2017 IEEE International Symposium on High Performance Computer Architecture

Exploring Hyperdimensional Associative Memory

Mohsen Imani[‡], Abbas Rahimi[*], Deqian Kong[‡], Tajana Rosing[‡], and Jan M. Rabaey[*]
[‡]CSE, UC San Diego, La Jolla, CA 92093, USA
[*]EECS, UC Berkeley, Berkeley, CA 94720, USA
{moimani, deqian, tajana}@ucsd.edu; {abbas, jan}@eecs.berkeley.edu

***Abstract*—Brain-inspired hyperdimensional (HD) computing emulates cognition tasks by computing with hypervectors as an alternative to computing with numbers. At its very core, HD computing is about manipulating and comparing large patterns, stored in memory as hypervectors: the input symbols are mapped to a hypervector and an associative search is performed for reasoning and classification. For every classification event, an associative memory is in charge of finding the closest match between a set of *learned* hypervectors and a *query* hypervector by using a distance metric. Hypervectors with the i.i.d. components qualify a memory-centric architecture to tolerate massive number of errors, hence it eases cooperation of various methodological design approaches for boosting energy efficiency and scalability. This paper proposes architectural designs for hyperdimensional associative memory (HAM) to facilitate energy-efficient, fast, and scalable search operation using three widely-used design approaches. These HAM designs search for the nearest Hamming distance, and linearly scale with the number of dimensions in the hypervectors while exploring a large design space with orders of magnitude higher efficiency. First, we propose a digital CMOS-based HAM (D-HAM) that modularly scales to any dimension. Second, we propose a resistive HAM (R-HAM) that exploits timing discharge characteristic of nonvolatile resistive elements to approximately compute Hamming distances at a lower cost. Finally, we combine such resistive characteristic with a current-based search method to design an analog HAM (A-HAM) that results in faster and denser alternative. Our experimental results show that R-HAM and A-HAM improve the energy-delay product by 9.6× and 1347× compared to D-HAM while maintaining a moderate accuracy of 94% in language recognition.**

***Keywords*-Associative memory; Hyperdimensional computing; Neuromorphic computing; Non-volatile memory.**

I. Introduction

Maintaining a deterministic model of computing ultimately puts a lower bound on the amount of energy and performance scaling that can be obtained. This bound is primarily set by the variability and reliability of the composing devices. It is therefore worth exploring alternative computational models that enable further size and energy scaling by abandoning the deterministic requirement. At the heart of this paper is an alternative model of computing that help to circumvent or redefine these bounds. Hyperdimensional (HD) computing [1] is based on the understanding that brains compute with *patterns of neural activity* that are not readily associated with numbers. However, due to the very size of the brain's circuits, we can model neural activity patterns with points of a high-dimensional space, that is, with hypervectors. When the dimensionality is in the thousands (e.g., D=10,000), it is called hyperdimensional. Operations on hypervectors can be combined into interesting computational behavior with unique features that make them robust and efficient.

HD computing builds upon a well-defined set of operations with random hypervectors, is extremely robust in the presence of failures, and offers a complete computational paradigm that is easily applied to learning problems [1]. Its main differentiation from other paradigms is that data are represented as approximate patterns, which can favorably scale for many learning applications. Examples include analogy-based reasoning [2], latent semantic analysis [3], language recognition [4], [5], text classification [6], gesture recognition [7] and prediction from multimodal sensor fusion [8], [9]. These applications use various encoding operations on sparse or dense hypervectors, but the common denominator of all them is the extensive use of search operation in the associative memory requested for every event of classification or recognition.

In this paper, we propose three architectural designs for hyperdimensional associative memory (HAM). We exploit the holographic and distributed nature of hypervectors to design memory-centric architectures with *no asymmetric error protection* that further allows us to effectively combine approximation techniques in three widely-used methodological design approaches: digital design, memristive-based design, and analog design. We propose a digital HAM (D-HAM) using the CMOS technology, a resistive HAM (R-HAM) using the resistive dense elements, and an analog HAM (A-HAM) using current-based analog search circuitries. These HAM designs exploit approximation techniques including structured sampling and voltage overscaling with bit width optimization for current-based comparators that cooperatively explore a large design space. We assess the benefits and limitations of each design approach by measuring energy, delay, their product, and area for varying parameters including dimensions, number of classes, and classification accuracy. Our experimental results show that R-HAM improves the energy-delay product by 9.6× and the area by 1.4× compared to D-HAM; A-HAM surpasses these improvements to 1347× and 3× respectively, while maintaining a moderate classification accuracy of 94% in language recognition. Nevertheless, A-HAM becomes susceptable to process variations in low voltages.

The rest of paper is organized as follows: Section II provides background in HD computing. Section III describes the

2378-203X/17 $31.00 © 2017 IEEE
DOI 10.1109/HPCA.2017.28

details of the proposed HAM designs. Section IV presents our experimental results. Section V concludes this paper.

II. Hyperdimensional 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 computing. HD computing [1] explores this idea by looking at computing with ultra-wide words – that is, with very high-dimensional vectors, or hypervectors. There exist a huge number of different, nearly orthogonal hypervectors with the dimensionality in the thousands [10]. This lets us combine such hypervectors into a new hypervector using well-defined vector space operations, while keeping the information of the two with high probability.

Hypervectors are holographic and (pseudo)random with i.i.d. components. A hypervector contains all the information combined and spread across all its components in a full holistic representation so that no component is more responsible to store any piece of information than another. These unique features make a hypervector robust against errors in its components. Hypervectors can be manipulated with arithmetic operations, such as *binding* that forms a new hypervector which associates two hypervectors, and *bundling* that combines several hypervectors into a single composite hypervector. The reasoning in HD computing is based on similarity between the hypervectors. This similarity is measured by a distance metric. In this paper, we target an application of HD computing for identifying the language of text samples, based on encoding consecutive letters into a hypervector.

A. Language Recognition: An Example

Recognizing the language of a given text is the first step in all sorts of language processing, such as text analysis, categorization, translation, etc. High-dimensional vector models are popular in natural-language processing and are used to capture word meaning from word-use statistics. These vectors are often called semantic vectors. Ideally, words with a similar meaning are represented by semantic vectors that are close to each other in the vector space, while dissimilar meanings are represented by semantic vectors far from each other [11]. Latent semantic analysis [11] is a standard way of making semantic vectors. It relies on singular value decomposition of a large matrix of word frequencies. It is computational heavy and scales poorly.

Random indexing [3] is an algorithm based on high dimensionality and randomness and it provides a simple and scalable alternative to methods based on principal components, such as latent semantic analysis. Random indexing represents information by projecting data onto vectors in a high-dimensional space. It is incremental and computes semantic vectors in a single pass over the text data. With the dimensionality in the thousands it is possible to calculate useful representations with fast, and highly scalable algorithms. We use random indexing for identifying the source language of text samples by generating and combining letter n-grams – n consecutive sequence of letters.

We accordingly design an architecture for recognizing a text's language by generating and comparing text hypervectors [4], [5]: the text hypervector of an unknown text sample is compared for similarity to precomputed text hypervectors of known language samples – the former is referred to as a query hypervector, while the latter are referred to as learned language hypervectors. The architecture has two main modules: encoding and associative memory. The encoding module projects an input text, composed of a stream of letters, to a hypervector in high-dimensional space. Then this hypervector is broadcast to the associative memory module for comparing with a set of learned language hypervectors. The associative memory returns the language that has the closet match; its energy consumption is about half of the HD architecture [5]. In the following, we describe these two modules in details.

1) Encoding Operations: The encoding module accepts a sequence of letters from a text and computes a hypervector that represents the text. Random indexing generates seed hypervectors that are initially taken from a 10,000-dimensional space and have an equal number of randomly placed 0s and 1s, i.e., $\{0,1\}^{10,000}$ [12]. Such hypervectors are used to represent the basic elements, e.g., the 26 letters of the Latin alphabet plus the (ASCII) space for text inputs. An *item memory* assigns every hypervector to a letter; this assignment is fixed throughout the computation, and formed 27 unique orthogonal hypervectors. The similarity between two hypervectors is measured by Hamming distance denoted as δ[1].

