Online Learning and Classification of EMG-Based Gestures on a Parallel Ultra-Low Power Platform Using Hyperdimensional Computing

Online Learning and Classification of EMG-Based Gestures on a Parallel Ultra-Low Power Platform using Hyperdimensional Computing

Simone Benatti[†§], Fabio Montagna[†§], Victor Kartsch[†], Abbas Rahimi[‡¶] Davide Rossi[†], Luca Benini[†‡],
[†]DEI, University of Bologna, Italy.
[†]Email: {fabio.montagna, victorjavier.kartsch, simone.benatti, davide.rossi, luca.benini}@unibo.it
[¶]Berkeley Wireless Research Center, University of California, Berkeley, CA. USA. Email: {abbas}@berkeley.edu
[‡]Integrated System Laboratory, ETHZ, Zurich, Switzerland. Email: {abbas, lbenini}@iis.ee.ethz.ch

***Abstract*—This work presents a wearable EMG gesture recognition system based on the hyperdimensional (HD) computing paradigm, running on a programmable Parallel Ultra-Low-Power (PULP) platform. The processing chain includes efficient on-chip training, which leads to a fully embedded implementation with no need to perform any offline training on a personal computer. The proposed solution has been tested on 10 subjects in a typical gesture recognition scenario achieving 85% average accuracy on 11 gestures recognition, which is aligned with the State-Of-the-Art (SoA), with the unique capability of performing online learning. Furthermore, by virtue of the Hardware (HW) friendly algorithm and of the efficient PULP System-on-Chip (SoC) (Mr. Wolf) used for prototyping and evaluation, the energy budget required to run the learning part with 11 gestures is 10.04mJ, and 83.2μJ per classification. The system works with a average power consumption of 10.4mW in classification, ensuring around 29h of autonomy with a 100mAh battery. Finally, the scalability of the system is explored by increasing the number of channels (up-to 256 electrodes), demonstrating the suitability of our approach as universal, energy-efficient biopotential wearable recognition framework.**

***Index Terms*—Hyperdimensional computing, embedded systems, PULP platform, HMI, gesture recognition.**

I. Introduction

Since interacting with hands represents one of the most intuitive ways to enable human-to-human or human-to-machine interactions , hand gesture recognition is the key element in designing solutions that can enable natural and advanced ways of communication between objects and users in many domains, in the wake of the IoT growing trend. The two major approaches used for hand gesture recognition are based on processing of information coming from video cameras [1] and from muscular activity [2].

Video-based hand gesture recognition relies on computer vision techniques which recognize users' hand gestures in a scene and decode the hand gestures using pattern recognition algorithms. Although this approach can decode a wide number of gestures, it suffers from ambient illumination variability and from possible obstacles in the line of sight. Furthermore, it requires a bulky setup, based on pre-installed environmental cameras.

Another viable approach is inspired by hand prosthetic systems, where the muscular activity detected by the electromyographic (EMG) signals is used to decode the user's intention. Commercial prosthetic systems [3], [4] decode predefined bursts of muscular contractions into a set of gestures (i.e. 2 consecutive contractions mean hand closures, 3 consecutive contractions stands for hand opening, etc.). Such method is highly reliable and amenable to implement on a wearable system. However, it requires a long learning curve and high levels of concentration of the user, since it is a non-intuitive interface [5].

An approach that is gaining traction is based on machine learning (ML) techniques to analyze EMG signal patterns during muscular contractions. It has been demonstrated that algorithms such as Linear Discriminant Analysis (LDA) [6], Artificial Neural Networks (ANN) [7], Support Vector Machines (SVM) [8], Recursive Least Square [9], Hidden Markov Models (HMM) [10], Naive Bayes [11], Independent Component Analysis (ICA) [12], as well as Convolutional Neural Networks (CNN) [13] reach high level of accuracy, restoring a natural gesture recognition both in prosthetics and consumer Human Machine Interaction (HMI) scenarios.

Using such machine learning algorithms allows to tailor the recognition strategies to the physiological characteristic of the users. However, EMG-based gesture recognition is strongly dependent on the subject (i.e. it depends on subject-specific characteristics such as muscular mass, skin thickness, strength of the mean voluntary contractions), and classification algorithms need a training set for each user. Furthermore, EMG setup is intrinsically variable [14]–[16] because of fiber crosstalk, skin perspiration, small movements of the skin-to-electrode interface, power line interference and donning/doffing. As such, small changes of the EMG traces can hinder pattern recognition, degrading the system performance down to unacceptable levels [17].

An orthogonal approach consists of extending the training dataset, integrating it with samples coming from multiple sessions to gain up to 20% recognition accuracy [16]. A major drawback of increasing the size of the training dataset or repeating the algorithm training lies into its high computational requirements. For instance, SVM training performs minimization of a given cost function by solving a convex optimization problem, while in ANN [18] back-propagation requires many iterations to converge. Furthermore, training sessions require a PC or a graphical interface as well as human intervention to

[§] Simone Benatti and Fabio Montagna have equal contribution

perform thresholding and labeling operations [19].

Developing a system capable of "one-shot" learning based on a computationally efficient and non iterative algorithm for online training has the potential of significantly improving HMIs based on EMG signals. In our previous work [20], we have presented an EMG gesture recognition inference algorithm based on a computing paradigm called hyperdimensional (HD) computing, and its hardware-optimized implementation on an energy-efficient Parallel Ultra-Low-Power (PULP) system on a chip (SoC). This work extends the paper presented in [20], describing the design and optimization of a full ultra-low-power EMG pattern recognition wearable system featuring both inference and online "one-shot" learning. Moreover, we validated the proposed embedded system, composed of the Mr. Wolf PULP SoC [21] coupled with a dedicated bio-potential ADC, acquiring data from 10 subjects and 11 gestures.

The main contributions of this paper are:

- The development of an algorithm suitable for real-time "one-shot" learning. To the best of our knowledge this is the first time that a full EMG-based wearable system with online learning is being proposed.
- Implementation and optimization of the algorithm on a parallel-ultra-low power architecture (i.e. Mr Wolf).
- Design of an integrated wearable interface for gesture recognition embedding the Mr. Wolf SoC, extensively tested on a dataset acquired online on 10 subjects, and 11 gestures.
- The evaluation of the proposed system with respect to other methods for embedded gesture recognition, including a comparison with a well-known state-of-the-art (SoA) method such as SVM [8].
- The exploration of the system scalability with respect to the number of electrodes available, following the general trend of bio-potential pattern recognition systems [22].

We present a complete system, which performs online and on-chip learning, paving the way to the design of devices with a fast training step for a wide range of different hand gestures. Our device is completely free of supplementary bench-top devices, hence it is fully embedded and portable. The proposed system is highly flexible and versatile, being capable of adapting to the changes in the initial setup and other interferences that can deteriorate the performance. Moreover, it can be easily adapted to other kinds of applications, where a higher number of acquisition channels is required, both for EEG [23] and EMG [22] based applications. The system reaches 90.40% accuracy on 11 gestures recognition, comparable to the SoA systems, within an energy budget of just 10.04mJ for the on-chip training and 83.20μJ for the recognition (inference), leading to an average power consumption of 10.04mW reaching 29 hours battery life with a 100mAh battery. Finally, we explore the capability of the proposed system scaling up the number of channels (up to 256 electrodes) and exploring the performance in term of energy consumption, demonstrating the scalability of the approach and the suitability of the system as an efficient bio-potential wearable recognition device.

II. Related Work

In recent years, several systems have become available to acquire and process EMG signals for hand gesture recognition. This task requires a multi-modal approach which involves sensor interfaces, computational platforms and algorithms. The sensor interfaces are mostly based on commercial SoA Analog Front Ends, specifically designed for bio-potential acquisition. Regarding computational platforms and algorithms, top performance classification methods rely on supervised machine learning algorithms to reliably classify acquired data, such as SVM, LDA [24] or ANN [7]. The recognition accuracy of these algorithms is above 85%, and they are implementable on wearable platforms [19]. Despite some other methods, like RLS [9], can solve multiclass problems with negligible computational overhead, deterministic training time, and performance comparable to the aforementioned algorithms, in this work we performed a quantitative comparison with SVM, which represents the SoA algorithm and widely accepted baseline framework for EMG-based pattern recognition, already tested in several embedded implementations [24]–[29].

However, the major drawback common to all these approaches is related to the robustness of the setup, since the EMG signal is intrinsically affected by high variability due to both physiological and environmental factors. Once the training phase is performed on a given dataset, the algorithm does not generalize if signals somehow drift. Therefore, the reliability of such systems would greatly benefit of a fast real-time training, which can be repeated when needed. Unfortunately, in these approaches, the training phase is based on minimization of convex costs functions [30], backpropagation [31] or eigen decomposition [32]. Some of these techniques are iterative, with a convergence time that depends on the number of iterations required to minimize the error cost function. Furthermore, their computational and memory requirements severely hamper their implementation on resource-constrained platforms. For instance, LDA, which has a training faster than SVM and ANN [16], has time complexity of $O(mnt + t^3)$ with a memory footprint of $O(mn+mt+nt)$ memory, where m is the number of samples, n is the number of features and t is defined as $min(m, n)$ [24]. As a result, whereas the number of samples/features grows, the system becomes too resource-hungry to be adapted to a low-power platform. This is a common drawback shared among all these classical approaches, since they need to store all training examples for model computation, and this eventually limits the number of examples that can be considered for learning in a resource-limited embedded system.

