Cross-Layer Design of Vector-Symbolic Computing: Bridging Cognition and Brain-Inspired Hardware Acceleration

SHUTING DU*, Purdue University, West Lafayette, United States
MOHAMED IBRAHIM*, Georgia Institute of Technology, Atlanta, United States and The University of Texas at Dallas, Richardson, United States
ZISHEN WAN, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, United States
LUQI ZHENG, Purdue University, West Lafayette, United States
BOHENG ZHAO, Purdue University, West Lafayette, United States
ZHENKUN FAN, Georgia Institute of Technology, Atlanta, United States
CHE-KAI LIU, Georgia Institute of Technology, Atlanta, United States
TUSHAR KRISHNA, Georgia Institute of Technology, Atlanta, United States
ARIJIT RAYCHOWDHURY, Georgia Institute of Technology, Atlanta, United States
HAITONG LI, Purdue University, West Lafayette, United States

Vector Symbolic Architectures (VSAs), also known as hyperdimensional (HD) computing, are increasingly deployed in cognitive applications due to their simple and efficient operations. The widespread adoption has, in turn, spurred the development of a diverse set of hardware solutions that optimize VSA performance for embedded and edge AI systems. Despite these advances, there remains a lack of comprehensive, unified discussion on the co-design and co-evolution of VSA algorithms and hardware. This survey aims to bridge that gap by linking theoretical, software-level explorations with efficient hardware architectures and emerging technology fabrics for VSAs, providing co-design insights that are accessible to both algorithm and hardware communities. First, we introduce the principles of vector-symbolic computing, including its core mathematical operations and learning paradigms. Second, we provide an in-depth discussion on hardware technologies for VSAs, analyzing analog, mixed-signal, and digital circuit design styles. We compare hardware implementations of VSAs by carrying out detailed analysis of their performance characteristics and tradeoffs, from which we distill design guidelines that are applicable across arbitrary VSA formulations. Third, we discuss a methodology for cross-layer design of VSAs that

*Both authors contributed equally to this research.

This work was supported in part by the U.S. National Science Foundation under Award No. 2425498 with industry partners as specified in the Future of Semiconductors (FuSe2) program, and in part by the UPWARDS program. The work was also supported in part by CoCoSys, one of seven centers in JUMP 2.0, a Semiconductor Research Corporation (SRC) program sponsored by DARPA..
Authors' Contact Information: Shuting Du, Purdue University, West Lafayette, Indiana, United States; e-mail: du335@purdue.edu; Mohamed Ibrahim, Georgia Institute of Technology, Atlanta, Georgia, United States and The University of Texas at Dallas, Richardson, Texas, United States; e-mail: mibrahim@utdallas.edu; Zishen Wan, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, Georgia, United States; e-mail: zishenwan@gatech.edu; Luqi Zheng, Purdue University, West Lafayette, Indiana, United States; e-mail: zheng782@purdue.edu; Boheng Zhao, Purdue University, West Lafayette, Indiana, United States; e-mail: zhao969@purdue.edu; Zhenkun Fan, Georgia Institute of Technology, Atlanta, Georgia, United States; e-mail: zfan87@gatech.edu; Che-kai Liu, Georgia Institute of Technology, Atlanta, Georgia, United States; e-mail: che-kai@gatech.edu; Tushar Krishna, Georgia Institute of Technology, Atlanta, Georgia, United States; e-mail: tushar@ece.gatech.edu; Arijit Raychowdhury, Georgia Institute of Technology, Atlanta, Georgia, United States; e-mail: arijit.raychowdhury@ece.gatech.edu; Haitong Li, Purdue University, West Lafayette, Indiana, United States; e-mail: haitongli@purdue.edu.

ACM 1558-3465/2026/4-ART
https://doi.org/10.1145/3807784

identifies synergies across layers and explores key ingredients for hardware/software co-design of VSAs. Finally, as a concrete case study of this methodology, we present an in-memory computing hardware design for VSA-based hierarchical cognition, illustrating how the proposed co-design principles translate into efficient architectures. The paper concludes with a discussion of open research challenges and opportunities for future explorations.

CCS Concepts: • **Hardware → Emerging technologies**; **Application specific integrated circuits**; • **Computing methodologies → Bio-inspired approaches**.