The arithmetic operations on the hypervectors are defined as follows. Binding of two hypervectors A and B is done by component-wise `XOR` and denoted as $A \oplus B$. The result of the operation is new hypervector that is dissimilar to its constituent vectors i.e., $\delta(A \oplus B, A) \approx 5,000)$ for $D = 10,000$; hence `XOR` is well suited for associating two hypervectors. Binding is used for variable-value association and, more generally, for the mapping. Bundling operation is done via component-wise majority function and denoted as $[A+B+C]$. The majority function is augmented with a method for breaking ties if the number of component hypervectors is even. The result of the majority function preserves similarity to its component hypervectors i.e., $\delta([A+B+C], A) < 5,000$. Hence, the majority function is well suited for representing sets. The third operation is a permutation, $\rho(A)$, that rotates the hypervector coordinates. Practically, it can be implemented as a cyclic right-shift by one position. The permutation operation generates a hypervector, which is unrelated to the given hypervector $\delta(\rho(A), A) \approx 5,000$. This operation is commonly used for storing a sequence of tokens in a single hypervector. In geometrical sense, the permutation rotates the hypervector in the space.

[1]Hamming distance, distance, and δ are used interchangeably throughout this paper.

For example, the sequence trigram ($n = 3$) of a-b-c, is stored as the following hypervector, $\rho(\rho(A) \oplus B) \oplus C = \rho(\rho(A)) \oplus \rho(B) \oplus C$. This efficiently distinguishes the sequence a-b-c from a-c-b, since a rotated hypervector is uncorrelated to all the other hypervectors. The encoding module bundles all the trigram hypervectors across the input text to generate the text hypervector as the output of encoding. This encoding is used for both training and testing. During training when the language of the input text in known, we refer to the text hypervector as a learned language hypervector. Such learned hypervectors are stored in the associative memory. During testing, we call the text hypervector as a query hypervector. The query hypervector is sent to the associative memory module to identify its source language.

2) Associative Memory: We consider 21 European languages [13], consequently at the end of training phase, we will have 21 language hypervectors, each of which is stored separately in a row of the associative memory. Determining the language of an unknown text is done by comparing its query hypervector to all the learned hypervectors that results in 21 Hamming distances. Finally, the associative memory finds the minimum Hamming distance and returns its associated language as the language that the unknown text has been written in.

As mentioned earlier, HD computing offers a complete and universal computational paradigm. For instance, the aforementioned algorithm can be reused to perform other tasks such as classification of news articles by topic with similar success rates [6]. While these applications have a single streaming input, other applications with analog and multiple sensory inputs can equally benefit from HD computing [8], [9], [7].

B. Robustness

HD computing is amazingly tolerant to errors. The random hypervector seeds are independent and identically distributed, a property that is preserved by the encoding operations (binding, bundling, and permutation) performed on them. Hence, a failure in a component is not "contagious" [5].

Figure 1 shows the classification accuracy as a function of number of bits error in computing Hamming distance. As shown, HD computing still exhibits its *maximum* classification accuracy of 97.8% with up to 1,000 bits error in computing the distance (i.e., when up to 10% of hypervector components are faulty). We exploit such robustness property of HD computing to design efficient associative memory that can tolerate the error in any part of a hypervector. Further increasing the error in distance metric to 3,000 bits, slightly decreases the classification accuracy to 93.8%. We call this range of the classification accuracies as *moderate* accuracy that has up to 4% lower accuracy compared the the maximum of 97.8%. Accepting the moderate accuracy opens more opportunity for aggressive optimizations. However, increasing the error to 4,000 bits reduces the classification accuracy below 80%.

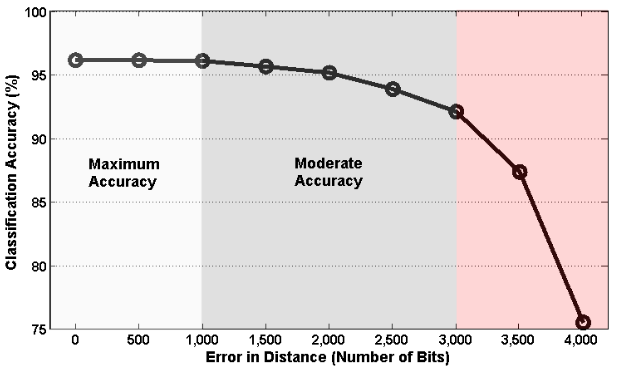


Figure 1. Language classification accuracy with wide range of errors in Hamming distance using $D = 10,000$.

III. Hyperdimensional Associative Memory

The state-of-the-art associative memories require improvements for comparing vectors with a dimensionality in thousands [5], [14], [15], [16]. They also lack techniques for uniformly tolerating errors in any vector component and hence cannot efficiently exploit the holographic feature of hypervectors. Further, most of them are implemented as a content addressable memory (CAM) without ability of finding the minimum distance [14], [15], [16]. In this section, we propose HAM architectures for three methodological design approaches of digital, memristive, and analog with ability of finding the minimum distance.

A. Digital HAM

Conventional CAMs typically check the availability of a input query pattern among the stored (or learned) patterns, however finding the closest pattern is of our need for reasoning among hypervectors. We propose a digital CMOS-based hyperdimensional associative memory, called D-HAM, that provides faster search compared to [5]. After the training phase, D-HAM stores a set of learned hypervectors in a CAM. For an input query hypervector, D-HAM finds a learned hypervector that has the nearest Hamming distance to the query.

Figure 2 shows the structure of proposed D-HAM consisting of two main modules: (1) CAM: it forms an array of $C \times D$ storage elements, where C is the number of hypervectors as the distinct classes and D is the dimension of a hypervector. During a classification event, each learned hypervector is compared with an input query hypervector using an array of `XOR` gates. An `XOR` gate detects the similarity between its inputs by producing a 0 output as the match and a 1 output as the mismatch. Therefore, in every row the number of `XOR` gates with an output of 1 represents Hamming distance of the input query hypervector with the corresponding hypervector that is stored in the row. (2) Distance computation: it composed of parallel counters and comparators that compute the distances using the outputs of `XOR` gates. D-HAM requires a set of C counters each with $\log D$ bits. Each counter is assigned to a row, and iterates

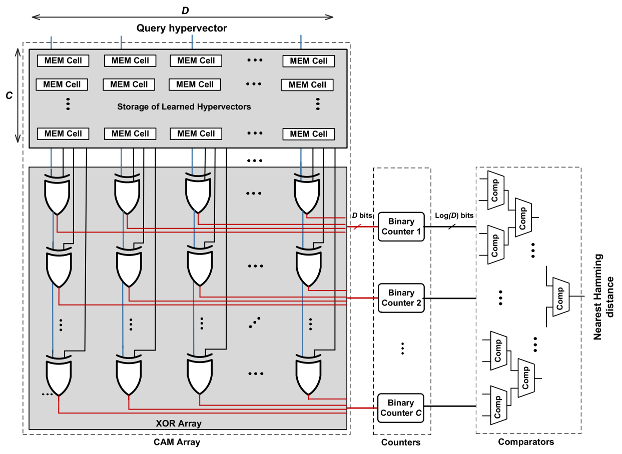


Figure 2. Overview of D-HAM.

Table I
ENERGY AND AREA PARTITIONING FOR D-HAM.

D-HAM	Modules	Area (mm^2)	Energy (pJ)
D=10,000	CAM Array	15.2	4976.9
	Counters and comparators	10.9	1178.2
d=9,000	CAM Array	13.7	4479.2
	Counters and comparators	10.2	1131.1
d=7,000	CAM Array	10.6	3483.8
	Counter and comparator	8.3	883.6

through D output bits of the XOR gates to count the number of 1s. The value of a counter determines the distance of the input query hypervector with the corresponding stored hypervector in the row. Finally, D-HAM requires to find the minimum distance among the the C counter values. It structures a binary tree of comparators with height of $\log C$ to find a hypervector with the minimum distance from the input query hypervector.

The D-HAM structure takes advantage of a scalable digital design that can be easily extended arbitrary dimensions. Table I shows the partitioning of the energy consumption and the area of D-HAM configured for $C = 100$ and $D = 10,000$. This D-HAM requires a CAM array including $100 \times 10,000$ XOR gates, 100 counters, and 99 comparators of 14 bits. The result shows that D-HAM consumes 6155.2 pJ energy for each query search, where the CAM array consumes 81% of the total energy. The CAM array is the energy bottleneck since each component of the query hypervector needs to be compared with the same component of the learned hypervectors. This imposes a high switching activity due to nature of XOR gates consuming considerable amount of energy.