Recently, a few works have investigated the use of deep neural networks, such as CNN [13] or combinations of CNN and Recurrent Neural Networks (RNN) [38]. Using complex topology of deep networks, these approaches can recognize up to 50 gestures with accuracy values ($> 80\%$), which is suitable for the design of a real time controller. Unfortunately, the training of deep neural networks requires a huge amount of data to converge and it requires dedicated GPU servers, resulting even more computationally demanding w.r.t. conventional approaches.

TABLE I: Comparison between SoA EMG embedded pattern recognition systems and our platform. Systems in [34] and [35] run in computationally hungry platforms and are not suitable for long-term operation, and, although the systems presented in [33] and [10] have a comparable energy/classification as this work, our system is capable of providing more than 2.7x improvement in battery life.

	Platform Architecture	ISA	Technology [nm]	Max Frequency [MHz]	Algorithm	Online learning	#Channles	#Gestures	Energy[mJ]/ classification	Battery [h]
Benatti [33] :	1-core	Cortex M4	90	168	SVM	no	8	7	0.0891	9.97
Liu [34] :	1-core	Cortex A8	65	720	Muscoskeletal Model	no	8	-	1100.0000	0.27
Zhang [35] :	1-core	Cortex A8	65	720	LDA	yes	4	3	1100.0000	0.27
Gentile [36] :	1-core	Cortex M4	90	168	Threshold	no	1	1	1.0660	11.38
Liu X [37] :	1-core	Cortex M4	90	80	ANN	no	4	10	25.1500	2.94
Rossi [10] :	1-core	Cortex M4	90	168	HMM/SVM	no	4	6	0.0891	9.97
Falih [11] :	4-core	Cortex A53	65	1200	Naive Bayes	no	8	4	-	-
Benatti17 [19] :	4-cores	OR32Xpulp	130	40	SVM	no	3	3	2.8320	10.00
This work :	**8-cores**	**RV32IMFC Xpulp**	**40**	**500**	**HDC**	**Yes**	**8**	**11**	**0.0832**	**28.46**

Other techniques rely on linear and non-linear modeling of the arm movements, such as the work presented in [39] or the study proposed in [40], where EMG signals are decomposed to extract the neural information and classified with an HMM algorithm. In this vein, solutions based on blind separation of the motoneuron activation [12], [41], which aim for detecting which muscular fibers are actuated during the execution of a given movement, are rapidly gaining ground. Such approaches rely on backpropagation techniques for the training and, moreover, they are based on high performance setup (i.e. $>$64 channels with bandwidth of several kHz).

The lesson learned from this overview shows that the implementation of learning algorithms for pattern recognition is still an open challenge for EMG-based controllers. Luckily, in literature there are some attempts to move pattern recognition design gesture recognition systems toward a wearable form factor, complying with strict energy constraints.

Embedded implementation of computationally demanding algorithms requires a careful design of the system architecture and of the digital computational platform. The most widely used computing platform is the ARM Cortex-M4 [42], a single core RISC-based processor designed for embedded applications. Thanks to the DSP extension of the ISA and Floating Point (FP) support, it represents a good trade off between computational capabilities and energy efficiency. It is widely used in EMG processing, for instance in [33] where a complete gesture recognition system on a ARM cortex-M4 microcontroller is implemented with a SVM used to recognize up to 7 hand gestures with a 8 channel EMG setup. Other works based on this architecture are presented in [36] and in [37] where an ANN with one hidden layer is implemented on a 4 EMG channel setup and it is capable to recognize up to 10 gestures with accuracy higher than 80%. Other works rely on more powerful platforms, to implement more complex models, like the one presented in [34], which leverages an ARM Cortex-A8 processor to execute the EMG gesture recognition using non linear modeling approach. However such high-end systems based on OMAP [43] feature a power consumption significantly higher than ones traditionally exploited for embedded platforms, which is not suitable for a wearable solution with reasonable battery lifetime. Finally, the work presented in [11] leverages a power-hungry Raspberry Pi board (up to 2.3 W of power consumption) with a 4-cores Cortex A53 where a Naive Bayes classifier is implemented to recognize up to 5 EMG gestures for the control of a wheelchair.

All the aforesaid works present the implementation of the classification stage of the algorithm, while the learning is done off-line on a bench-top PC. A first attempt to implement the learning on the embedded platform was tested in [35] where authors implemented the LDA learning on a ARM Cortex-A8 processor. Such a solution allows minimal learning capability but the setup is based on 4 sensors and can recognize only 3 gestures. Our approaches for learning and classification combine the low-power and high-performance capabilities of a PULP programmable platform with a novel brain-inspired algorithm [44] that is extremely robust against low signal-to-noise ratio (SNR) and large variability in both data and computing platform. Computational complexity of the proposed HD computing approach scales linearly with the number of electrodes [45] and maintains its accuracy with various types of biosignal acquisitions, while the PULP platform is highly optimized for parallel applications that require extreme energy efficiency.

Table I provides a quantitative summary of the state of the art of EMG-based pattern recognition embedded systems, including a comparison of energy per classification and battery duration (assuming a 100mAh battery). Systems in [34] and [35] are based on a high-performance ARM Cortex-A8 processor, and although they provide high computational power, they are not suitable for long-term operation due to their high power consumption. The systems presented in [33] and [10] have the lowest energy/classification ratio, which is comparable to our work, but they lack battery life due to the high duty cycle needed. Our system leverages an output classification latency of 8ms, largely compliant with real-time requirements, and as a consequence, it provides more than 2.7x improvement in battery life compared to the SoA systems.

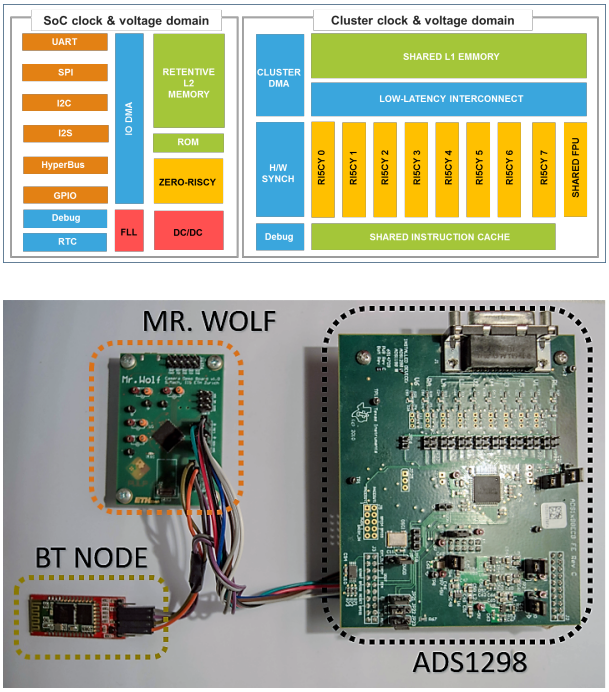


Fig. 1: TOP : Architecture of the Mr Wolf chip Test setup of the systems compared in this work. (BOTTOM) Picture of the tested system. EMG data are acquired with an external ADC (ADS1298), connected to the wolf chip. Output and communication are managed via a BT node.

III. System Architecture

This section describes the hardware and software architecture of the proposed EMG-based gesture recognition system. An image of the proposed system and the diagram of the chip architecture are shown in Fig. 1.

A. Hardware Platform

The proposed system is based with an 8 channel commercial Analog Front End (AFE) connected to the Mr.Wolf SoC [21] breakout board. The AFE, widely used in bio-potential acquisition platforms, is connected in fully differential configuration to an array of passive EMG surface gel-based electrodes, placed in a ring configuration around the upper forearm of the subject. The IC back-end streams the data via SPI to the digital platform. The setup is developed for measurement and characterization purpose, using development boards, but the whole system can be integrated in a single PCB with 40x20mm form factor, suitable for wearable applications.

In this work we developed a multicore low power platform, based on the Mr.Wolf SoC [46] implemented in CMOS 40nm technology. The SoC is a multi-core programmable processor that includes a tiny RISC-V processor (zero-risky) [47], for control functions, and a powerful cluster of 8 RISC-V (ri5cy) processors with extensions for energy efficient digital signal processing [47]. Two memory levels (L1 and L2) are available on the SoC: a 64kB of L1 memory - single cycle latency, multi-bank, which enables shared-memory parallel programming models (e.g. OpenMP), and a 512kB L2 with 15 cycles latency. The parallel cluster accesses this memory with a dedicated direct memory access (DMA) controller. Moreover, the SoC features a full set of peripherals and, to minimize the quantity of interactions with the controlling core, the data transfers are managed by a multi-channel I/O DMA. Since many applications (like in EEG or EMG processing algorithms) require highly dynamic data and need fast computation, the cores in the cluster share two floating-point units (FPU). A dedicated hardware block (HW Sync) supports fast event management, parallel thread dispatching, and synchronization, enabling a fine-grained parallelism, which leads to a high energy efficiency in parallel workloads. An internal DC/DC converter connected to an external battery delivers voltages between 0.8V and 1.1V, maximizing power efficiency. In sleep mode, the voltage regulator is turned off and the real-time clock is powered by a low dropout regulator (LDO) and is fed by a 32kHz crystal oscillator. The LDO controls programmed wake-up and part of the L2 memory for application state retentions for fast wake-up. In deep sleep, the power consumption is reduced to 72 μW (from VBAT) assuming the RTC is active and no data retention, and 108 μW assuming full L2 retention. Hence, duty cycling is extremely effective on Mr. Wolf to decrease average power at low workloads.