Additional Key Words and Phrases: Vector symbolic architectures (VSA), hyperdimensional (HD) computing, accelerators, hardware/software co-design, in-memory computing

1 Introduction

Processing on streams of real-time data was one of the main motivators behind the early development of electronic circuits. Hence, it was no surprise that the enabling of digital integrated computing in the 1970s also led to specialized processors for signal processing. While initially focused on single-dimensional streams (e.g., audio [43]), advances in image and video processing rapidly led to processors that operated on two-dimensional and three-dimensional data, leading to vector and matrix processors (early 1980s) [110]. With digital neural networks gaining popularity, the need to operate on even higher-dimensional structures such as tensors has become necessary, leading to processors such as the Google TPU [86]. This trajectory suggests a continued trend towards computing in increasingly higher-dimensional spaces, driven by innovative applications across various domains, such as security, optimization, and cognition. Therefore, computing in high-dimensional (HD) spaces has received substantial interest not only from mathematicians but also from researchers in neuroscience and electronic-circuit design [38, 155, 185].

Among various approaches to computing in HD spaces, the field of *Vector Symbolic Architectures* (VSAs)—also referred to as Hyperdimensional Computing (HDC)—has emerged as a promising and effective approach, taking inspiration and guidance from the neural architecture of biological systems [89]. The conceptualization of this approach is founded on the idea that key aspects of human memory, perception, and cognition can be modeled by computing with HD distributed vector representations (or "hypervectors"), which capture the rich phenomena inherent in the collective activity of large populations of neurons in the brain [28, 178]. Sensory data, state variables, and high-level concepts are all mapped into vectors in an HD space, and an algebra over these vectors is then used to combine information to form new representations that can be used as the basis for further processing [103]. This compositional structure is central to the expressive power of VSAs and enables a broad range of cognitive and learning paradigms, spanning from classification, factorization, clustering, planning, reasoning, and problem-solving [42, 50, 104, 129, 149, 177], to regression [20, 63], structured visual perception [59, 101, 140] and distributed paradigms such as federated learning [136].

Furthermore, VSAs exhibit three inherent qualities that naturally facilitate efficient hardware implementation. The first is *robustness to errors*, a capability that stems from computing with distributed HD representations [136, 156]. The impact of this capability is that VSA hardware does not require ultra-reliable elements, enabling operation with minimal energy consumption even under low signal-to-noise conditions [10, 155]. The second quality is *element-wise independence*, which arises as computations are performed independently across all vector elements. This independence permits significant computational acceleration, with multiple operations executed simultaneously across the hardware fabric [20]. The third quality is *versatility*, which enables rematerialization of VSA primitives, potentially on-the-fly, to adapt to a wide range of computational tasks [168]. This flexibility is due to the ability of HD vectors to encode diverse types of data and relationships within a unified framework. Together, these qualities lay the foundation for growing research on VSA hardware development, supported by ongoing advancements in semiconductor technologies [4] and physical design flows [115].