1) Accuracy-Energy Tradeoff in D-HAM: To address this energy issue in D-HAM, we exploit the distributed property of hypervectors that enables us to accurately compute Hamming distance from an arbitrary subset of hypervector components ($d < D$). This enables D-HAM to applies sampling on the hypervector with i.i.d. components. As shown in Figure 1, such a sampling ratio can impact the classification accuracy [17]. To meet the maximum classification accuracy, D-HAM ignores 1,000 bits and computes Hamming distance for the rest of $d = 9,000$ bits. In this way, D-HAM ensures the error-free computation of Hamming distance on 90% of the hypervector bits while intentionally eliminating the 10% of the bits. Further, ignoring the dimensions up to 3,000 bits (i.e., $d = 7,000$) guarantees the moderate classification accuracy (see Figure 1). Such sampling results in energy saving in D-HAM which is linearly related to the size of sampling: 7% (or 22%) energy saving is achieved with $d = 9,000$ (or $d = 7,000$) for the maximum (or moderate) classification accuracy compared to baseline D-HAM with $D = 10,000$.

We should also note that other CMOS-based digital implementations of associative memories have similar energy inefficiencies. For instance, SRAM-based CAMs consume 8× higher energy than the SRAM cells for each search operation [18]. In addition, like all CMOS-based designs, these CAMs also have large idle power. High energy consumption of such ternary CAMs prevents their application for large pattern classification and ultimately limits their usage to networking and IP lookup [19]. In the next section, we show how this problem can be addressed by using new memory technologies.

B. Resistive Memory

Higher density and lower energy consumption of non-volatile memories (NVM) open new opportunities to design energy-efficient CAMs. NVM-based CAMs can be designed using different NVM elements such as resistive memory (i.e., memristor). Resistive CAMs have comparable read operation to SRAMs enabling their extended application to approximate computing [14]. In this work, we use CAMs based on the resistive elements and address their endurance issue by limiting the write stress only to once for each training session.

The general structure of a resistive element (memristor) is based on metal/oxide/metal. Two metal layers (e.g., Pt) sandwich an oxide layer based on Ta, Ti and HF [20], [21], [22]. The metal/diode connection usually shows Ohmic behavior. The data stores on cell based on the memristor resistance state. The device can switch to ON mode by applying a negative bias and to OFF mode by applying a positive voltage across the device. The read operation can perform by applying a small voltage across source line (SL) and read bitline (BL) nodes and sensing the current or the dropped voltage using a sense amplifier.

C. Resistive HAM

In this section, we describe our design method to utilize resistive elements in HAM called a resistive HAM or R-HAM. Figure 3(a) shows the overview of the proposed R-HAM, consisting of two main modules. (1) Resistive CAM: it stores the learned hypervectors by using a crossbar of resistive cells. The resistive CAM is partitioned to M stages as shown in Figure 3(b) each containing D/M bits with C rows, where D and C are the hypervector dimension and the number of classes, respectively. Each CAM stage finds the distance between the query hypervector and the learned hypervectors in parallel. (2) Distance computation: R-HAM uses a set of parallel counters to count the number of Hamming distances in all partial CAM stages corresponding to each row. This counter is different from the conventional binary counters that D-HAM uses because the CAM stages in R-HAM generate a non-binary code (See Figure 3(c)). This new coding has a lower switching activity compared to the dense binary coding. More details about the functionality and implementation of this module will be explained in the next section. Finally, R-HAM uses similar comparators as D-HAM to find a row with the minimum Hamming distances.

Although, both D-HAM and R-HAM use similar counters and comparators, R-HAM achieves higher energy efficiency. As mentioned in Section III-A, D-HAM spends 81% of the total energy consumption and 58% of the total area for storing and comparing the hypervectors in the CAM array which is replaced with the dense memristive crossbar in R-HAM. The memristive CAM improves both energy and area of R-HAM: (1) The energy is reduced due to the lower switching activity in the crossbar compared to D-HAM which uses `XOR` gates to find mismatches. Table II shows the average switching activity of D-HAM and R-HAM for different block sizes. R-HAM shows lower switching activity in large block sizes, and exhibits about 50% lower switching activity compared to D-HAM with blocks of 4 bits. Further, the resistive CAM array can operate in the lower supply voltage bringing additional energy saving. (2) The area is reduced by tightly integrating the storage array and the mismatch finding logic in the same crossbar. In contrast to D-HAM which uses a large `XOR` array to determine the mismatches, R-HAM uses high-density memristive crossbars for both storage and partial determination of mismatches.

1) Nearest Distance in Resistive Crossbars: Our R-HAM finds the nearest distance by efficiently using timing property of memeritive crossbars. R-HAM consists of a CAM array where all cells in a row share the same match line (ML) to represent a hypervector. The search operation has two main phases: precharging and evaluation. There is a set of C row drivers that precharge all the MLs before the search operation. In the evolution, a set of input buffers distribute the components of the input query among all the rows through D vertical bitlines (BLs). The buffer role is to strengthen the input signal such that all the rows receive the signal at the same time. During the evaluation, any cell with a value differing from the input query component discharges the ML. Therefore, all MLs will be discharged, except the ML for the row that contains all matched cells. This matched row can be detected by the sense amplifier circuitry and by sampling the ML voltage at certain time.

Table II
AVERAGE SWITCH ACTIVITY OF D-HAM AND R-HAM.

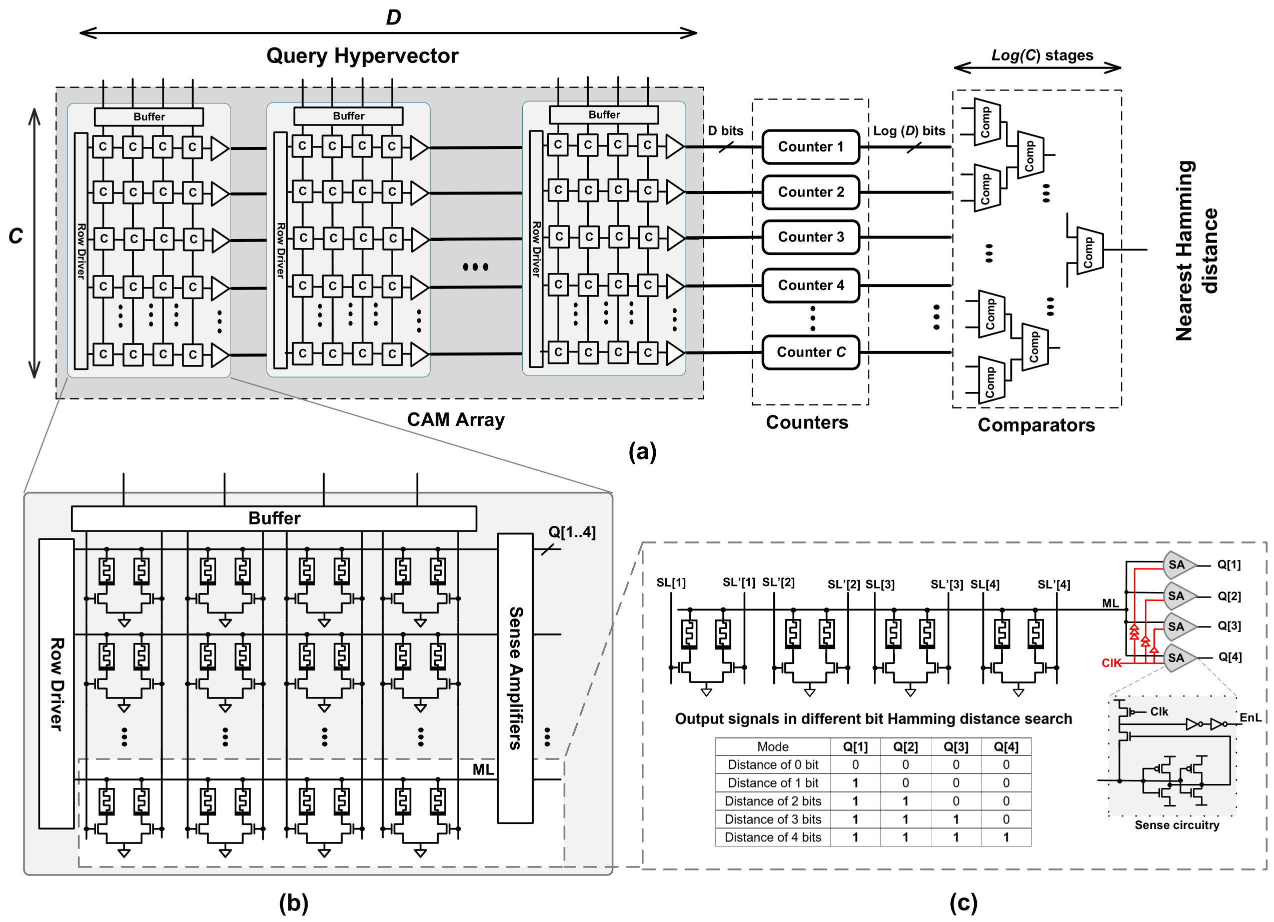
Block size	R-HAM	D-HAM
1 bit	25%	25%
2 bits	21.4%	25%
3 bits	18.3%	25%
4 bits	13.6%	25%