B. Hyperdimensional Computing

Brain-inspired HD computing explores the emulation of cognition by computing with points in a HD space, that is, with hypervectors, as an alternative to computing with numbers [44]. Hypervectors are d-dimensional holographic (pseudo)random vectors with independent and identically distributed (i.i.d.) components. When the dimensionality is in the thousands, e.g. $d \geq$1000, there exist a huge number of nearly orthogonal hypervectors [48]. In HD computing we combine such hypervectors into new hypervectors using well-defined vector space operations, defined such that the resulting hypervector is unique with fixed-width.

To ease hardware implementation, we consider dense binary hypervectors that are initially taken from $\{0,1\}^d$ resulting in equal number of randomly placed 0s and 1s. HD Computing supports a full algebra to manipulate these seed hypervectors by using multiplication, addition, and permutation (MAP) operations. When using dense binary codes, MAP operations can be simply implemented by the componentwise XOR ($\oplus$) as multiplication, the componentwise majority function ($[+]$) as addition, and one-bit circular rotation (ρ) as permutation. The addition produces a hypervector *similar* to the input hypervectors and it can represent sets, while multiplication generates a *dissimilar* hypervector and is used to bind hypervectors. The permutation also produces a dissimilar pseudo-orthogonal hypervector, which is used to create hypervectors representing sequences of hypervectors.

Based on the use of these MAP operations, an encoder can be designed for various tasks, e.g., EMG [20], [22], [49], EEG [23], [50], ECoG [51], ExG [45], or in general pattern processing [52]. The encoder emits a hypervector representing the event of interest that is then fed into an associative memory (AM) for training and inference. During

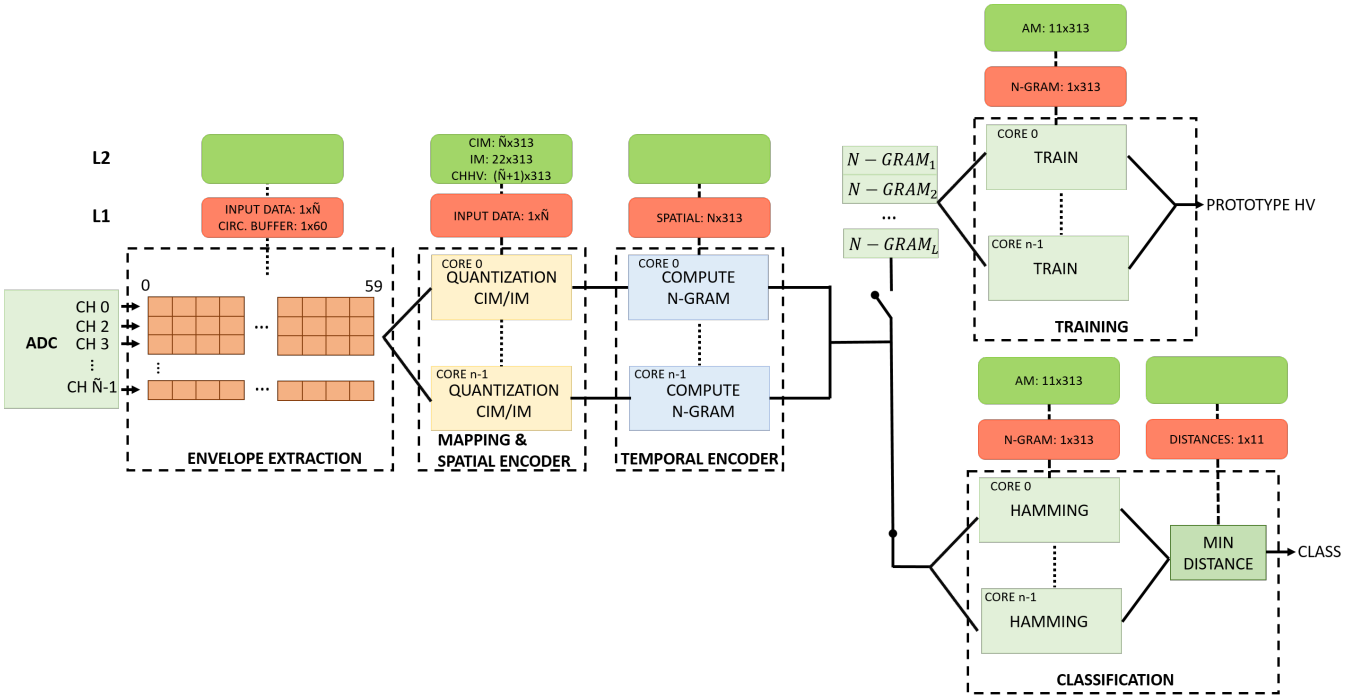


Fig. 2: Implementation on PULP platform of the processing chain. The first three kernels are in common for both training and classification phase. The open/closed switches are there just for a visual idea and the meaning is that once the model is created during the training phase, we will not use that kernel anymore and we pass to the classification kernel.

training, the output hypervector of the encoder is added in the AM as a *learned* pattern. This allows continue updates into AM without requiring to store the raw data or features from the previous examples. During inference, the output of the encoder is compared with the learned patterns. Comparison is based on a distance metric over the vector space. The AM uses Hamming distance, defined as the number of different components of two binary hypervectors. In the following, we describe the encoding approach we developed for EMG signals.

First, to map the features into the HD space we utilize item memory (IM) and continuous item memory (CIM) [49] matrices. The IM is composed of random orthogonal ($\perp$) hypervectors (i.e., $E_1 \perp E_2 ... \perp E_i$), each one associated to the i-th input channel. The CIM contains a set of orthogonal endpoint hypervectors, mapped on the discretized values of the input channels. For instance, if input values are discretized in K levels, we will have K hypervectors ($V_1..V_K$) where V_1 and V_K are associated respectively to the minimum and maximum input values. The hypervectors in CIM are generated by a linear interpolation between the two orthogonal endpoints [49].

The IM and CIM are initialized one time and kept constant during the training and the classification. Once the seed hypervectors are generated from IM and CIM, they are combined by the MAP encoders to represent the event of interest, e.g., a gesture.

The first encoding is obtained with a componentwise XOR between E and V resulting, at instant t:

$$S^t = [(E_1 \oplus V^t_{l(1)}) + ... + (E_i \oplus V^t_{l(i)})]. \quad (1)$$

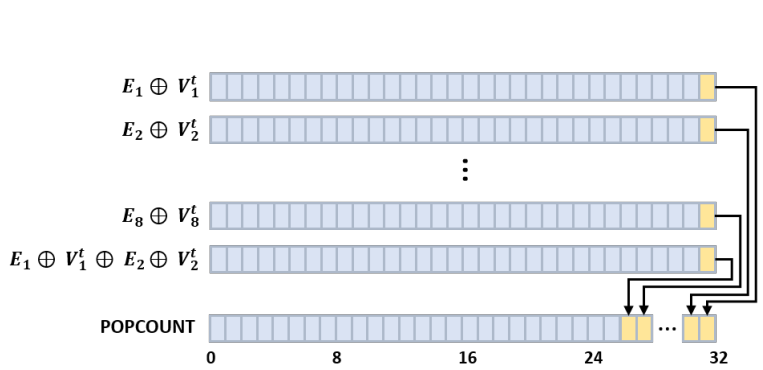
where E_i is the IM vector corresponding to the i-th input channel and $V_{l(i)}$ is the CIM vector corresponding to the $l(i)$-th discretized level of the input sample of channel i. Assuming that, at time 0, channel 1 acquires a new sample and, after the envelope extraction, a signal of amplitude equal to 21 is produced, we can bind this information by $E_1 \oplus V^0_{21}$. Then, the envelope extraction of channel 2 produces a signal with a smaller amplitude, for instance around 7, thus $E_2 \oplus V^0_7$. This is done for all the channels and the related amplitude values, resulting in $S^0 = [(E_1 \oplus V^0_{21}) + (E_2 \oplus V^0_7) + ... + (E_8 \oplus V^0_{15})]$ at instant 0. In this way, the encoder captures the spatial information of a gesture, and can achieve slightly higher accuracy. However, if input data need a temporal component, the temporal encoder extracts information by considering N consecutive hypervectors, from instant t to $t+N-1$. The temporal encoder employs permutation and multiplication to capture order of hypervectors generated by the spatial encoder.

Thus, N spatial hypervectors form an N-gram hypervector (T), defined as:

$$T = S^t \oplus \rho S^{t+1} \oplus \rho^2 S^{t+2} \oplus ... \oplus \rho^{N-1} S^{t+N-1} \quad (2)$$

where ρ^k means applying permutation k times (i.e., k-bit rotation of hypervector). The spatial and temporal encoders are common to both training and classification. The N parameter used in the algorithm is kept constant (i.e. equal to 1) during both training and testing.