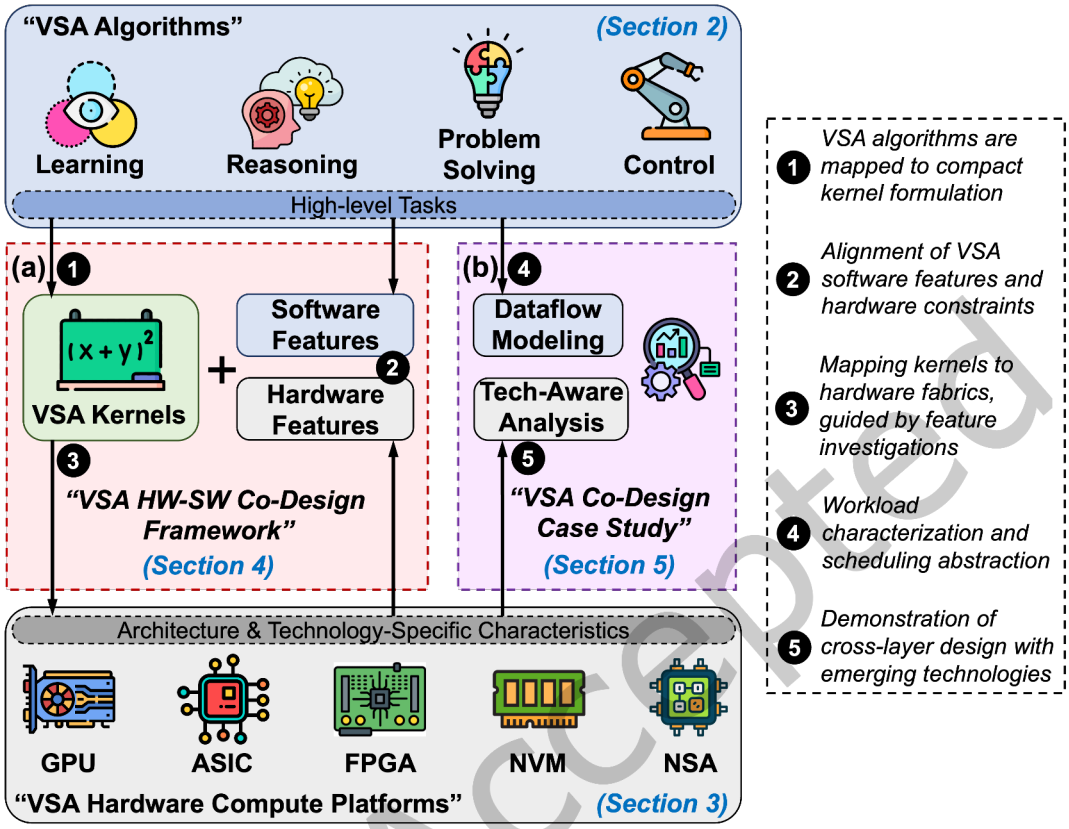

Fig. 1. An overview of VSA system design methodologies, including (a) an alternative hardware/software co-design framework, illustrating the mapping of high-level VSA algorithms into kernel formulations and hardware platforms through alignment of software features and hardware features; (b) a demonstration of a co-design case study through workload abstraction and technology-aware design.

Hence, research on the hardware development of VSAs has received significant attention in recent years, leveraging a broad diversity of hardware platforms. These include embedded CPUs/GPUs [91, 135, 173], FPGA platforms [80, 163, 164], custom application-specific integrated circuits (ASICs) [33, 37, 73], neuromorphic spiking arrays (NSA) [9, 147, 159], and emerging non-volatile memory (NVM) fabrics [93, 114]. Studies in this field have demonstrated highly efficient solutions, achieving energy savings and performance enhancements of several orders of magnitude compared to VSA implementations on general-purpose processors. Consequently, hardware advancements in VSAs have become crucial enablers for machine intelligence at the edge, providing advanced cognitive capabilities while meeting stringent energy and form-factor constraints [14, 133, 208].

Despite advances in today's hardware developments, they still face limitations in adaptability to VSA models and their ability to generalize across different tasks and precision levels; such characteristics are inherent in today's multi-modal VSA-centric flows [67]. Most current hardware lacks reconfiguration capabilities, thereby hindering systematic and scalable interplay between VSA algorithms and hardware. Typically, they employ a rigid,

unstructured, and ad-hoc design methodology. This approach also cannot meet the growing demand for heterogeneous cognitive models, such as neuro-symbolic systems [188, 189] and generative cognitive architectures [45]. Specifically, it shows limited capability in integrating VSA dataflows with other computational primitives like tensor units [207]. These limitations prevent the full exploitation of HD computations for next-generation VSA systems. Bridging this gap requires a structured and unified design approach, leading to a systematic and efficient interplay between hardware and software, as shown in Figure 1(a).

Realizing a unified design approach involves developing a flexible *hardware-software co-design* framework that fully exploits the inherent qualities of VSAs: robustness to noise, element-wise independence, and versatility. To achieve this goal, this framework should support multiple VSA models or representations, enabling adaptability to diverse computational tasks [167]. Structured and unified integration, as illustrated in Figure 1(b), requires the compact formulation of VSA kernels, consideration of algorithm-specific features and dataflows, and adherence to hardware constraints and performance limits. By streamlining interactions among these components, the framework allows VSA kernels to be effectively mapped to hardware fabrics. This process should be guided by thorough feature investigations to ensure optimal performance. Advanced design methodologies, such as modular and parameterizable hardware blocks, can facilitate this adaptability, and machine learning techniques can also be incorporated for real-time optimization and task-specific tuning to further enhance efficiency. Ultimately, this unified approach will create versatile and high-performance VSA systems capable of meeting the diverse demands of next-generation cognitive applications.

