pp. 16.1.1–16.1.4.

[8] P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cogn. Comput.*, vol. 1, no. 2, pp. 139–159, Oct. 2009, doi: 10.1007/s12559-009-9009-8.

[9] P. Kanerva, *Sparse Distributed Memory*. Cambridge, MA, USA: MIT Press, 1988.

[10] P. Kanerva, "Computing with 10,000-bit words," in *Proc. 52nd Annu. Allerton Conf. Commun. Control Comput.*, Sep. 2014, pp. 304–310.

[11] R. W. Gayler, "Vector symbolic architectures answer Jackendoff's challenges for cognitive neuroscience," in *Proc. Joint Int. Conf. Cogn. Sci. (ICCS/ASCS)*, 2003, pp. 133–138.

[12] B. Emruli, R. W. Gayler, and F. Sandin, "Analogical mapping and inference with binary spatter codes and sparse distributed memory," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Aug. 2013, pp. 1–8.

[13] M. A. Kelly and R. L. West, "A framework for computational models of human memory," in *Proc. AAAI Fall Symp.*, pp. 376–381, 2017.

[14] A. Rahimi, S. Benatti, P. Kanerva, L. Benini, and J. M. Rabaey, "Hyperdimensional biosignal processing: A case study for EMG-based hand gesture recognition," in *Proc. IEEE Int. Conf. Rebooting Comput.*, Oct. 2016, pp. 1–8.

[15] A. Moin *et al.*, "An EMG gesture recognition system with flexible high-density sensors and brain-inspired high-dimensional classifier," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–5.

[16] F. Montagna, A. Rahimi, S. Benatti, D. Rossi, and L. Benini, "PULP-HD: Accelerating brain-inspired high-dimensional computing on a parallel ultra-low power platform," in *Proc. 55th Annu. Design Automat. Conf. (DAC)*, 2018, pp. 111:1–111:6. [Online]. Available: http://doi.acm.org/10.1145/3195970.3196096

[17] A. Rahimi, P. Kanerva, J. D. R. Millán, and J. M. Rabaey, "Hyperdimensional computing for noninvasive brain-computer interfaces: Blind and one-shot classification of EEG error-related potentials," in *Proc. 10th ACM/EAI Int. Conf. Bio-Inspired Inf. Commun. Technol. (BICT)*, 2017, pp. 1–8.

[18] A. Rahimi, A. Tchouprina, P. Kanerva, J. D. R. Millán, and J. M. Rabaey, "Hyperdimensional computing for blind and one-shot classification of EEG error-related potentials," *Mobile Netw. Appl.*, pp. 1–12, Oct. 2017, doi: 10.1007/s11036-017-0942-6.

[19] A. Burrello, K. Schindler, L. Benini, and A. Rahimi, "One-shot learning for iEEG seizure detection using end-to-end binary operations: Local binary patterns with hyperdimensional computing," in *Proc. IEEE Biomed. Circuits Syst. Conf. (BioCAS)*, 2018, pp. 1–4.

[20] P. Kanerva, J. Kristoferson, and A. Holst, "Random indexing of text samples for latent semantic analysis," in *Proc. 22nd Annu. Conf. Cogn. Sci. Soc.*, 2000, p. 1036. [Online]. Available: http://www.rni.org/kanerva/cogsci2k-poster.txt

[21] P. Kanerva, "Binary spatter-coding of ordered k-tuples," in *Proc. Int. Conf. Artif. Neural Netw. (ICANN)*, in Lecture Notes in Computer Science, vol. 1112, 1996, pp. 869–873.

[22] P. Kanerva, "What we mean when we say 'What's the Dollar of Mexico?': Prototypes and mapping in concept space," in *Proc. AAAI Fall Symp. Quantum Inf. Cogn. Social Semantic Process.*, 2010, pp. 2–6.

[23] A. Joshi, J. T. Halseth, and P. Kanerva, "Language geometry using random indexing," in *Proc. 10th Int. Conf. Quantum Interact. (QI)*, J. A. de Barros, B. Coecke, and E. Pothos, Eds. Cham, Switzerland: Springer-Verlag, 2017, pp. 265–274, doi: 10.1007/978-3-319-52289-0_21.