During the training, an N-gram hypervector is generated for each trial and added as a *prototype* hypervector related to the class of the trial. The associative memory (AM) contains the same number of prototypes as the number of classes. During




```
#pragma omp parallel num_threads(CORE)
{
#pragma omp master
{
        QUANTIZATION;
}
#pragma omp barrier
#pragma omp for
        for(i = 0; i < 313; i++){

        for(j = 0; j < CHANNELS + 1; j++){

            p.extructu      //bit 31
            p.insert        //bit 31
            ...
            p.extructu      //bit 0
            p.insert        //bit 0

        }
        p.cnt
    }
}
```

Fig. 3: Illustration of how built-ins (p.extract, p.insert, p.cnt) used in the spatial encoder (left). A code snippet to show the usage of builtins and parallelism (OpenMP directives) in the processing chain (right).

the classification, an N-gram is produced from unseen gesture that we call it *query* hypervector. In the AM, the Hamming distances are computed between the query hypervectors and prototypes hypervectors (as learned patterns) to find the label associated to the minimum distance.

IV. Implementation on the PULP platform

HD Computing benefits of a computationally efficient learning algorithm, suitable for resource-constrained platforms. In this paragraph, we describe our implementation of the online and on-chip training giving details on the optimization and demonstrating the feasibility of this approach in fully-embedded energy efficient devices.

Before starting the online on-chip training, the algorithm creates the matrices (*IM/CIM*) required for mapping the samples in the high dimensional space. They should be calculated once with parameters chosen for set-up (number of electrodes and quantization levels), at the beginning of the training phase, and they remain constant for the entire duration of the session. The random orthogonal binary hypervectors are created through a random generation of 32-bit unsigned integer values. For instance, in an eight channel setup IM requires eight orthogonal hypervectors, which are generated through the *rand()* function (included in the GCC library) as 313 random 32-bit unsigned integer values, corresponding to a 10'000-D binary hypervectors ($313 \times 32 \approx 10'000$). The 10'000 dimension has been chosen to define a very high dimensional space able to demonstrate important properties of hyperdimensional representation [53]. Thus, the binary hypervectors are compacted into 313 32-bit unsigned integer variables (i.e. each bit represent one element of the hypervector) to drastically reduce the memory requirements of the application and, hence, the memory accesses.

Similarly, the CIM is computed considering the quantization levels required to represent the input data (i.e. 22 levels), and mapping these levels in an orthogonal hyperspace. The number of discrete levels for quantization has to be carefully selected, depending on the signal we are working with. After an exploration of the data acquired from the 10 subjects involved in the experiment and referring to [20], [49], we set a linear quantization on 22 levels, enough to approximate appropriately the input signals. A finer grain (i.e. more discrete levels) requires a higher number of hypervectors stored in memory (CIM matrix) with no relevant gain in performance, while a lower number of quantization levels affects the classification accuracy.

A. Implementation and Optimization on Mr.Wolf

Aggressive code optimizations can be achieved by compacting the hypervectors in 32-bit integers, leveraging simple operations such as componentwise majority, XOR and shift. The representation of the elements of the hypervectors using unsigned integer variables requires several bitwise operations (i.e. read/insert from/into a 32-bit word) and an operation to count the number of bits set to 1 in a 32-bit word (the so-called popcount). The Ri5cy augmented RISC-V ISA [20] includes several bit manipulation instructions (builtins) to perform this operation in 1 clock cycle, leading to an aggressive performance optimization. The builtins used in this application are *p.extractu, p.insert and p.cnt*. The formers two, *p.extractu and p.insert*, are used respectively to read and set the value assumed by a given bit in an unsigned 32-bit register, while the latter *p.cnt* returns the number of 1s set in a word.

These builtins are used in the spatial encoder to calculate the componentwise majority after binding each channel to its signal level in the HD space and in the learning part to bound all the hypervectors associated to the training samples to create the *prototype* hypervector. Finally, the popcount is used to compute the Hamming distances. In Fig. 3 an example of the use of the builtins is shown. The i components of the hypervectors need to be extracted (i.e., bit-by-bit) to perform the componentwise majority operation and to count the number of bits set to 1 for the majority voting.

To perform the majority voting, the number of channels must be odd. Hence, if the number of channels (i) is even, to avoid randomness in the case of the same number of

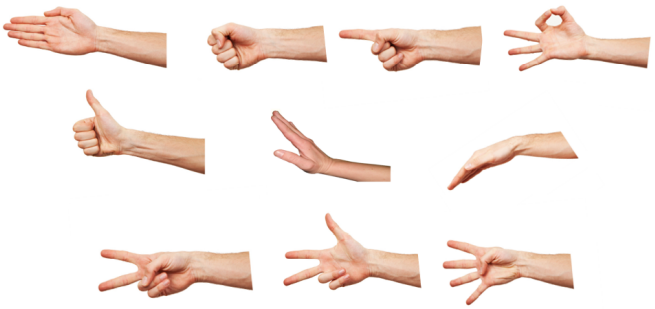


Fig. 4: Gestures used for the testing. Open hand, fist, index, 2-fingers pinch, rest position.

0's and 1's, one reproducible but random hypervector is generated through a componentwise XOR between two bound hypervectors (i.e. the first and the second). For instance, in our case study, with 8 input channels, we use nine bound hypervectors to perform the majority voting. After that, the popcount (p.cnt) is used to evaluate if the number of 0's is higher with respect to the number of bits set to 1. If there are more 1's than 0's, the related bit is set to 1 in the spatial hypervector.

B. Parallelization and Memory Requirements

As shown in Fig. 2, the training and the testing phases have several functions in common. In particular, the feature extraction and the kernels used to map the features into the HD space are the same for both phases. When a new sample acquired by the ADC is available, we extract the envelope of the signals using the RMS upon a circular buffer of dimension N equal to 60. After several simulations, changing the dimension of the buffer (from 30 to 150), we evaluated the differences in classification accuracy. Between 60 and 150 the loss in accuracy is negligible ($\sim$1%), while using a smaller length leads to a more significant loss ($\sim$4% with a buffer size of 30 samples). The feature vector is composed of the envelopes extracted from the signals of each acquisition channel. After the feature extraction, the HD computing algorithm begins. The first kernel maps the features to the HD space, and performs the spatial encoding among all the acquisition channels. In this work, we are considering eight acquisition channels. To map the acquired samples in the HD space, we use the CIM matrix, discretizing the samples in 22 linearly distributed levels. Then, we combine the resulting hypervectors with the one contained in IM related to each channel using the MAP operations.

In the MAP and SPATIAL ENCODER kernel, the parallelization is performed at data level, where the workload is equally distributed among the cores of the cluster. Hence, each core performs the encoding operations on a portion of the hypervector. The cores execute the componentwise XOR operation between CIM and IM and the componentwise majority to create the spatial hypervector. Fig. 3 (right) shows a code snippet of this kernel with the usage of the OpenMP directives for the parallelization.

The spatial hypervector (1x313 32-bit integer array) is stored in the low latency access L1 memory and requires 2kB of memory. Due to the size of the CIM and IM matrices, respectively 22x313 (27 kB) and 8x313 (5 kB), they are stored in the L2 memory. The latency of the accesses to this memory is masked implementing a double buffering policy in which data are transferred from L2 memory to L1 memory through the DMA, while the cores are processing data already available.

If the temporal information contained in the signals is required (N-grams greater than one), the hypervectors in output to the SPATIAL ENCODER goes in the TEMPORAL ENCODER. In this kernel, a sequence of N spatial hypervectors are combined through a componentwise XOR operation after shifting them by one position as permutation. The vector in output to this kernel is the N-gram hypervector (it requires 2 kB, stored the in L1 memory), and serves as input for the AM kernel.

During the training phase, we compute the prototype hypervector and we store it in the L2 memory. The amount of memory required to store the AM matrix depends on the number of gestures we are considering. Storing a single prototype hypervector requires around 2kB, scaling linearly with the number of gestures.

The IM, CIM and AM matrices are computed at runtime and stored in L2 memory for Mr. Wolf, while for the STM32F4-Discovery board they are stored in the FLASH memory.

V. Experimental Results

Evaluating the tradeoff between computational complexity and accuracy is one of the key elements in the selection of a ML algorithm for resource-constrained systems, especially when dealing with online learning capabilities. This section compares both training and classification performance of the proposed HD computing algorithm against the most widely used algorithm for gesture recognition (i.e. SVM), and their optimized implementation on Mr. Wolf and on a commercial ultra-low-power MCU (STM32F407), use as a reference for comparison.