1.1 Contributions of the Paper

The purpose of this paper is twofold. First, it surveys key principles of VSAs, state-of-the-art hardware implementations, and algorithms, thereby complementing previous algorithm- or system-level reviews for VSAs [14, 103–105, 167]. Second, it provides a comprehensive and unified discourse on the convergence of hardware and algorithms, aiming to bridge the gap between theoretical explorations and the development of efficient VSA hardware architectures. We reason that an efficient realization of a broader set of VSA methods necessitates a holistic understanding of all system requirements [44], ranging from algorithm-level features and kernels down to hardware-level characteristics and design opportunities (Figure 1). The outcome of the proposed study is a holistic framework that lays out the principles of hardware-software co-design and integration in VSA systems. To the best of our knowledge, this is the *first* paper to address co-design principles and attributes for VSA systems, aiming to inspire the design of next-generation cognitive computing systems.

1.2 Organization of the Paper

The rest of this paper is organized as follows. We first review the principles of VSA computing and the different VSA methods (Section 2), laying the foundation for subsequent hardware-software co-design. This is followed by a comprehensive discussion of hardware implementations of VSAs, analyzing analog, mixed-signal, and digital circuit design styles to date (Section 3). We then establish a co-design framework that facilitates synergistic interactions between algorithm and hardware layers of abstraction (Section 4). We further illustrate an application of this framework that aims to create embodied cognition in autonomous machines (Section 5). Finally, we examine research challenges and opportunities for future explorations (Section 6).

2 Principles of Vector-Symbolic Computing

This section provides a comprehensive overview of vector-symbolic computing and its fundamental concepts. We delve into the definition and mathematical foundations of high-dimensional distributed representation (Section 2.1). We also present key VSA learning paradigms, with examples illustrating the need for VSA algorithm-hardware co-design (Section 2.2).

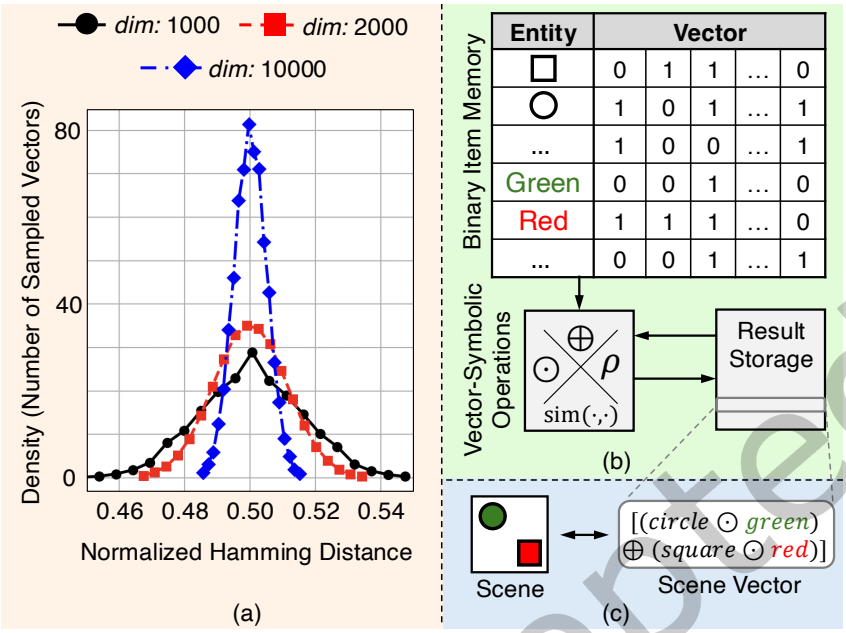

Fig. 2. Background of Vector-Symbolic Computing. (a) Orthogonality in high dimensions becomes more pronounced, i.e., density concentration becomes sharper around 0.5, as vector dimensionality increases. (b) Basic elements of vector-symbolic computations: item memory and algebraic operations. (c) A superposed, higher-order scene vector constructed via vector-symbolic operations.