[24] A. Rahimi, P. Kanerva, and J. M. Rabaey, "A robust and energy-efficient classifier using brain-inspired hyperdimensional computing," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, Aug. 2016, pp. 64–69.

[25] F. R. Najafabadi, A. Rahimi, P. Kanerva, and J. M. Rabaey, "Hyperdimensional computing for text classification," in *Proc. Design Automat. Test Eur. Conf. Exhib. (DATE)*, 2016. [Online]. Available: https://www.date-conference.com/system/files/file/date16/ubooth/37923.pdf

[26] M. Imani, T. Nassar, A. Rahimi, and T. Rosing, "HDNA: Energy-efficient DNA sequencing using hyperdimensional computing," in *Proc. IEEE EMBS Int. Conf. Biomed. Health Inf. (BHI)*, Mar. 2018, pp. 271–274.

[27] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[28] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 54, A. Singh and J. Zhu, Eds. Fort Lauderdale, FL, USA: PMLR, Apr. 2017, pp. 1273–1282. [Online]. Available: http://proceedings.mlr.press/v54/mcmahan17a.html

[29] W. House, "Consumer data privacy in a networked world: A framework for protecting privacy and promoting innovation in the global digital economy," *J. Privacy Confidentiality*, 2013.

[30] B. Blankertz *et al.*, "Neurophysiological predictor of SMR-based BCI performance," *NeuroImage*, vol. 51, no. 4, pp. 1303–1309, 2010.

[31] S. Benatti *et al.*, "A versatile embedded platform for EMG acquisition and gesture recognition," *IEEE Trans. Biomed. Circuits Syst.*, vol. 9, no. 5, pp. 620–630, Oct. 2015.

[32] D. Wang, D. Miao, and G. Blohm, "Multi-class motor imagery EEG decoding for brain-computer interfaces," *Frontiers Neurosci.*, vol. 6, p. 151, Oct. 2012. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2012.00151

[33] A. K. Jaiswal and H. Banka, "Local pattern transformation based feature extraction techniques for classification of epileptic EEG signals," *Biomed. Signal Process. Control*, vol. 34, pp. 81–92, Apr. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S174680941730006X

[34] R. Chavarriaga and J. D. R. Millán, "Learning from EEG error-related potentials in noninvasive brain-computer interfaces," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 18, no. 4, pp. 381–388, Aug. 2010.

[35] S. Sakhavi, C. Guan, and S. Yan, "Parallel convolutional-linear neural network for motor imagery classification," in *Proc. 23rd Eur. Signal Process. Conf. (EUSIPCO)*, Aug. 2015, pp. 2736–2740.

[36] O. Räsänen, "Generating hyperdimensional distributed representations from continuous-valued multivariate sensory input," in *Proc. 37th Annu. Meeting Cogn. Sci. Soc.*, 2015, pp. 1943–1948.

[37] M. Imani, D. Kong, A. Rahimi, and T. Rosing, "VoiceHD: Hyperdimensional computing for efficient speech recognition," in *Proc. IEEE Int. Conf. Rebooting Comput. (ICRC)*, Nov. 2017, pp. 1–8.

[38] P.-C. Huang and J. M. Rabaey, "A bio-inspired analog gas sensing front end," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 9, pp. 2611–2623, Sep. 2017.

[39] T. A. Plate, *Holographic Reduced Representations*. Stanford, CA, USA: CLSI Publications, 2003.

[40] C. Eliasmith, *How to Build a Brain: A Neural Architecture for Biological Cognition*, ser. Oxford Series on Cognitive Models and Architectures. Oxford, U.K.: Oxford Univ. Press, 2013.

[41] R. W. Gayler, "Multiplicative binding, representation operators & analogy," in *Advances in Analogy Research: Integration of Theory and Data from the Cognitive, Computational and Neural Sciences*, D. Gentner, K. J. Holyoak, and B. N. Kokinov, Eds. Sofia, Bulgaria: New Bulgarian Univ., 1998, pp. 1–4. [Online]. Available: http://cogprints.org/502/

[42] E. P. Frady, D. Kleyko, and F. T. Sommer, "A theory of sequence indexing and working memory in recurrent neural networks," *Neural Comput.*, vol. 30, no. 6, pp. 1449–1513, 2018, doi: 10.1162/neco_a_01084.