Ten able-bodied subjects (aged 26-42) without previous history of neurological or muscular disorders were involved in the experiment. All participants provided written consent to participate in the experiments. Initially, the system requires the subject to perform a given gesture and collect the data needed for the HD training. The data were acquired using the ADS1298 in the 8 channels fully-differential configuration. EMG passive wet Ag-AgCl electrodes (25mm diameter) were placed around the forearm at the same distance from each other with the arm facing downwards. Inter-electrode distance (i.e. distance between positive and negative channels) was set to 10mm. The gesture set was: *open hand, power grip, index pointed, 2-fingers pinch, wrist supination, wrist pronation, number two, number three, number four* and *rest position*, in the order shown in Fig. 4. During the acquisition, the subjects repeated each movement 6 times, holding the position with medium intensity of muscular contraction for 5 seconds and separating gestures with 5 second of rest position to avoid

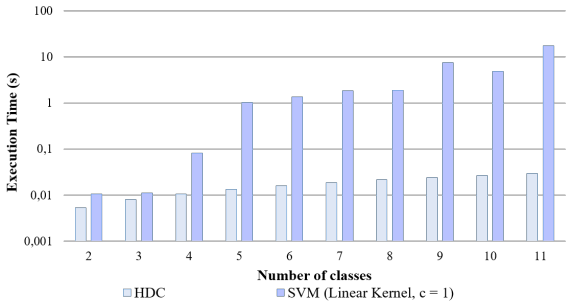


Fig. 5: Estimation of the execution time for the training of SVM and HD Computing increasing the number of gestures (from 1 to 11). Results are presented in logarithmic scale.

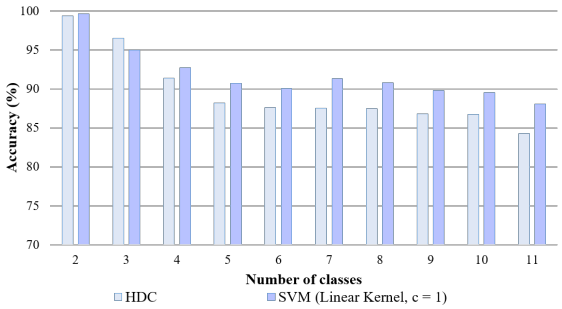


Fig. 6: Average accuracy obtaining by SVM and HD computing, using the same data collected by 10 subjects, increasing the number of gestures (from 1 to 11).

muscular fatigue. 25% of the samples for each acquired gesture were used for the training , while the whole dataset is used for testing. This approach has results very similar to the traditional procedure used in gesture recognition applications, where the data used for training and testing are separated by trials (i.e. some trials used for training and the others for testing). We tested offline the 2 approaches and we obtained accuracy difference below 1%. This is due to the fact that gestures are basically ”static”, i.e. the contraction level is maintained almost constant all over the duration of the trial (excluding the transients), resulting in similar contraction patterns for all the trials of a given gesture.

A. Comparison between SVM and HD computing of Training Times and Classification Accuracy

One of the most important advantages of HD computing is the possibility to perform the training of the algorithm ’one-shot’, paving the way to the real-time implementation and execution of the training phase. This section compares the computational load for the training of the SVM against HD and the accuracy in classification, starting with two gestures and scaling up to determine the computational cost of the two algorithms in different application scenarios (i.e. where a different number of recognized gestures are required). In this experiments, the SVM uses a linear kernel (c=1), with data normalized between 0 and 1. Such configuration has been selected after experimental evaluation, and represents the best trade-off between performance and computational complexity for this setup. While it has been observed that, for some signals such as EEG, the temporal information is extremely important to achieve good classification accuracy, this is not the case for EMG signals [33], [49], whereas experiments are based on ”static” gestures. Thus, the results in this section are presented considering N=1 for all subjects (the same for training and classification).

Results are shown in Fig. 5. The graph is plotted in logarithmic scale for the sake of clarity. It is noteworthy that the training time of HD computing algorithm is deterministic and grows linearly when the number of classes increases. On the contrary, SVM training time is not deterministic (i.e. it depends on the convergence time of the algorithm) and, more importantly, it increases super-linearly with the number of gestures, resulting not affordable for embedded implementation if the number of gestures is higher than 5. Overall, the training time of the SVM is between 2 and 3 orders of magnitude higher than the HD Computing.

After the characterization of the training time of the SVM against the HD computing we have measured the accuracy of the classification obtained with the 2 classifiers. The average accuracy values are shown in Fig. 6. Results are comparable in terms of accuracy, showing that HD computing has a maximun 4% accuracy loss wrt SVM, resulting suitable for a hand gesture controller [54].

B. Evaluation of Execution Performance

In this section, we discuss results of the HD computing implementation (both training and testing) on the proposed embedded system. Table II shows the execution times of the algorithm on Mr. Wolf incrementally exploiting the features of the architecture, namely the DSP extensions available on the core (Mr. Wolf 1-core built-ins), and the parallelism of the 8 cores available on the cluster (Mr. Wolf 8-cores built-ins). The implementation on an STM32F4-DISCOVERY board (ARM Cortex M4) is also shown in the table as reference. Results refer to hypervectors of 10’000-D, N equal to 1 (i.e. no temporal information), 11 classes (10 gestures and the rest position). We set the N parameter to 1 to show the fastest case for training and classification, and also because the MAP+ENCODERS kernel increases linearly, increasing the temporal window length. This results also in a better generalization of the application, since the same level of temporal encoding is used for all the subjects. In more subject-specific applications (beyond the scope of this work), it is possible to adapt the temporal encoders scaling the value of N, adding more temporal information, besides the RMS [49].

Considering the upper part of the table (TRAINING), the first column shows the cycles required for the execution of the RMS and MAP+ENCODERS for one sample (each 100ms). To better understand the part of the processing we are taking into account, we can refer to Fig. 7 that shows where the computational kernels are executed.

Theoretically, to collect the training set, we need to continuously extract the envelope from the signals. As already

TABLE II: HD Computing Execution times on the target architectures, with 10,000-D, N=1. (Cyc, su) stand for (cycles, speed-up). The total energy/class reported, is the result of the addition of the contribution of these functions without considering the energy during idle periods.

		ARM CORTEX M4			Mr. Wolf 1 core				Mr. Wolf 1 core built-ins				Mr. Wolf 8 cores built-ins			
	Kernel	**cyc(k)**[a]	**cyc(k)**[b]	**E(μJ)**[d]	**cyc(k)**[a]	**cyc(k)**[b]	**su**[c]	**E(μJ)**[e]	**cyc(k)**[a]	**cyc(k)**[b]	**su**[c]	**E(μJ)**[e]	**cyc(k)**[a]	**cyc(k)**[b]	**su**[c]	**E(μJ)**[e]
TRAIN	RMS	13.78	399.62	202.42	6.82	197.78	2.02	24.99	6.82	197.78	2.02	24.99	0.89	25.81	7.66	5.13
	MAP+ENCODERS	537.71	15'593.59	7'898.90	569.10	16'503.90	0.94	2'085.49	215.35	6'245.15	2.50	789.16	27.94	810.26	19.25	161.22
	TRAINING		41'305.84	20'923.37		25'620.47	1.61	3'237.50		16'696.69	2.47	2'109.85		2'136.18	19.34	425.05
	TOT TRAIN		57'299.05	29'024.70		42'322.15	1.59	5'347.99		23'139.60	2.47	2'924.01		2'972.25	19.33	591.41
TEST	RMS	13.78		6.98	6.82		2.02	0.86	6.82		2.02	0.86	0.89		7.66	0.17
	MAP+ENCODERS	537.71		272.37	569.10		0.95	71.91	215.35		2.50	27.21	27.94		19.24	5.55
	AM	70.65		35.78	68.59		1.03	8.66	24.19		2.92	3.05	7.23		10.06	1.43
	TOT TEST	622.15		315.14	644.48		0.97	81.44	246.37		2.53	31.13	36.06		17.25	7.17

[a] cycles per sample, [b] cycles per class, [c] speed-up wrt ARM Cortex M4, [d] 168MHz@1.85V [e] 100MHz@0.8V

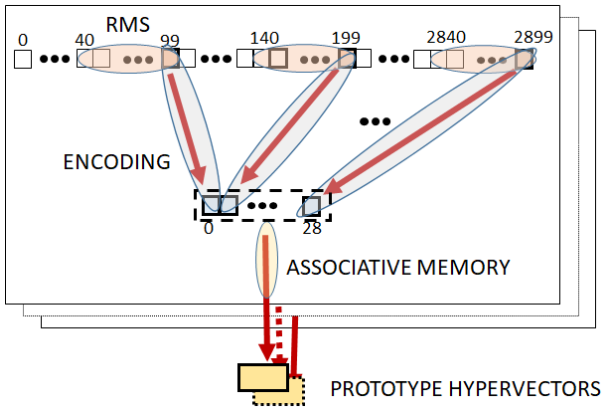


Fig. 7: Sequence of the computational kernels of the HDC training. The first kernel is the RMS, computed on 60-samples windows, every $100ms$. Output of the RMS is used by the encoding kernel to calculate the AM vectors. Once the 29 vectors of the associative memory are calculated, they are used to calculate the prototype hypervectors (1 for each class of the problem)

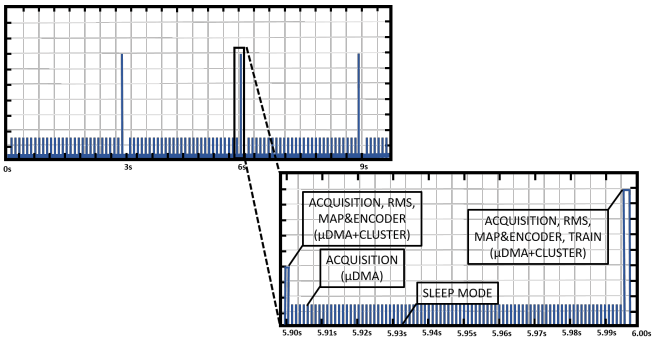


Fig. 8: Power envelope of the training algorithm. EMG samples are acquired during *Acquisition* steps,whereas PULP cluster is turned off and uDMA manages data transfer from SPI to uDMA. Once the buffer for RMS computation is ready, the cluster is switched on and calculates RMS value and hypervectors for the encoder (*RMS MAP and ENCODING*).

discussed, to reduce the number of samples used for the training and to cover the entire shape of the gesture, we use a downsampling factor of 100 (this factor can vary depending on the number of samples used for the training and the duration of the gesture). In this way, it is possible to compute the RMS only 1 time after the acquisition of the 100th sample (Fig. 8).