2.1 Computing with High-Dimensional Vectors

Vector-symbolic computing is a method that encodes atomic attributes and patterns as high-dimensional distributed vector representations, commonly known as *hypervectors*, which typically comprise 1,000 dimensions or more. The high dimensionality allows for a rich encoding space, enabling the representation of complex information with a high degree of redundancy, which contributes to robustness against noise. Hypervectors are generated using various methods, such as random initialization, where each element in the vector is randomly assigned a value from a predefined set or a probability distribution [89]. This initialization mechanism results in *item hypervectors* that are nearly orthogonal to each other, as the likelihood of any two randomly generated hypervectors having a significant overlap is extremely low. The quasi-orthogonality feature becomes more pronounced as the length of the hypervectors is increased, as illustrated in Figure 2(a). A collection of randomly generated item vectors is typically referred to as the *item memory*.

Furthermore, vector-symbolic computing allows for constructing complex representations by combining hypervectors through basic mathematical operations, resulting in superposed vectors with the same dimensionality [46]. The basic mathematical operations used in such computations are vector similarity, binding, bundling, and permutation, as illustrated in Figure 2(b). For instance, a visual scene can be represented by integrating hypervectors for individual objects and their attributes, as shown in Figure 2(c). The resulting superposed vector retains the high-dimensional nature of the original hypervectors while embodying a richer and more nuanced representation of the combined elements. A brief description of vector-symbolic operations is presented below.
Vector Similarity: This operation, denoted by $\text{sim}(\cdot, \cdot)$, quantifies how closely related two hypervectors are in the high-dimensional space. This measure can be based on various metrics, depending on the specific architecture

and application. Common similarity measures include cosine similarity, Hamming distance, and dot product. Cosine similarity, computed as $\text{sim}(a, b) = \frac{a \cdot b}{\|a\| \|b\|}$, ranges from -1 to 1, indicating how aligned the vectors are. Alternatively, Hamming distance, which counts the number of differing bits between binary hypervectors, can be used for comparing vectors and is commonly adopted due to its amenability to low-cost hardware design [33, 133]. Similarity measures help determine the degree of resemblance between vectors and are essential for all vector-symbolic applications.

Binding: Binding, denoted by $\odot$, is an operation designed to model how the brain *connects* input information as key-value pairs. This operation operates on two input hypervectors and generates another hypervector that belongs to the same space (e.g., $x = a \odot b$). An example implementation of this operation is XOR, which is often used when binding binary hypervectors. The generated output hypervector x is quasi-orthogonal to its inputs (a and b). This property is attributed to the fact that the binding operation does not preserve intrinsic vector representations; rather, it can only preserve *similarity information* that governs vector relationships [154]: $\text{sim}(a \odot c, b \odot c) \approx \text{sim}(a, b)$.

On the other hand, the inverse operation for binding, also called the release or unbinding operation ($\ominus$), can retrieve information by disentangling the joint representation [102]. For instance, if we have a vector x obtained by binding a and b ($x = a \odot b$), applying the unbinding operation $x \ominus b$ will produce a vector that closely approximates a. Specifically, $\text{sim}(x \ominus b, a) \gg \text{sim}(x \ominus b, \epsilon)$, where ϵ is a random hypervector. This formula indicates that the unbinding operation effectively recovers the original vector a or a close approximation from the composite vector x, with high similarity.

Bundling: Bundling, denoted by $\oplus$, is an operation designed to model how the brain *memorizes* input information as a set of entities. This operation fuses information from all input hypervectors and generates a hypervector that represents their mean, that is, the output is maximally similar to all of the inputs: $\text{sim}(a \oplus b \oplus c, a) \gg \text{sim}(a \oplus b \oplus c, \epsilon)$, where ϵ is a random hypervector. The bundling of binary hypervectors is typically implemented using a simple addition operation and then applying a majority function over the sum vector to generate a binary vector. In some implementations, the summed vector is not binarized, preserving its higher-resolution structure [182]. The downside of such implementations is the significant increase in computational and memory requirements, as the representation must store and process real-valued or high-bit-depth vectors rather than binary ones.