[43] M. R. Ahsan, M. Ibrahimy, and O. Khalifa, "EMG signal classification for human computer interaction: A review," *Eur. J. Sci. Res.*, vol. 33, no. 3, pp. 480–501, 2009.

[44] J. Rosen, M. Brand, M. B. Fuchs, and M. Arcan, "A myosignal-based powered exoskeleton system," *IEEE Trans. Syst. Man Cybern. A, Syst. Humans*, vol. 31, no. 3, pp. 210–222, May 2001.

[45] D. Yang, L. Jiang, Q. Huang, R. Liu, and H. Liu, "Experimental study of an EMG-controlled 5-DOF

anthropomorphic prosthetic hand for motion restoration," *J. Intell. Robot. Syst.*, vol. 76, nos. 3–4, pp. 427–441, Dec. 2014, doi: 10.1007/s10846-014-0037-6.

[46] P. W. Ferrez and J. D. R. Millán, "You are wrong!—Automatic detection of interaction errors from brain waves," in *Proc. 19th Int. Joint Conf. Artif. Intell.*, 2005, pp. 1–6.

[47] P. Ferrez and J. D. R. Millán, "Error-related EEG potentials in brain-computer interfaces," Ph.D. dissertation, STI, Lausanne, Switzerland, 2007.

[48] K. K. Ang, Z. Y. Chin, C. Wang, C. Guan, and H. Zhang, "Filter bank common spatial pattern algorithm on BCI competition IV datasets 2a and 2b," *Frontiers Neurosci.*, vol. 6, no. 1, p. 39, 2012. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2012.00039

[49] S. Saeedi, R. Chavarriaga, R. Leeb, and J. D. R. Millán, "Adaptive assistance for brain-computer interfaces by online prediction of command reliability," *IEEE Comput. Intell. Mag.*, vol. 11, no. 1, pp. 32–39, Feb. 2016.

[50] D. Schmidt and M. Sillanpää, "Evidence-based review on the natural history of the epilepsies," *Current Opinion Neurol.*, vol. 25, no. 2, pp. 159–163, 2012.

[51] Y. Li, K. Dzirasa, L. Carin, and D. E. Carlson, "Targeting EEG/LFP synchrony with neural nets," in *Advances in Neural Information Processing Systems*, I. Guyon *et al.*, Eds. Red Hook, NY, USA: Curran Associates, 2017, pp. 4623–4633. [Online]. Available: http://papers.nips.cc/paper/7048-targeting-eeglfp-synchrony-with-neural-nets.pdf

[52] E. Osipov, D. Kleyko, and A. Legalov, "Associative synthesis of finite state automata model of a controlled object with hyperdimensional computing," in *Proc. 43rd Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Oct. 2017, pp. 3276–3281.

[53] D. J. McFarland, L. M. McCane, S. V. David, and J. R. Wolpaw, "Spatial filter selection for EEG-based communication," *Electroencephalogr. Clin. Neurophysiol.*, vol. 103, no. 3, pp. 386–394, Sep. 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0013469497000222

[54] F. Montagna, S. Benatti, and D. Rossi, "Flexible, scalable and energy efficient bio-signals processing on the pulp platform: A case study on seizure detection," *J. Low Power Electron. Appl.*, vol. 7, no. 2, p. 16, 2017. [Online]. Available: http://www.mdpi.com/2079-9268/7/2/16

[55] A. Litwin-Kumar, K. D. Harris, R. Axel, H. Sompolinsky, and L. F. Abbott, "Optimal degrees of synaptic connectivity," *Neuron*, vol. 93, no. 5, pp. 1153–1164, Mar. 2017, doi: 10.1016/j.neuron.2017.01.030.

[56] D. A. Rachkovskij, "Representation and processing of structures with binary sparse distributed codes," *IEEE Trans. Knowl. Data Eng.*, vol. 13, no. 2, pp. 261–276, Mar. 2001.