Figure 4(a) shows the normalized ML discharging voltage curves during the search operation for different distances (i.e., the number of mismatches on the ML). The ML discharging speed depends on the distance value. For example, a row that has a Hamming distance of 2 from the input query discharges ML about 2× faster than a row with a Hamming distance of 1. This timing characteristic can be used to identify the distance of the query hypervector with the learned hypervectors. However, there is not a linear dependency between the speed of ML discharging and Hamming distances. As Figure 4(a) shows, the first mismatch usually has higher impact on ML discharging compared to the last mismatch. Because the ML discharging current saturates after having the first few mismatches, many later mismatches do not change the ML discharging speed. For example, there is a distinguishable time difference between Hamming distances of 1 and 2 while Hamming distances of 4 and 5 have similar ML discharging time. In this experiment, each row has solely 10 bits that clearly indicates the limitation of the method for higher dimensionality. Such restriction has been already observed that limits the approximate search for resistive CAMs with small dimensionality of 64 bits [14].

To address this non-uniform ML discharging time, we split the R-HAM array to M shorter blocks. Among various configurations, we have observed that the maximum size of a block can be 4 bits for accurate determination of the different distances. To further alleviate the current saturation issue, we use memristor devices with reported large ON resistance [23] that provide stable ML voltage for the better distinction between the distances at a cost to slower search operation. Figure 4(b) shows the ML discharging time of the 4 bits block for different distances. This figure shows that the timing difference between the distances become approximately uniform.

We accordingly design a sense amplifier for R-HAM that can identify the difference between Hamming distances of 1, 2, 3 and 4 by measuring the ML discharging time. As shown in Figure 4(b), sampling at T_0 corresponds to ML without any mismatches (i.e., a Hamming distance of 0). Similarly sampling at T_1 detects a row with a Hamming distance of 1 and 0. Figure 3(c) shows the structure of the sense amplifier with the tuned sampling time to detect each distance. Our design uses four distinct sense amplifiers to detect Hamming distances of 0, 1, 2, and 3. Then, based on table shown in Figure 3(c), it can identify the number of mismatches on each row by generating a non-binary code. We use a buffer to generate a small delay (≈0.1 ns) on the clock of the sense amplifiers. To justify this delay, we change the size of buffer to set the buffer delay. To guarantee the correct functionality in the presence of variations, we design the CAM and sense circuitry considering 10% process variation on the transistors (size and threshold voltage) and the resistance values.

Figure 3(c) shows inside of a block of R-HAM with 4 bits. To compute a distance from the outputs of these blocks a counter counts the number of mismatches in all partial blocks. This counter is designed to work with the coding



Output signals in different bit Hamming distance search

Mode	Q[1]	Q[2]	Q[3]	Q[4]
Distance of 0 bit	0	0	0	0
Distance of 1 bit	1	0	0	0
Distance of 2 bits	1	1	0	0
Distance of 3 bits	1	1	1	0
Distance of 4 bits	1	1	1	1

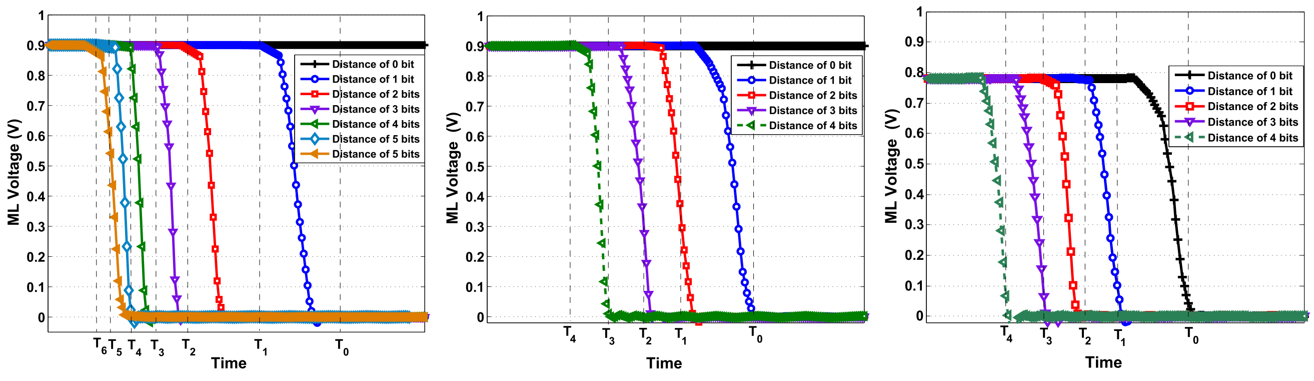
Figure 3. Overview of R-HAM: (a) Resistive CAM array with distance computation; (b) A 4 bits resistive block; (c) Sensing circuitry with non-binary code generation.

produced as the output of the sense amplifiers. The proposed encoding decreases the bit difference between the output signals. For example, patterns with distance of 3-bits and 4-bits have three bits difference in the binary representation (0011vs0100), while encoding reduces this switching to single bit (1110vs1111). Mathematically considering all possible matching combinations shows that proposed encoding significantly reduces R-HAM switching activity compared to D-HAM especially when using larger block sizes (See Table II).

2) Accuracy-Energy Tradeoff in R-HAM: To exploit the robustness of HD computing for improving energy efficiency, R-HAM supports tow techniques. First, it applies a sampling technique similar to D-HAM by ignoring up to 10% of the blocks, i.e., accepting up to 1,000 bits error in the distance (See Figure 1). These 250 blocks (out of the total 2500 blocks) can be directly excluded form the R-HAM design to save energy while meeting the maximum classification accuracy. Further ignoring the number of blocks to 750 decreases the classification accuracy to the moderate level.

Second technique leverages the holographic property of the hypervectors by intentionally distributing the erroneous bits to the large number of blocks rather than jamming them into few blocks. To do so, R-HAM overscales the supply voltage of every block to 780 mV such that a block is restricted to undergo a maximum of one bit error in Hamming distance (See Figure 4(c)). With this technique, 40% (or 100%) of the total blocks can operate with a lower voltage while providing the maximum (or the moderate) classification accuracy. As a results, these blocks quadratically save the energy while computing a distance metric in which the cumulative effect of their errors is acceptable for the classification using HD computing. To implement the voltage overscaling, we use an energy-efficient and fast voltage supply boosting technique [24].

Figure 5 compares the energy saving of R-HAM using the two techniques of sampling and voltage overscaling. Targeting the maximum accuracy, the sampling technique achieves a relative energy saving of 9% by turning 250 blocks off, while the voltage overscaling technique achieves almost 2× higher saving by reducing the voltage for 1000 blocks. This trend of relative energy saving is consistent



(a) 10 bits CAM (b) 4 bits CAM w/o voltage overscaling (c) 4 bits CAM with voltage overscaling

Figure 4. Match line (ML) discharging time and its relation to detecting Hamming distance for various CAMs.