Permutation: The permutation, denoted by $\rho(\cdot)$, is a unary operation that involves deterministically reordering the elements of a given hypervector, such as through cyclic rotation of the whole vector. This operation is essential for encoding sequences and ordered data, allowing hypervectors to represent the temporal or spatial arrangement of features and events [85, 94]. For example, let x_g, x_o, and x_d be the hypervector representations of the letters "g", "o", and "d", respectively. To encode the sequence of letters in the word "good", permutation is applied to each subsequent letter's hypervector and then combined. Specifically, the word "good" can be constructed as $x_g \odot \rho(x_o) \odot \rho^2(x_o) \odot \rho^3(x_d)$, where $\rho^k(\cdot)$ denotes the k-th permutation applied to the hypervector. Note that an arbitrary vector x and its permutation $\rho(x)$ are quasi-orthogonal to each other, meaning they can be easily distinguished from one another in the high-dimensional space.

2.2 VSA Algorithm Domains

Supervised Learning: Supervised learning with VSA has been proposed during the early stage of the development of its framework [88], which is also the foundation of subsequent VSA-inspired applications. Steps in VSA classification can be categorized into three steps: *encoding*, *training*, and *inference*. In the encoding phase, the feature $\vec{F}_{1\times n}$ is first projected to high-dimensional vector (hypervector) $\vec{H}_{1\times N}$ by multiplying the encoder $\vec{E}_{n\times N}$, with the element in the encoder sampled with random distribution. These encoders create an embedding $\phi : \mathbb{R}^n \rightarrow \mathbb{R}^N$ such that $< \phi(x), \phi(y) > \approx k(x, y)$, where k is a shift-variant kernel [183]. Most of the existing VSA

encoders originate from the fact that the Fourier transform of a shift-invariant kernel k is a probability measure, a result that originates from Bochner's Theorem [161].

After the encoding phase, training is performed to form the class hypervector $\vec{C}$. VSA is designed as a hardware-friendly framework that supports *single-pass* training. In the single-pass training phase, all the hypervectors belonging to label l are aggregated to form $\vec{C_l}$. To further increase the accuracy of the VSA framework, inspired by the perceptron learning algorithm, the *iterative* training scheme subtracts the wrongly predicted class hypervector $\vec{C_{l'}}$ and adds the correctly predicted class hypervector $\vec{C_l}$:

$$\begin{aligned} C_l &\leftarrow C_l + \eta(1-\delta) \\ C_{l'} &\leftarrow C_{l'} - \eta(1-\delta) \end{aligned} \tag{1}$$

where η is the learning rate for fine-tuning and δ is the similarity value (e.g., cosine similarity) of the class hypervector and the query. After the training, the VSA model is ready for inference, where the query feature is first processed with the same encoder for projecting into high-dimensional space, and then searched across all the class hypervectors with a pre-defined similarity metric as mentioned in Section 2.1. Finally, the highest similarity value is the predicted class for this query. Existing VSA hardware designs focus mostly on accelerating similarity search, as such an operation dominates the inference run-time. However, there are cases that the encoder dominates the inference run-time, such as time-series encoding [143], illustrating the need for *top-down perspectives* on the VSA co-design framework.

Unsupervised Learning: Unsupervised learning with VSA involves investigating the inherent statistical relationship between data in their high-dimensional representations. Imani et al. proposed HDCluster that adopts the K-means principle to generate clusters through iterations of comparison and refinement [76]. Central hypervectors are randomized evenly within HD space and iteratively updated by averaging all hypervectors annotating the same label. Learning phase ceases when the step of update becomes negligible. Computational efficiency could be improved through weighted updating of central hypervectors with the confidence level generated through cluster assignment phase [62]. Much resembling Deep Neural Networks, training and clustering (similarity search) could be performed with various types of operand: floating-point central hypervectors permitted more accurate updates ($C_l \leftarrow C_l + x$), while binarized hypervectors enabled more hardware-friendly similarity search using hamming distance ($sim(C_l^b, x)$). Applications in other realms include spectrum clustering [202] and traffic profiling [5], which demonstrate the efficiency and efficacy of VSA-based unsupervised learning.

On top of Holographic Reduced Representation (HRR) [151], hyperseed [148] facilitates unsupervised learning through optimizing a projection hypervector s between the encoded input hypervector space D and a mapped hypervector space P in complex domain ($D \ominus s \rightarrow P$). In each training iteration, the most divergent input hypervector is identified and assigned an anchor in the mapped hyperplane, representing a center for similar inputs. Such mapping informatics is recorded by incrementing projectory hypervector with the anchor-data binding pair $s \leftarrow d \odot p + s$. Since (un)binding preserves similarity between hypervectors, the mapping capability would be enhanced to project input hypervectors similar to d to the newly designated center. Through incorporating straying outliers, the mapped hyperspace is gradually distributed with clusters. Hyperseed achieved comparable performance against self-organizing map (SOM) [18] while significantly reducing the number of running passes by alleviating parallel search overhead with neuromorphic hardware [117].