[57] O. J. Räsänen and J. P. Saarinen, "Sequence prediction with sparse distributed hyperdimensional coding applied to the analysis of mobile phone use patterns," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 9, pp. 1878–1889, Sep. 2016.

[58] M. Laiho, J. H. Poikonen, P. Kanerva, and E. Lehtonen, "High-dimensional computing with sparse vectors," in *Proc. IEEE Biomed. Circuits Syst. Conf. (BioCAS)*, Oct. 2015, pp. 1–4.

[59] D. Kleyko, A. Rahimi, D. A. Rachkovskij, E. Osipov, and J. M. Rabaey, "Classification and recall with binary hyperdimensional computing: Tradeoffs in choice of density and mapping characteristics," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published.

[60] M. Imani, J. Hwang, T. Rosing, A. Rahimi, and J. M. Rabaey, "Low-power sparse hyperdimensional encoder for language recognition," *IEEE Design Test*, vol. 34, no. 6, pp. 94–101, Dec. 2017.

[61] C. S. Daw, C. E. A. Finney, and E. R. Tracy, "A review of symbolic analysis of experimental data," *Rev. Sci. Instrum.*, vol. 74, no. 2, pp. 915–930, 2003, doi: 10.1063/1.1531823.

[62] O. Yilmaz, "Symbolic computation using cellular automata-based hyperdimensional computing," *Neural Comput.*, vol. 27, no. 12, pp. 2661–2692, 2015, doi: 10.1162/NECO_a_00787.

[63] M. Schmuck, L. Benini, and A. Rahimi (Jul. 2018). "Hardware optimizations of dense binary hyperdimensional computing: Rematerialization of hypervectors, binarized bundling, and combinational associative memory." [Online]. Available: https://arxiv.org/abs/1807.08583

[64] D. Kleyko, S. Khan, E. Osipov, and S.-P. Yong, "Modality classification of medical images with distributed representations based on cellular automata reservoir computing," in *Proc. IEEE 14th Int. Symp. Biomed. Imag. (ISBI)*, Apr. 2017, pp. 1053–1056.

[65] D. Widdows and T. Cohen, "Reasoning with vectors: A continuous model for fast robust inference," *Logic J. IGPL*, vol. 23, no. 2, pp. 141–173, 2014.

[66] T. S. Kumar, V. Kanhangad, and R. B. Pachori, "Classification of seizure and seizure-free EEG signals using local binary patterns," *Biomed. Signal Process. Control*, vol. 15, pp. 33–40, Jan. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1746809414001384

[67] K. Schindler, H. Gast, M. Goodfellow, and C. Rummel, "On seeing the trees and the forest: Single-signal and multisignal analysis of periictal intracranial EEG," *Epilepsia*, vol. 53, no. 9, pp. 1658–1668, 2012.

[68] J. Snell, K. Swersky, and R. S. Zemel (Mar. 2017). "*Prototypical networks for few-shot learning*." [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1746809414001384

[69] M. Imani, A. Rahimi, D. Kong, T. Rosing, and J. M. Rabaey, "Exploring hyperdimensional associative memory," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2017, pp. 445–456.

[70] *Monitoring Error-Related Potentials*. [Online]. Available: http://bnci-horizon-2020.eu/database/data-sets

[71] O. Bertrand, F. Perrin, and J. Pernier, "A theoretical justification of the average reference in topographic evoked potential studies," *Electroencephalogr. Clin. Neurophysiol.*, vol. 62, no. 6, pp. 462–464, 1985. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0168559785900589

[72] X. He, D. Cai, and P. Niyogi, "Laplacian score for feature selection," in *Proc. 18th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2005, pp. 507–514. [Online]. Available: http://dl.acm.org/citation.cfm?id=2976248.2976312

[73] *BCI Competition IV-2a (Four Class Motor Imagery)*. [Online]. Available: http://bnci-horizon-2020.eu/database/data-sets

[74] Y. B. Kim and U. M. O'Reilly, "Large-scale physiological waveform retrieval via locality-sensitive hashing," in *Proc. 37th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Aug. 2015, pp. 5829–5833.