In the RMS kernel we can see that just changing the architecture (passing from ARM Cortex M4 to 1-core Mr. Wolf) leads to an improvement of 2.0×. This improvement is mainly due to the hardware loops and the floating-point Fused Multiply and Accumulate (FMA) available in Mr. Wolf. In fact, in this part of the processing chain, we are considering floating point variables that are successively binarized into hypervector. This kernel is parallelized splitting the acquisition channels (i.e. 8, 16, 32, etc.) to calculate envelope values among the cores inside the cluster reaching a nearly ideal speed-up.

The MAP+ENCODERS kernel results slightly slower when executing on a single-core of Mr. Wolf, when no bitwise manipulation extensions are being used. This is due to some of the instructions available on the ARM Cortex M4 architecture (i.e. *load and shift* and *load 32-bit immediate*), useful to compute the majority function (based on population count operation), heavily relying on load-and-shift and 32-bit load immediate operations not available in the baseline RISC-V Instruction Set Architecture (ISA). However, this kernel can be highly optimized in Mr. Wolf taking advantage of the DSP extensions. Since the built-ins to extract/insert bits into a 32-bit word require as input a 5-bit immediate to indicate the position of the bit we are considering, we unrolled the inner loop including this function to remove the dependency with the loop index, fully exploiting the capabilities of these extensions, leading to an improvement of 2.5× over Cortex M4 (when considering a single core).

Furthermore, this kernel demonstrates almost ideal parallelism due to the lack of dependences between the different iterations of the inner loops. In fact parallelizing the execution of the code on 8 cores improves this gain up to 19.2× with respect to the ARM Cortex M4 execution time, showing a nearly ideal speed-up when comparing its execution time with respect to the execution on a single core (i.e. 7.7×). The last kernel takes all the N-grams related to the training samples and creates the prototype hypervectors for all classes. Built-ins can be effectively used also in this case to optimize the execution, since the majority operation is required also in this kernels, leading to 2.4× better performance (unrolling the loop as in the previous case) compared to ARM Cortex M4 and 19.3× when exploiting the parallel processing cluster.

In the second part of Table II (TEST), results for the testing of the HD Computing algorithm are shown. The first two kernels (RMS and MAP+ENCODERS) have been already discussed in the description of the training results. The last kernel is the AM, where the *query* is classified in one of the possible classes (11 in this particular case). Exploiting the parallel execution leads to a saturation on the speed-up because of the small quantity of workload to distribute to the cores. In fact, the speed-up obtained using 8-cores Mr. Wolf is equal to 3.3× (10.0× wrt ARM Cortex M4). This small gain does not impact significantly on the overall speed-up (6.8× wrt 1-core Mr. Wolf with builtins and 17.3× wrt ARM Cortex M4) as the dominant part is the MAP+ENCODER kernel.

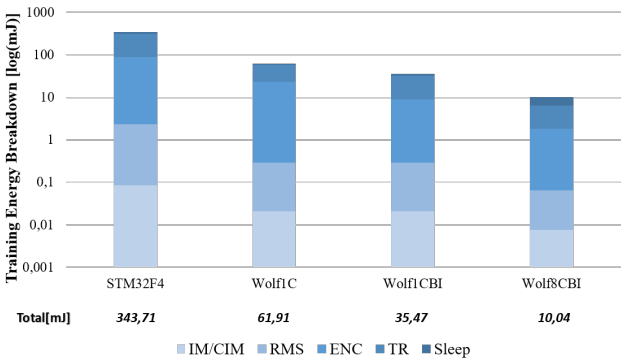
C. Evaluation of Energy Consumption

In this section, we make a comparison in term of energy consumption between the target architectures. The commercial ARM Cortex M4 MCU works at an operating frequency of 168MHz and 1.85V, while we set the operating frequency of Mr. Wolf architecture to 100MHz (the maximum is 450MHz) at 0.8V, the most efficient operating point. Fig. 8 gives insights about the training of the algorithm. Each gesture is held for three seconds.

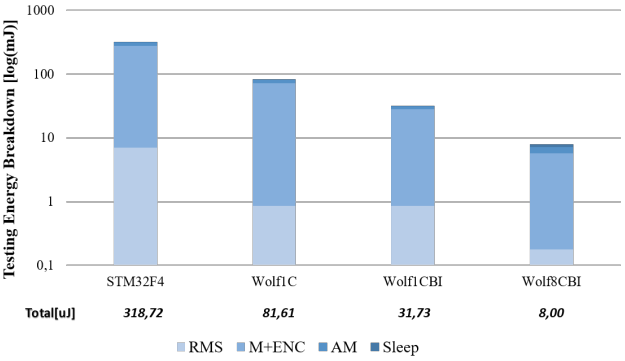
EMG signals are acquired at 1kHz, hence a new sample is available each 1ms. To reduce the number of samples required to train the algorithm, while maintaining a clear idea of the shape of the gesture, we calculate the RMS value on 60 samples each 100ms (see Fig. 7). Hence, we can store in memory the acquired samples using the μDMA keeping the PULP cluster switched off, and then, after the acquisition of the 100th sample, we can wake up the cluster and perform the RMS and the MAP+ENCODERS kernel creating a new hypervector.

As soon as all the hypervectors required by the training are generated, we can generate the prototype hypervector that will be used for the classification. When no elaboration and no acquisition are required, we can put the cluster and part of the SoC in deep sleep to further save energy. Table II also shows the energy consumption for each kernel of the training and the testing of the algorithm. The total energy consumption for the training of the algorithm on the ARM Cortex M4, including all eleven gestures, is equal to 29.02mJ, while the energy consumption on the single core Mr. Wolf is 5.35mJ (5.4 energy boost). This improvement mainly derives from the difference in technology (40nm for Mr. Wolf and 90nm for ARM Cortex M4) and the differences in the operating state. The ARM Cortex M4 operates at its maximum operating frequency (168MHz) at 1.85V, while for Mr. Wolf, which can work at a maximum frequency of 500MHz, finds its most efficient operating point at 100MHz. We can also lower the voltage operating in near-threshold (0.8V). The use of built-ins can improve this gap in energy and exploiting the parallel computing on eight cores, reaching an energy boost of 49.1×.

During the classification, the energy boosts are more significant because they represent the energy consumed by the MCU for most of the lifetime of the application. The boost achieved by changing the architecture (from the commercial MCU to the single core Mr. Wolf) leads to a 3.9× improvement in energy. Optimizing through the built-ins gives a further 10.1× improvement while splitting the workload among eight cores allows us to obtain up to 44.0× gain, with an energy consumption of 7.2μJ. Considering also the energy required by Mr. Wolf (8-cores) at deep sleep, the system can provide nearly 300h of autonomy. In this evaluation, we do not include the ADC power to better showcase the benefits of the architecture.



(a)



(b)

Fig. 9: Energy breakdown during training (a) and testing (b) using the HDC on the target platforms. The bars are plotted in log scale to better show the energy contribution of each function. The total energy consumption at the bottom of the figure is reported in linear scale. Working with 8-cores, Mr. Wolf achieves the lowest energy consumption, saving up to 34x energy with respect to the commercial STM32F4 MCU.

Fig. 9(a) and 9(b) show the energy breakdown during the training an testing. Both plots are represented in a logarithmic scale to better exhibit the energy contribution of each function at different orders of magnitude. In fact, this is already denoting the benefits of using Mr. Wolf as a processing architecture. During both phases, the energy consumption is the result of the contribution of the power modes. When processing is required, the MCUs will run at the maximum allowed core frequencies (Run mode), and when in idle, they will be put in deep-sleep.

During training, the contributions in energy for the execution of the kernel functions in nearly all architectures are similar, i.e., dominated by the ENC and TRAINING kernels. The case is different for Mr. Wolf (8-cores), where the energy

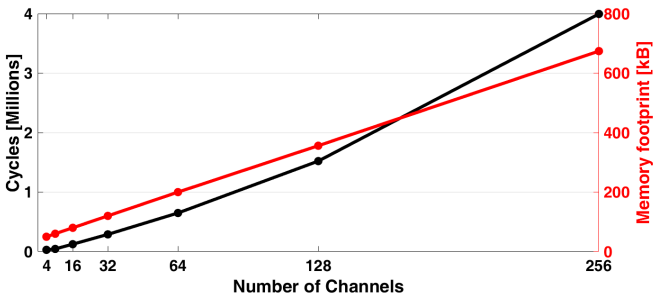


Fig. 10: Performance and memory footprint of HD computing, increasing the number of channels. The results refer to 8 cores Mr. Wolf execution with built-ins.