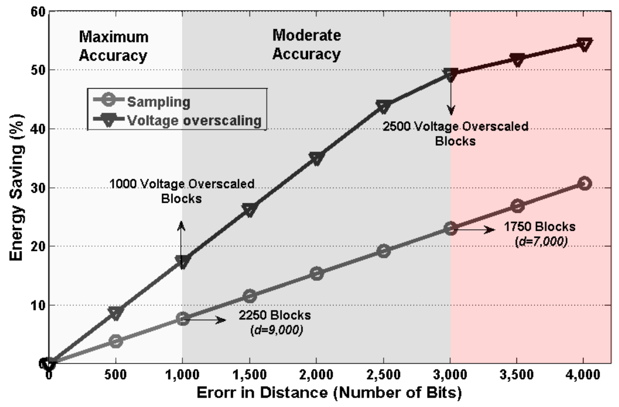


Figure 5. Energy saving of R-HAM using structured sampling versus distributed voltage overscaling.

with targeting the moderate accuracy: 22% by turning off 750 blocks, and 50% by reducing the voltage for all the 2500 blocks. However, R-HAM can not maintain such linear energy saving beyond 2,500 bits error in the distance. This is because all the blocks are already under the voltage overscaling, and accepting more than 2,500 bits error requires some blocks to accept a Hamming distance of 2. The energy gain that a block of R-HAM can achieve by accepting a distance of 2 (by operating at 720 mV) is very similar to the distance of 1 (i.e., 780 mV).

D. Analog HAM Design

We propose an analog HAM (A-HAM) to exploit the timing characteristics of the ML by observing discharging current to compute its distance. Figure 6(a) shows the overall architecture of the proposed A-HAM. We use a memristor device with high OFF/ON resistance ratio [25] to design a ternary CAM (TCAM) cell. The A-HAM consists of an array of TCAM cells forming a $D \times C$ crossbar similar to R-HAM. The A-HAM design searches for the query hypervector among all TCAM rows in parallel and then compares their currents using a set of Loser Takes All (LTA) blocks [26]. The LTA blocks form a binary tree with height of $\log C$.

The ML discharging current is related the number of mismatched cells. A buffer block senses the ML current and sends it to LTA block to be compared with the next row. A row with a large number of mismatches has a higher amount of discharging current. The goal is to find a row that has the minimum number of mismatches and thus the minimum discharging current. Therefore, the binary tree blocks of LTA compares the output current of every two neighbor rows to find the row that has the minimum Hamming distance with the query hypervector.

However, such current comparison cannot be directly scaled to large dimensions. This is because the discharging current of the rows will be very close, hence the precision of the LTA cannot determine the minimum distance. Moreover, both ML discharging current and the LTA blocks are sensitive to the process and the voltage variations. To address these issues, we propose ML stabilizer and multistage search operation techniques in the following sections.

1) Mtach Line Stabilizer: In conventional TCAMs, an ML current saturation, described in the following, is the main limitation in identifying the number of mismatches. For instance, when a single cell in a row does not match with an input data, the ML discharges with I1 current. However, having two mismatched cells does not result in the same I1 leakage current on every cell causing a total discharging current of less than 2*I1. This non-linearity of current-voltage dependency is more pronounced in large dimensions, where having $D > 7$ cells has minor impact on the total ML discharging current. This is so-called current saturation and occurs due to the ML voltage-current dependency. In the current saturation, a large number of mismatched cells drops the ML voltage immediately and decreases the passing current through each cell. This makes detecting the exact number of mismatches challenging.

To identify the number of mismatches on a row, we need to have a fixed supply voltage on the ML during the search operation. In this condition, the ML voltage depends on the number of mismatched cells. In contrast to the conventional TCAMs which work based on ML discharging voltage, our

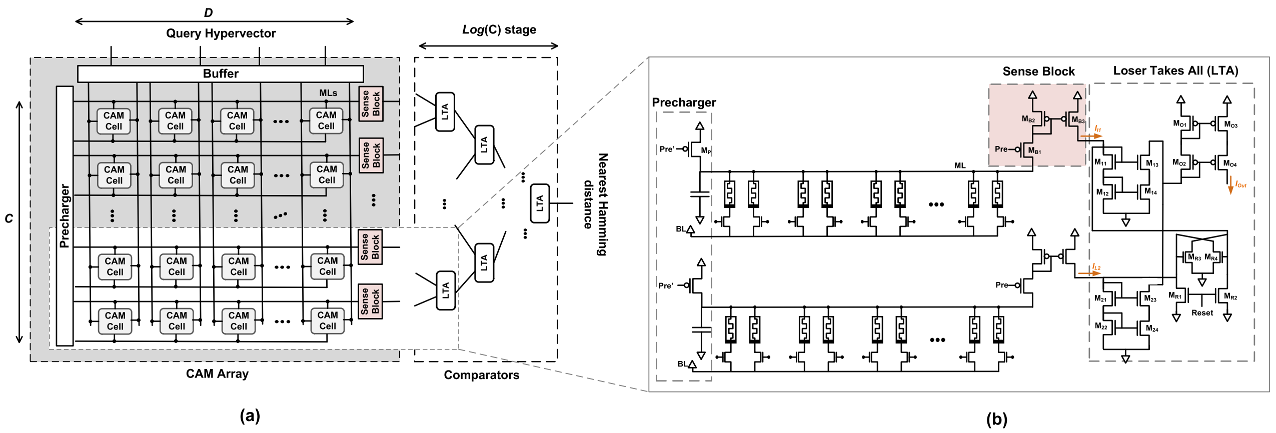


Figure 6. Overview of A-HAM: (a) Resistive CAM array with LTA comparators; (b) Circuit details of two rows.

design is current-based. A-HAM stabilizes the ML in a fixed voltage during the search operation and identifies the number of mismatched cells by tracking the current passing through the sense circuitry.

Figure 6(b) shows two rows of the proposed A-HAM. Before the search operation, the ML is charged using a precharge transistor (M_p). During the precharge mode, all A-HAM cells deactivate by connecting select lines of all cells to zero voltage. After precharge mode, the search operation starts on the TCAM cells. The select lines activate TCAM cells by the input data. Any cell that has a different value with the select line value discharges the ML. In this case the M_{B1} transistor is activated and tries to fix the ML voltage to the supply voltage. Thus the currents of all mismatched cells pass through the M_{B1} and M_{B2} transistors. The input buffer mirrors the M_{B1} current to a branch containing the M_{B3} transistor. This stage sends the data to the LTA block to be compared with the discharging current of the next row. The value of I_{L1} current linearly depends on the number of mismatches in each row. The precision of the detection and the number of mismatches depend on the accuracy of the LTA block.

The LTA block accepts two input currents and returns a line with the lower current (See Figure 6(b)). The LTA block is based on the current mirror and a reset circuitry which compares the currents in a different resolution. The bit resolution of this comparison identifies the number of mismatches that our design can detect.

2) Multistage A-HAM: The ML discharging current increases the energy consumption of A-HAM to higher than that of a conventional TCAM. To address this issue, we use resistive devices with high ON resistance [25] to reduce the discharging current of each missed cell. However, the large ON resistance imposed the following issues: (1) It degrades the sense margin by decreasing the ON to OFF current ratio. Therefore, we use the memristor devices with very large OFF/ON resistance [25] to provide enough sense margin and stability. (2) In addition, the large ON resistance slows down the search operation by increasing the response time ($T = RC$). This creates a tradeoff between the energy consumption and search speed. Our evaluation shows that A-HAM with $R_{ON} \sim 500K$ and $R_{OFF} \sim 100G$ is able to identify a Hamming distance of up to 512 bits using the LTA blocks with 10 bits resolution. However, increasing the dimension limits the distance difference that can be identified across the rows. Figure 7 shows the the minimum detectable Hamming distance with increasing the dimension. For D=256 and lower, A-HAM provides a resolution of one bit in comparing Hamming distances of various rows. Increasing D=10,000 increases the minimum detectable distance to 43 bits, i.e., A-HAM using a single stage can not differentiate between Hamming distances lower than 42. This precision loss in comparing the distances can be improved by the following multistage technique.

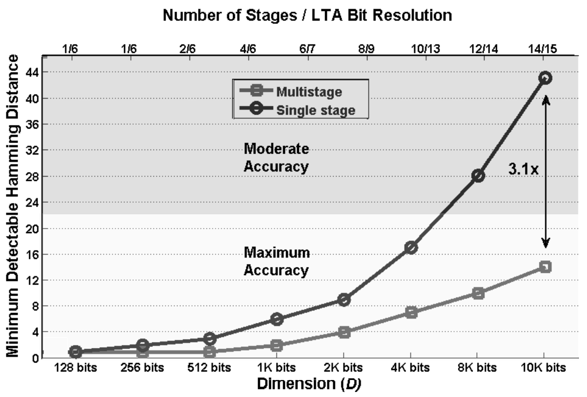


Figure 7. Minimum detectable distance in A-HAM.