[75] Y. B. Kim and U. M. O'Reilly, "Analysis of locality-sensitive hashing for fast critical event prediction on physiological time series," in *Proc. 38th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Aug. 2016, pp. 783–787.

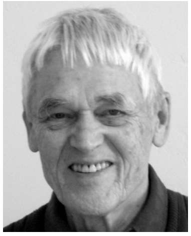
[76] D. A. Rachkovskij, "Binary vectors for fast distance and similarity estimation," *Cybern. Syst. Anal.*, vol. 53, no. 1, pp. 138–156, Jan. 2017, doi: 10.1007/s10559-017-9914-x.

ABOUT THE AUTHORS

Abbas Rahimi (Member IEEE) received the B.S. degree in computer engineering from the University of Tehran, Tehran, Iran, in 2010 and the M.S. and Ph.D. degrees in computer science and engineering from the University of California San Diego, La Jolla, CA, USA, in 2015, followed by two years postdoctoral research in the Department of Electrical Engineering and Computer Sciences at the University of California Berkeley, Berkeley, CA, USA.

He then joined the Department of Information Technology and Electrical Engineering, ETH Zurich, Zurich, Switzerland, in 2017, as a postdoctoral fellow. He is also affiliated with the Berkeley Wireless Research Center, Berkeley, CA, USA. His research interests include brain-inspired computing, approximate computing, massively parallel integrated architectures, embedded systems and software with an emphasis on improving energy efficiency and robustness.

Dr. Rahimi was awarded an ETH Zurich Postdoctoral Fellowship. His doctoral dissertation has received the 2015 Outstanding Dissertation Award in the area of "New Directions in Embedded System Design and Embedded Software" from the European Design and Automation Association (EDAA). He has also received the Best Paper at BICT 2017 and the Best Paper Candidate at DAC 2013.

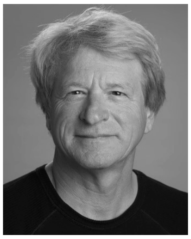


Pentti Kanerva received the Ph.D. degree in philosophy from Stanford University, Stanford, CA, USA.

He worked 20 years designing and building computer systems, followed by 30+ years of research into understanding brains in computing terms. His dissertation was published in the book *Sparse Distributed Memory* by (Cambridge, MA, USA: MIT Press, 2003) and his subsequent research includes binary spatter code, random indexing, and hyperdimensional computing. He has held research positions at NASA Ames Research Center, Swedish Institute of Computer Science, and the Redwood Neuroscience Institute, and is now a researcher at the Redwood Center for Theoretical Neuroscience, University of California Berkeley, Berkeley, CA, USA.

Luca Benini (Fellow IEEE) is a Professor of Digital Circuits and Systems at ETH Zurich, Zurich, Switzerland, and is also a Professor at the University of Bologna, Bologna, Italy. His research interests are in energy-efficient system design and multicore SoC design. He is also active in the area of energy-efficient smart sensors and sensor networks.

Prof. Benini is a Fellow of the Association for Computing Machinery (ACM) and a member of the Academia Europea. He received the IEEE CAS Mac Van Valkenburg Award.



Jan M. Rabaey (Fellow IEEE) holds the Donald O. Pederson Distinguished Professorship at the University of California Berkeley, Berkeley, CA, USA. He is a founding director of the Berkeley Wireless Research Center (BWRC) and the Berkeley Ubiquitous SwarmLab. He has been involved in a broad variety of startup ventures. He has made high-impact contributions to a number of fields, including advanced wireless systems, low power integrated circuits, sensor networks, and ubiquitous computing. His current interests include the conception of the next-generation integrated wireless systems over a broad range of applications, as well as exploring the interaction between the cyber world and the biological world.

Prof. Rabaey is the recipient of major awards, among which is the IEEE Mac Van Valkenburg Award, the European Design Automation Association (EDAA) Lifetime Achievement award, and the Semiconductor Industry Association (SIA) University Researcher Award. He is amember of the Royal Flemish Academy of Sciences and Arts of Belgium.