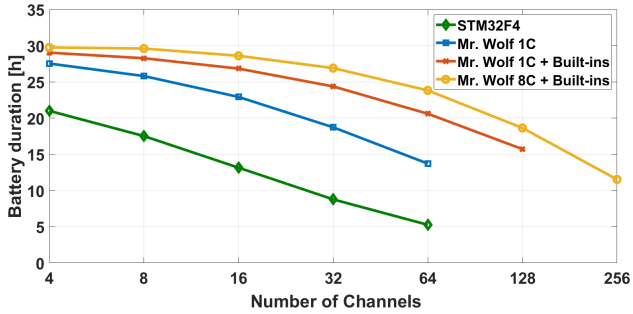


Fig. 11: Battery duration of the target applications including the static power of the ADC with an increasing number of channels. With a fixed latency of 40ms per classification, Mr. Wolf 8-cores, even with a single core, can provide the same performance than the commercial STM32 (up to 64 channels).

required during the active time becomes comparable with the energy employed in deep sleep mode, thus, approaching the lower bounds of the energy consumption. This is possible by taking advantage of the parallelization of the code and the technology improvements previously presented. The same trend is noticeable during the testing (i.e., real-time classification), where the energy of most computationally expensive function, MAP+ENCODERS (M+ENC) kernel, becomes more comparable with the deep-sleep energy when moving towards the architectures with higher core count. As a result, Mr. Wolf (8-cores) works with an average power consumption of 0.82mW, demonstrating that complex signal processing and ML can be executed at an extremely low power budget.

VI. Scalability

The results presented in Section V focus on the use case addressed in this work for EMG hand gesture recognition with 8 channels and an N-gram size of one. This HD computing framework can handle more complex tasks (i.e. EEG processing) where a higher number of channels for a larger coverage and larger N-gram size for a wider temporal window are required [20]. To demonstrate the capabilities of this framework to handle other type of applications, we assess the scalability by increasing the number of channels from 4 to 256 channels. Increasing the number of channels affects both execution time and memory requirements. As shown in Fig. 2, the processing

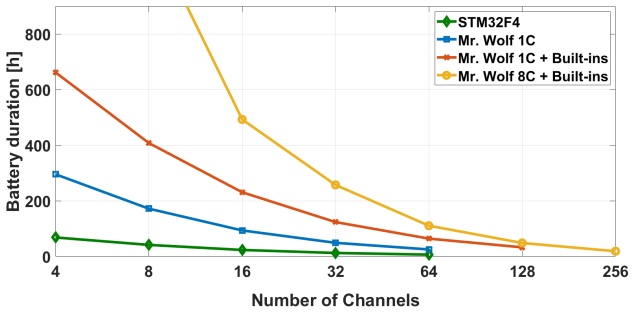


Fig. 12: Battery duration of the target applications with an increasing number of channels. This figure excludes the static power of the ADC to better showcase the power characteristics of the target architectures. The single core version of Mr. Wolf (without built-ins) can provide the same autonomy than the commercial ARM Cortex-M4 MCU (up to 64 channels). With built-ins, the same core offers more than one day of battery life (up to 128 channels), and the 8-core version, up to 18h (256 channels).

chain can be divided in two part. The first part includes the kernels "channel dependent" (RMS and MAP+ENCODERS), for which the performance changes increasing the number of channels, while the second part includes the kernels "channel independent" that have no impact on the performance scaling up the number of channels. Fig. 10 shows that the RMS and the MAP+ENCODERS kernels linearly increase the execution time scaling up the number of channels (from 4 to 256), so as the memory requirements for the correct execution of the processing chain. This can be extended for the testing part, as the number of cycles for the AM kernel (channel independent) decreases the impact on the overall performance as the number of channels increases. Similarly, we can extend this discussion for the training part taking into account that the number of cycles reported on the graph has to be multiplied by the number of samples required for the training of each class and the cycles required to create the prototype hypervector has to be added.

Furthermore, we have also explored the impact on the battery life when scaling the number of channels for the training part. Fig. 11 shows the autonomy of the target architectures including the static power of the ADC and considering a battery of 100mAh. With a classification latency of 50ms, the STM32 and Mr. Wolf (1-core without built-ins) can only be scaled up to 64 channels because they are not able to meet this latency constraint. Nevertheless, Mr. Wolf denotes a more efficient operation by providing up to 3.9x more autonomy than the STM32. If we include the built-ins, the same core can now be scaled up 128 channels, providing more than 48h of continuous operation and demonstrating the benefits of the built-in instructions. Nonetheless, for the current application, Mr. Wolf (8-cores with built-ins) exceeds all other architectures (for every number of channels) mainly thanks to the distribution of the task into the different cores.

Another interesting fact comes from the line shapes, where all Mr. Wolf architectures will show a pseudo-constant average

autonomy when increasing the number of channel up to 64 (with respect to four channels), with only 32% of autonomy degradation, while the commercial MCU will show peak degradation of 74%, i.e., a linear and steep decrease. This difference is due to the negligible power contribution of Mr. Wolf with respect to the ADC power. When scaling up the system with more channel, Mr. Wolf will still provide 19h to 12h of operation for 128 and 256 respectively, adequate for daily use.

We also present in Fig. 12 the autonomy of the architectures without including the ADC contributions, which can be subject of further optimizations depending on the application, and it is currently out of the scope of this work. This representation also highlights the benefits of using the PULP architecture, where again, Mr. Wolf exceeds the commercial MCU, being able to provide from 1'400h to 19h of continuous operation with an increasing number of channels.

VII. Conclusion

In this paper we presented a complete framework for EMG gesture recognition. Our solution includes a novel algorithm for fast online training, implemented on a programmable multicore platform. Leveraging the HW-SW co-design which ranges from algorithm to embedded optimization, we aim to enable a simple and scalable solution for supervised pattern recognition. In fact, the supervised learning approach is widely used in pattern recognition applications, but it lacks versatility, due to heavy computational requirements of the training stage.

Our solution has been tested on 10 subjects in a typical gesture recognition scenario. HD computing with online learning reaches 85% accuracy on 11 gestures recognition, which is aligned with the SoA. Furthermore, by virtue of the efficient Mr. Wolf multicore processor, the energy budget required to run the learning part with 11 gestures is 10.04mJ, and 83.2μJ for one classification. The system works to a average power of 10.4mW in classification, ensuring around 29h of autonomy with a battery of 100mAh.

As future work, we plan to integrate the system in a miniaturized form factor and to optimize the power consumption. Furthermore, we plan to integrate and test this solution for a higher number of channels and to extend the HD computing based solutions for other biopotentials, such as EEG signals.

Acknowledgment

This work has been partially supported by the European H2020 FET project OPRECOMP under Grant 732631, and the EU's Horizon 2020 Research and Innovation Program through the project MNEMOSENE under Grant 780215.

References

[1] T. Starner, J. Weaver, and A. Pentland, "Real-time american sign language recognition using desk and wearable computer based video," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 20, no. 12, pp. 1371–1375, 1998.

[2] T. S. Saponas, D. S. Tan, D. Morris, R. Balakrishnan, J. Turner, and J. A. Landay, "Enabling always-available input with muscle-computer interfaces," in *Proceedings of the 22nd annual ACM symposium on User interface software and technology*. ACM, 2009, pp. 167–176.

[3] touch bionics, http://www.touchbionics.com/products, 2018.

[4] Ottobock, https://www.ottobockus.com/prosthetics/upper-limb-prosthetics/solution-overview/myoelectric-prosthetics/, 2018.

[5] R. Meattini, S. Benatti, U. Scarcia, D. D. Gregorio, L. Benini, and C. Melchiorri, "An semg-based humanrobot interface for robotic hands using machine learning and synergies," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 8, no. 7, pp. 1149–1158, July 2018.

[6] H. Zhang, Y. Zhao, F. Yao, L. Xu, P. Shang, and G. Li, "An adaptation strategy of using lda classifier for emg pattern recognition," in *Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE*. IEEE, 2013, pp. 4267–4270.

[7] M. R. Ahsan, M. I. Ibrahimy, and O. O. Khalifa, "Electromygraphy (emg) signal based hand gesture recognition using artificial neural network (ann)," in *2011 4th International Conference on Mechatronics (ICOM)*, May 2011, pp. 1–6.

[8] M. A. Oskoei, H. Hu *et al.*, "Support vector machine-based classification scheme for myoelectric control applied to upper limb." *IEEE Trans. Biomed. Engineering*, vol. 55, no. 8, pp. 1956–1965, 2008.

[9] P. Zhang and J. Peng, "Svm vs regularized least squares classification," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, vol. 1, Aug 2004, pp. 176–179 Vol.1.

[10] M. Rossi, S. Benatti, E. Farella, and L. Benini, "Hybrid emg classifier based on hmm and svm for hand gesture recognition in prosthetics," in *Industrial Technology (ICIT), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1700–1705.

[11] A. D. I. Falih, W. A. Dharma, and S. Sumpeno, "Classification of emg signals from forearm muscles as automatic control using naive bayes," in *2017 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, Aug 2017, pp. 346–351.