We observed that the ML voltage cannot be fixed during the search operation for the large dimensions. This degrades the resolution of identifying the distances for the hypervectors with D=256 and higher. Even using the LTA with higher resolution (>10 bits) cannot provide the acceptable accuracy. To address this issue, we split the search operation to multiple shorter stages and calculate the mismatched current of each part separately. Then, an additional current mirror circuit adds these partial currents (I_1 and I_2 currents

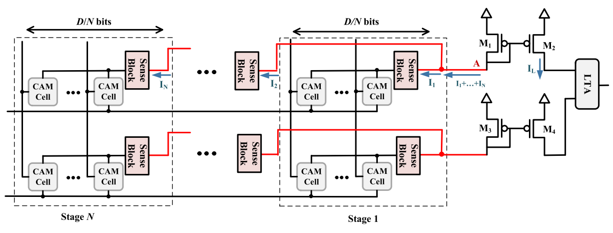


Figure 8. Multistage A-HAM architecture.

in the example of Figure 8) in node A. The LTA compares the IL currents for different rows. In A-HAM, the minimum detectable distance can be controlled by the number stages and the bit width of the LTA blocks.

Figure 7 shows the minimum distance that can be identified by A-HAM using multi stages; the top X-axis shows the number stages and the LTA bit resolution that is used for achieving such distance resolution. This multistage search technique enables the extension of the minimum detectable distance of 1 to D=512. For D=10,000, the minimum detectable Hamming distance is improved to 14 bits using 14 stages. This precision in distinguishing the distances is sufficient to maintain the classification accuracy. We observe that the minimum Hamming distance among any learned language hypervector with the other 20 learned language hypervectors is 22; and the next minimum distance is 34. Intuitively, hypervector within a language family should be closer to each other than hypervectors for unrelated languages. Therefore, the LTA blocks with the minimum detectable Hamming distance lower than 22 bits does not impose any misclassification (the boarder is shown in Figure 7). However, increasing the dimensionality, or the variations in the process and the voltage of the LTA blocks can increase the minimum detectable distance that degrades the classification accuracy discussed in Section IV-F.

3) Accuracy-Energy Tradeoff in A-HAM: A-HAM has three main sources of the energy consumption: the resistive CAM, the sense circuitry, and the LTA blocks. For the CAM with large number of rows, the input buffers slow down the search operation and dominate the CAM energy. The sense circuitry fixes the ML voltage level and works as the sense amplifier. Its energy consumption is related to the average time that the search operation is continued. Our results show that LTA blocks are the main source of A-HAM energy consumption in large sizes. The LTA bit width can be reduced to lower the energy consumption at a cost to loss of classification accuracy. For D=10,000, we optimize the bit with of the LTA blocks to 14 bits (and 11 bits) such that A-HAM with 14 stages can meet the maximum (and the moderate) classification accuracy while improving its relative energy-delay product by 1.3× (and 2.4×).

IV. Experimental Results

In this section, we first present our experimental setup for the application of language recognition and its dataset. We then compare energy, delay, area, scalability and accuracy of the proposed D-HAM, R-HAM, A-HAM.

Table III
Recognition accuracy as a function of D.

Dimensionality (D)	256 bits	512 bits	1K bits	2K bits	4K bits	10K bits
D-HAM and R-HAM	69.1%	82.8%	90.4%	94.9%	96.9%	97.8%
A-HAM	69.1%	82.8%	90.4%	94.9%	96.5%	97.3%

A. Language Recognition Dataset

We consider an application for recognition of 21 European languages. The sample texts are taken from the Wortschatz Corpora [13] where large numbers of sentences for a selected languages are available. We train each language hypervector based on about a million bytes of text. To test the ability of identifying the language of unseen text samples, we select test sentences from Europarl Parallel Corpus [27] as an independent text source. This corpus provides 1,000 samples of each language, and each sample is a single sentence. The accuracy recognition metric used throughout this paper is the percentage of these 21,000 test samples that are identified correctly. This accuracy is measured as the microaveraging that gives equal weight to each per-sentence classification decision, rather than per-class.

B. Experimental Setup for HAM

For D-HAM, we use a standard digital ASIC flow to design dedicated hardware. We describe D-HAM, in a fully parameterized manner, using RTL System-Verilog. For the synthesis, we use *Synopsys Design Compiler* with the TSMC 45 nm technology library, the general purpose process with high V_{TH} cells. The design is optimized for a cycle time of 160 ns. We extract its switching activity during pos-synthesis simulations in *ModelSim* by applying the test sentences. Finally, we measure their power consumptions using *Synopsys PrimeTime* at (1 V, 25 °C, TT) corner.

We design circuit-level R-HAM and A-HAM using *HSPICE* simulator in 45 nm technology. For the memristive crossbar, we use devices with large OFF/ON resistance ratio to design the stable CAM with large sense margin [28]. For R-HAM, this crossbar is voltage overscaled to 0.78 V while the rest of components are operated at nominal 1 V. However, for A-HAM the crossbar is operated at 1 V due to its sensitivity to lower voltages while the LTA analog blocks work at 1.8 V. The LTA blocks are designed considering 10% process variations on threshold voltage and transistor size, using 5000 Monte Carlo simulations.

C. Scalability of HAM

Here, we measure the energy, the search delay, and their product to assess the scalability of the HAM designs for various configurations of the dimension (D) and the number of classes (C).

1) Dimensions: Table III lists the classification accuracy for various D. This indicates the net effect of dimension reduction on the classification accuracy when no approximation techniques is applied. The three designs exactly exhibit the same classification accuracy for D=2,000 and lower. However, for the higher dimensions A-HAM shows slightly lower classification accuracy: 0.5% lower accuracy with D=10,000. This lower accuracy of A-HAM is caused by the precision of the LTA blocks that can detect a minimum Hamming distance of 14 bits with D=10,000 (See. Figure 7). This issue can potentially limits the scalability of A-HAM for the larger dimensions discussed in Section IV-F.

The dimensionality of hypervectors has direct impact on the search delay and the energy. Figure 9 shows the energy consumption, the search delay, and the energy-delay product of D-HAM, R-HAM and A-HAM when D changes from 512 to 10,000. There is no approximation and each dimension results in its maximum accuracy listed in Table III. Our result shows that the energy consumption of D-HAM and R-HAM linearly increase with the number of dimensions. Indeed, larger dimension in D-HAM and R-HAM slows down the search operation because of the longer interconnect size; the bit width of counters and comparators grows logarithmically with the number of dimensions. In A-HAM, the LTA block is the dominant term of energy consumption. For each dimension, we set the number of stages and the resolution of the LTA blocks that are shown in Figure 7. A-HAM shows the slowest rate of increase in the energy and the delay for the higher dimensionalities, because A-HAM tunes its accuracy by solely changing the resolution of the LTA blocks that does not require significant extra hardware overhead. Our results show that by increasing the dimensionality by 20×, the energy (or the delay) of the HAM design is increased by: 8.3× (2.2×) in D-HAM, 8.2× (2.0×) in R-HAM, and 1.9× (1.7×) in A-HAM.

We should note that a digital HD computing with D=10,000 and without any approximation technique enables 53% energy saving compared to a conventional machine learning method [5]. However the robustness can be weakened against the approximation-induced errors by reducing the dimensionality.

2) Number of Classes: Figure 10 shows the energy consumption of the HAM designs with D=10,000 while using different C. For each configuration, we generate C random hypervectors that resemble the learned hypervectors by having equal number of randomly places 0s and 1s. All the HAM designs with the larger C require the larger input buffers to distribute the input query hypervector among all the classes at the same time that results in higher energy consumption. D-HAM and R-HAM with the larger number of classes require larger bitwise counters and comparators to find the nearest distance among the classes. Comparing D-HAM with R-HAM shows that D-HAM has higher energy and delay for larger C since its bitwise comparisons are done in the CAM array that dominates the energy consumption. However, such CAM array in R-HAM consumes less energy than its counters and comparators due to the lower switching activity and supply voltage.

Increasing C has the maximum impact on the energy and the delay of A-HAM compared to D-HAM and R-HAM. In A-HAM the LTA blocks are the dominant term of energy consumption. The number of LTA blocks and the search delay of LTA structure increase linearly and logarithmically with C. Therefore, the energy consumption of A-HAM is significantly impacted by the number of classes. By increasing C from 6 to 100, the energy (or the delay) of the HAM design is increased by: 12.6× (3.5×) in D-HAM, 11.4× (3.4×) in R-HAM, and 15.9× (4.4×) in A-HAM.

D. Energy Saving

Figure 11 shows the energy-delay product of R-HAM and A-HAM normalized to D-HAM for the errors in Hamming distance. In D-HAM, the energy-delay improves linearly by increasing the errors as D-HAM excludes more dimensions during the distance calculation. The R-HAM shows a higher rate in energy-delay saving compared to D-HAM thanks to applying the voltage overscaling on more blocks. This saving rate is faster for A-HAM by reducing the resolution of the LTA blocks. Targeting the maximum (or the moderate) classification accuracy, R-HAM achieves 7.3× (9.6×) and A-HAM achieves 746× (1347×)lower energy-delay product compared to D-HAM. Overall, A-HAM is highly amenable to be used when a lower classification accuracy is required: by switching from the maximum accuracy to the moderate accuracy A-HAM achieves 2.4× lower energy-delay product, while the R-HAM does it for 1.4×. Such improvement in A-HAM is due to the faster search delay with the LTA blocks with low bitwidth resolution. However, the search latency in R-HAM does not change with lower accuracy since the voltage overscaling can be solely applied to the CAM blocks.

E. Area Comparison

The area comparison of D-HAM, R-HAM and A-HAM using D=10,000 and C=100 is shown in Figure 12. D-HAM consumes most of its area in the CAM array for the bit level comparison; it is also penalized by its interconnect complexity. The area of R-HAM is 1.4× lower than D-HAM because of using high density memeristive elements in the CAM design. However, R-HAM cannot fully utilize such dense technology as it requires to insert digital counters and comparator for every 4 bits block. A-HAM resolves this issue by using the current-based searching with analog circuitry that allows every CAM stage to include ≈700 memristive bits. Overall, A-HAM achieves 3× lower area than D-HAM and its LTA blocks occupy 69% of the total A-HAM area.

F. Limitations

In a nutshell, among the HAM designs, A-HAM exhibits the best energy-delay scaling with both increasing the dimensionality (See Figure 9) and lowering the classification accuracy (See Figure 11). A-HAM also surpasses other designs in the area (See Figure 12). However, R-HAM shows slightly lower rate of increasing energy-delay with increasing the number of classes (See Figure 10). Nevertheless,

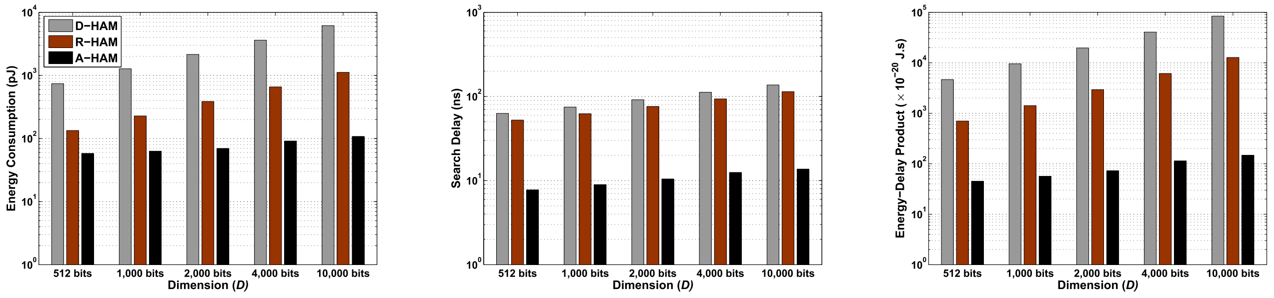


Figure 9. Impact of scaling D in energy consumption (pJ), search delay (ns) and energy-delay product with C=21.

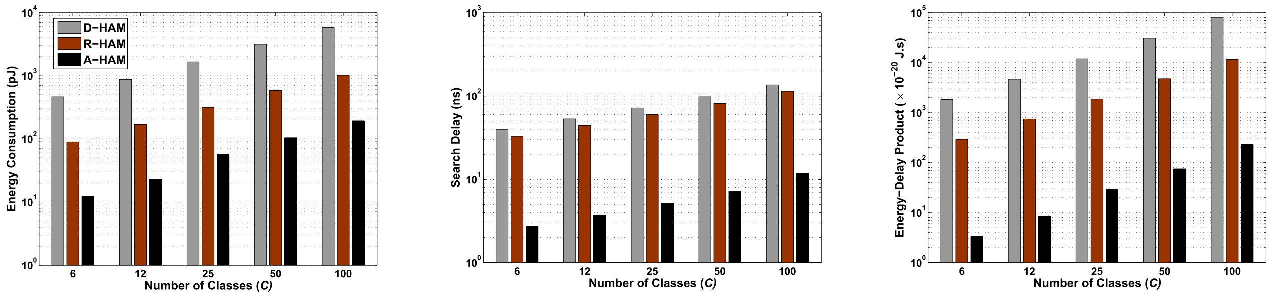


Figure 10. Impact of scaling C in energy consumption (pJ), search delay (ns) and energy-delay product with D=10,000.

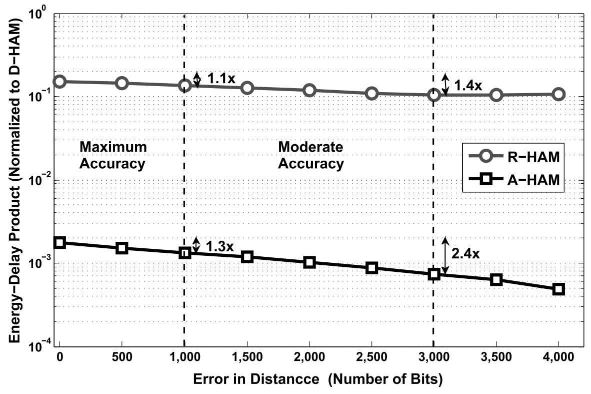


Figure 11. Energy-delay of the HAMs with accuracy.

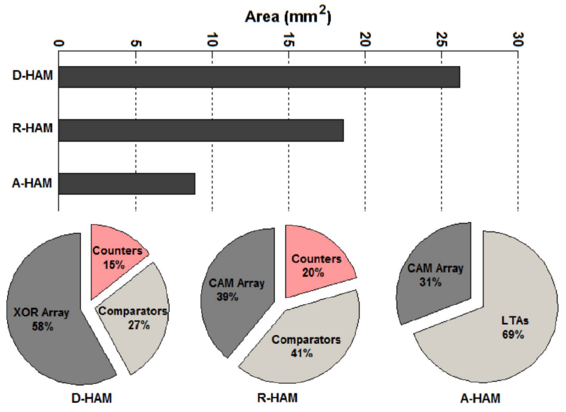


Figure 12. Area comparison between the HAMs.

R-HAM cannot fully exploit the high density of memristive elements since its digital counters and comparators have to be interleaved among the 4 bits blocks of the crossbar; R-HAM is also very sensitive to any voltage variation because the crossbar is already voltage overscaled to 0.78 V to accept 1 bit mismatch among the 4 bits (See Figure 4(c)). In the following, we assess the limitations of A-HAM.