[12] D. Farina and A. Holobar, "Characterization of human motor units from surface emg decomposition," *Proceedings of the IEEE*, vol. 104, no. 2, pp. 353–373, 2016.

[13] M. Atzori, M. Cognolato, and H. Müller, "Deep learning with convolutional neural networks applied to electromyography data: A resource for the classification of movements for prosthetic hands," *Frontiers in neurorobotics*, vol. 10, p. 9, 2016.

[14] A. Waris, I. Mendez, K. Englehart, W. Jensen, and E. N. Kamavuako, "On the robustness of real-time myoelectric control investigations: A multiday fitts law approach," *Journal of Neural Engineering*, 2018.

[15] A. J. Ishak, S. A. Ahmad, A. C. Soh, N. A. Naraina, R. M. R. Jusoh, and W. Chikamune, "Design of a wireless surface emg acquisition system," in *2017 24th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, Nov 2017, pp. 1–6.

[16] B. Milosevic, E. Farella, and S. Benatti, "Exploring arm posture and temporal variability in myoelectric hand gesture recognition," in *2018 7th IEEE International Conference on Biomedical Robotics and Biomechatronics (Biorob)*. IEEE, 2018, pp. 1032–1037.

[17] A. Waris, I. K. Niazi, M. Jamil, K. Englehart, W. Jensen, and E. N. Kamavuako, "Multiday evaluation of techniques for emg based classification of hand motions," *IEEE journal of biomedical and health informatics*, 2018.

[18] Z. Zainuddin, N. Mahat, and Y. A. Hassan, "Improving the convergence of the backpropagation algorithm using local adaptive techniques." in *International Conference on Computational Intelligence*. Citeseer, 2004, pp. 173–176.

[19] S. Benatti, G. Rovere, J. Bsser, F. Montagna, E. Farella, H. Glaser, P. Schnle, T. Burger, S. Fateh, Q. Huang, and L. Benini, "A sub-10mw real-time implementation for emg hand gesture recognition based on a multi-core biomedical soc," in *2017 7th IEEE International Workshop on Advances in Sensors and Interfaces (IWASI)*, June 2017, pp. 139–144.

[20] 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 *Proceedings of the 55th Annual Design Automation Conference*, ser. DAC '18. New York, NY, USA: ACM, 2018, pp. 111:1–111:6. [Online]. Available: http://doi.acm.org/10.1145/3195970.3196096

[21] A. Pullini, D. Rossi, I. Loi, A. D. Mauro, and L. Benini, "Mr. wolf: A 1 gflop/s energy-proportional parallel ultra low power soc for iot edge processing," in *ESSCIRC 2018 - IEEE 44th European Solid State Circuits Conference (ESSCIRC)*, Sept 2018, pp. 274–277.

[22] A. Moin, A. Zhou, A. Rahimi, S. Benatti, A. Menon, S. Tamakloe, J. Ting, N. Yamamoto, Y. Khan, F. Burghardt, L. Benini, A. C. Arias, and J. M. Rabaey, "An emg gesture recognition system with flexible high-density sensors and brain-inspired high-dimensional classifier," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2018, pp. 1–5.

[23] 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 Networks and Applications*, pp. 1–12, Oct 2017.

[24] S. Pancholi and A. M. Joshi, "Portable emg data acquisition module for upper limb prosthesis application," *IEEE Sensors Journal*, vol. 18, no. 8, pp. 3436–3443, 2018.

[25] M.-F. Lucas, A. Gaufriau, S. Pascual, C. Doncarli, and D. Farina, "Multi-channel surface emg classification using support vector machines and signal-based wavelet optimization," *Biomedical Signal Processing and Control*, vol. 3, no. 2, pp. 169–174, 2008.

[26] S. Bitzer and P. Van Der Smagt, "Learning emg control of a robotic hand: towards active prostheses," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. IEEE, 2006, pp. 2819–2823.

[27] A. Alkan and M. Günay, "Identification of emg signals using discriminant analysis and svm classifier," *Expert Systems with Applications*, vol. 39, no. 1, pp. 44–47, 2012.

[28] C. Castellini, E. Gruppioni, A. Davalli, and G. Sandini, "Fine detection of grasp force and posture by amputees via surface electromyography," *Journal of Physiology-Paris*, vol. 103, no. 3-5, pp. 255–262, 2009.

[29] C. Castellini and P. van der Smagt, "Surface emg in advanced hand prosthetics," *Biological cybernetics*, vol. 100, no. 1, pp. 35–47, 2009.

[30] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, ser. COLT '92. New York, NY, USA: ACM, 1992, pp. 144–152. [Online]. Available: http://doi.acm.org/10.1145/130385.130401

[31] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[32] H. Bensmail and G. Celeux, "Regularized gaussian discriminant analysis through eigenvalue decomposition," *Journal of the American statistical Association*, vol. 91, no. 436, pp. 1743–1748, 1996.

[33] S. Benatti, F. Casamassima, B. Milosevic, E. Farella, P. Schönle, S. Fateh, T. Burger, Q. Huang, and L. Benini, "A versatile embedded platform for emg acquisition and gesture recognition," *IEEE transactions on biomedical circuits and systems*, vol. 9, no. 5, pp. 620–630, 2015.

[34] J. Liu, F. Zhang, and H. H. Huang, "An open and configurable embedded system for emg pattern recognition implementation for artificial arms," in *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE*. IEEE, 2014, pp. 4095–4098.

[35] X. Zhang, H. Huang, and Q. Yang, "Real-time implementation of a self-recovery emg pattern recognition interface for artificial arms," in *Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE*. IEEE, 2013, pp. 5926–5929.

[36] P. Gentile, M. Pessione, A. Suppa, A. Zampogna, and F. Irrera, "Embedded wearable integrating real-time processing of electromyography signals," in *Multidisciplinary Digital Publishing Institute Proceedings*, vol. 1, no. 4, 2017, p. 600.

[37] X. Liu, J. Sacks, M. Zhang, A. G. Richardson, T. H. Lucas, and J. Van der Spiegel, "The virtual trackpad: An electromyography-based, wireless, real-time, low-power, embedded hand-gesture-recognition system using an event-driven artificial neural network," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 64, pp. 1257–1261, 2017.

[38] Y. Hu, Y. Wong, W. Wei, Y. Du, M. Kankanhalli, and W. Geng, "A novel attention-based hybrid cnn-rnn architecture for semg-based gesture recognition," *PloS one*, vol. 13, no. 10, p. e0206049, 2018.

[39] J. M. Hahne, F. Biessmann, N. Jiang, H. Rehbaum, D. Farina, F. Meinecke, K.-R. Müller, and L. Parra, "Linear and nonlinear regression techniques for simultaneous and proportional myoelectric control," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 22, no. 2, pp. 269–279, 2014.

[40] N. Jiang, K. B. Englehart, and P. A. Parker, "Extracting simultaneous and proportional neural control information for multiple-dof prostheses from the surface electromyographic signal," *IEEE transactions on Biomedical Engineering*, vol. 56, no. 4, pp. 1070–1080, 2009.

[41] K. R. Wheeler, M. H. Chang, and K. H. Knuth, "Gesture-based control and emg decomposition," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 36, no. 4, pp. 503–514, 2006.

[42] ARM Cortex M4, https://developer.arm.com/products/processors/cortex-m/cortex-m4, 2013.

[43] OMAP processor, http://www.ti.com, 2013.

[44] P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cognitive Computation*, vol. 1, no. 2, pp. 139–159, 2009. [Online]. Available: http://dx.doi.org/10.1007/s12559-009-9009-8

[45] A. Rahimi, P. Kanerva, L. Benini, and J. M. Rabaey, "Efficient biosignal processing using hyperdimensional computing: Network templates for combined learning and classification of exg signals," *Proceedings of the IEEE*, pp. 1–21, 2018.

[46] D. Rossi *et al.*, "Energy-efficient near-threshold parallel computing: The pulpv2 cluster," *IEEE Micro*, vol. 37, no. 5, pp. 20–31, September 2017.

[47] P. D. Schiavone *et al.*, "Slow and steady wins the race? a comparison of ultra-low-power risc-v cores for internet-of-things applications," in *PATMOS*, Sept 2017, pp. 1–8.

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

[49] 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 *IEEE International Conference on Rebooting Computing*, October 2016.

[50] A. Rahimi, P. Kanerva, J. del R Millán, and J. M. Rabaey, "Hyperdimensional computing for noninvasive brain–computer interfaces: Blind and one-shot classification of EEG error-related potentials," *10th ACM/EAI International Conference on Bio-inspired Information and Communications Technologies (BICT)*, 3 2017.

[51] 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 *Biomedical Circuits and Systems Conference (BioCAS), 2018 IEEE*, 2018, pp. 1–4.

[52] D. Kleyko, E. Osipov, A. Senior, A. I. Khan, and Y. A. ekercioglu, "Holographic graph neuron: A bioinspired architecture for pattern processing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 6, pp. 1250–1262, June 2017.

[53] P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cognitive Computation*, vol. 1, no. 2, pp. 139–159, 2009.

[54] S. Benatti, B. Milosevic, E. Farella, E. Gruppioni, and L. Benini, "A prosthetic hand body area controller based on efficient pattern recognition control strategies," *Sensors*, vol. 17, no. 4, p. 869, 2017.