In A-HAM, the LTA capability to detect the minimum Hamming distance can be significantly affected by the variations in process and voltage. To assess such susceptibility, we model the variations on the transistors length and the threshold voltage using a Gaussian distribution with 3σ of 0% to 35% of the absolute parameters value [29]; we also consider 5% and 10% variation on the supply voltage of the LTA blocks that reduces the supply voltage from the nominal 1.8 V to minimum of 1.71 V and 1.68 V, respectively. Figure 13 shows that increasing the process variations specially for the large voltage variations significantly reduces the minimum detectable Hamming distance of the LTA blocks. In the lower voltages, the process variation has more destructive impact on the LTA detectable Hamming distance compared to the nominal voltage. As shown, A-HAM with the nominal supply voltage and more than 15% process variation could degrade the classification accuracy below the moderate level; this also holds for 5% (or 10%) voltage variation combined with more than 10% (or 5%) process variation. Considering the 35% process variation, A-HAM with the nominal voltage, 5% and 10% voltage variations achieves 94.3%, 92.1%, and 89.2% classification accuracy, respectively. This might limit the usage of A-HAM

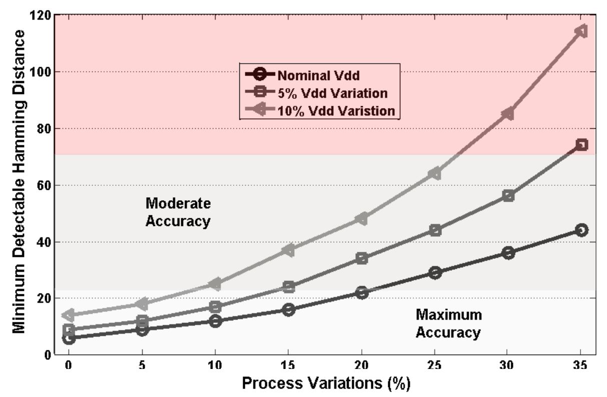


Figure 13. Impact of process and voltage variations for the minimum detectable Hamming distance in A-HAM.

for the smaller feature size or in conditions with low signal-to-noise ratio.

V. Conclusion

HD computing is about manipulating and comparing hypervectors stored in the associative memory. HD computing exhibits a robust behavior with faulty components thanks to its special brain-inspired properties: (pseudo)randomness with i.i.d. components, high-dimensionality, and holographic representations. This robustness allows us to design memory-centric architectures for efficient searches in the high-dimensional space by combining approximate techniques from three widely-used methodological design approaches. These HAM designs linearly scale with the hypervector dimensions and number of classes: increasing the dimensionality by a factor of 20 increases 1.9× the energy, and 1.7× the delay of A-HAM; increasing the classes by a factor of 16.6 increases 11.4× the energy, and 3.4× the delay of R-HAM. Compared to D-HAM, A-HAM reduced the area by 3× and significantly improve the energy-delay product by 746× for the maximum 97.8% classification accuracy, and 1347× for the moderate 94% accuracy. A-HAM is able to maintain such efficiencies in the presence of 10% voltage and 25% process variations.

VI. Acknowledgment

This work was supported by Systems on Nanoscale Information fabriCs (SONIC), one of the six SRC STARnet Centers, sponsored by MARCO, DARPA, and NSF grant #1527034.

References

[1] 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.

[2] P. Kanerva, "What we mean when we say "whats the dollar of mexico?": Prototypes and mapping in concept space," in *AAAI Fall Symposium: Quantum Informatics for Cognitive, Social, and Semantic Processes*, pp. 2–6, 2010.

[3] P. Kanerva, J. Kristofersson, and A. Holst, "Random indexing of text samples for latent semantic analysis," in *Proceedings of the 22nd annual conference of the cognitive science society*, vol. 1036, Citeseer, 2000.

[4] A. Joshi, J. Halseth, and P. Kanerva, "Language geometry using random indexing," *Quantum Interaction 2016 Conference Proceedings*, In press.

[5] A. Rahimi, P. Kanerva, and J. M. Rabaey, "A robust and energy efficient classifier using brain-inspired hyperdimensional computing," in *Low Power Electronics and Design (ISLPED), 2016 IEEE/ACM International Symposium on*, August 2016.

[6] F. R. Najafabadi, A. Rahimi, P. Kanerva, and J. M. Rabaey, "Hyperdimensional computing for text classification," *Design, Automation Test in Europe Conference Exhibition (DATE), University Booth*, 2016.

[7] A. Rahimi, P. K. 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 (ICRC 2016)*, October 2016.

[8] O. Räsänen and S. Kakouros, "Modeling dependencies in multiple parallel data streams with hyperdimensional computing," *IEEE Signal Processing Letters*, vol. 21, no. 7, pp. 899–903, 2014.

[9] O. Rasanen and J. Saarinen, "Sequence prediction with sparse distributed hyperdimensional coding applied to the analysis of mobile phone use patterns," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–12, 2015.

[10] P. Kanerva, "Encoding structure in boolean space," in *ICANN 98*, pp. 387–392, Springer, 1998.

[11] T. K. Landauer and S. T. Dumais, "A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge.," *Psychological review*, vol. 104, no. 2, p. 211, 1997.

[12] P. Kanerva, "Binary spatter-coding of ordered k-tuples," in *ICANN'96, Proceedings of the International Conference on Artificial Neural Networks* (, ed.), vol. 1112 of *Lecture Notes in Computer Science*, pp. 869–873, Springer, 1996.

[13] U. Quasthoff, M. Richter, and C. Biemann, "Corpus portal for search in monolingual corpora," in *Proceedings of the fifth international conference on language resources and evaluation*, vol. 17991802, p. 21, 2006.

[14] M. Imani, A. Rahimi, and T. S. Rosing, "Resistive configurable associative memory for approximate computing," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1327–1332, IEEE, 2016.

[15] S. Li, L. Liu, P. Gu, C. Xu, and Y. Xie, "Nvsim-cam: a circuit-level simulator for emerging nonvolatile memory based content-addressable memory," in *Proceedings of the 35th International Conference on Computer-Aided Design*, p. 2, ACM, 2016.

[16] J. Li, R. K. Montoye, M. Ishii, and L. Chang, "1 mb 0.41 μm^2 2t-2r cell nonvolatile tcam with two-bit encoding and clocked self-referenced sensing," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 4, pp. 896–907, 2014.

[17] I. Goiri, R. Bianchini, S. Nagarakatte, and T. D. Nguyen, "Approxhadoop: Bringing approximations to mapreduce frameworks," in *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '15, (New York, NY, USA), pp. 383–397, ACM, 2015.

[18] A. Goel and P. Gupta, "Small subset queries and bloom filters using ternary associative memories, with applications," *ACM SIGMETRICS Performance Evaluation Review*, vol. 38, no. 1, pp. 143–154, 2010.

[19] T. Kohonen, *Content-addressable memories*, vol. 1. Springer Science & Business Media, 2012.

[20] Y. Xie, "Emerging memory technologies," *Springer*, 2014.

[21] J. J. Yang, M. D. Pickett, X. Li, D. A. Ohlberg, D. R. Stewart, and R. S. Williams, "Memristive switching mechanism for metal/oxide/metal nanodevices," *Nature nanotechnology*, vol. 3, no. 7, pp. 429–433, 2008.

[22] Y. Chen, H. Lee, P. Chen, P. Gu, C. Chen, W. Lin, W. Liu, Y. Hsu, S. Sheu, P. Chiang, *et al.*, "Highly scalable hafnium oxide memory with improvements of resistive distribution and read disturb immunity," in *2009 IEEE International Electron Devices Meeting (IEDM)*, pp. 1–4, IEEE, 2009.

[23] K.-H. Kim, S. Gaba, D. Wheeler, J. M. Cruz-Albrecht, T. Hussain, N. Srinivasa, and W. Lu, "A functional hybrid memristor crossbar-array/cmos system for data storage and neuromorphic applications," *Nano letters*, vol. 12, no. 1, pp. 389–395, 2011.

[24] N. Pinckney, M. Fojtik, B. Giridhar, D. Sylvester, and D. Blaauw, "Shortstop: An on-chip fast supply boosting technique," in *2013 Symposium on VLSI Circuits*, pp. C290–C291, IEEE, 2013.

[25] J. J. Yang, D. B. Strukov, and D. R. Stewart, "Memristive devices for computing," *Nature nanotechnology*, vol. 8, no. 1, pp. 13–24, 2013.

[26] R. Dlugosz, A. Rydlewski, and T. Talaska, "Low power nonlinear min/max filters implemented in the cmos technology," in *Microelectronics Proceedings-MIEL 2014, 2014 29th International Conference on*, pp. 397–400, IEEE, 2014.

[27] P. Koehn, "Europarl: A parallel corpus for statistical machine translation," in *MT summit*, vol. 5, pp. 79–86, 2005.

[28] J. J. Yang, D. B. Strukov, and D. R. Stewart, "Memristive devices for computing," *Nature nanotechnology*, vol. 8, no. 1, pp. 13–24, 2013.

[29] C. Zhuo, D. Sylvester, and D. Blaauw, "Process variation and temperature-aware reliability management," in *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 580–585, European Design and Automation Association, 2010.