3 VSA Hardware: Architecture and Technology

To enhance the efficiency of VSA-based applications with optimized HD compute and memory kernels, leveraging tailored architectures along with appropriate device technologies is crucial. This section reviews hardware implementations for VSAs from digital platforms (e.g., microprocessors, FPGAs, etc.) to custom ASIC designs exploiting in-memory or near-memory computing architectures and emerging device technologies. The mapping

between these hardware designs and the kernel functionalities is discussed, along with the corresponding design tradeoffs.

3.1 Digital Hardware Platforms

With high overheads and a lack of VSA-specific parallelism, floating-point arithmetics in CPUs and GPUs do not match efficiently with the high-dimensional vector algebra for VSA applications, implying the need for HD-vector-centric processor architectures [32]. Taking advantage of the fine-grained parallelism and pipelining capabilities of FPGAs [20, 119, 122], prior work has demonstrated FPGA acceleration for diverse VSA workloads, including online hyperdimensional regression [20]. Salamat et al. [163] introduced an automated FPGA framework for HD computing acceleration. The design of Menon et al. [131] relied on a novel bit-serial word-parallel approach to enhance the spatial encoder in VSA bundling. Montagna et al. [135] proposed HD computing acceleration on the Parallel Ultra-Low Power (PULP) Platform, a software programmable cluster architecture, with a parallel processing chain method that separately parallelizes each component in the chain and distributes the computational tasks across multiple processing cores for performance optimization.

In addition to embedded system prototypes, custom ASICs have also been developed for hardware acceleration of HD computing. Datta et al. [32] proposed a programmable HD processor design utilizing a Hyper-dimensional Logic Unit (HLU) systolic array architecture for the HD encoder. This leads to a reconfigurable architecture capable of forming multiple HLU layers, and exhibits excellent energy efficiency across various supervised classification tasks, including language recognition and human face detection. In [33], the HDBinaryCore, a digital 28 nm CMOS chip, is the first silicon prototype as a programmable HD processor for biosignal processing. It adopts the architecture proposed in [32], allowing programmability through the specification of interconnection and operation for each HLU Layer.

3.2 In-Memory and Near-Memory Computing Architectures

The high-dimensional nature of hypervectors in VSA operations can incur intensive and frequent data movement between processing units and memory components, creating a bottleneck for VSA applications within traditional von Neumann architectures. Meanwhile, the similarity search among high-dimensional vectors as a key building block in VSAs inspires memory-centric designs. Benefiting from highly efficient in-/near-memory operations and significant computational parallelism, emerging in-memory computing (IMC) or near-memory computing (NMC) architectures present a promising path to address the VSA-specific memory walls. These architectures, specifically designed for VSAs, fall into two primary categories: digital IMC/NMC [3, 23, 36, 40, 57, 193], and analog IMC/NMC designs [71, 93, 136, 137].

At the system level, representative near-memory computing approaches prioritize integrating computation into off-chip memory hierarchy to mitigate data movement bottleneck. By embedding programmable or reconfigurable logic close to DRAM, architectures such as Tesseract [3], Active Memory Cube [23], and NDA [40] provide high-bandwidth substrates for parallel vector processing. Specifically, Tesseract and Active Memory Cube leverage 3D-stacked DRAM to integrate programmable cores within a logic base layer, while NDA stacks reconfigurable logic atop commodity DRAM to maintain standard interface compatibility. These near-DRAM processing approaches can be further adapted for VSA workloads. For example, the massive internal bandwidth and bank-level parallelism can enable large-scale execution of bitwise binding and bundling to be executed across the wide, unsegmented hypervector dimensions.

Complementing these system-level approaches, on-chip digital IMC architectures typically employ binary in-memory or near-memory logic circuits to accelerate VSA operations. Thus, the digital nature of these computations can eliminate the accuracy loss induced by analog-to-digital conversions. Wang et al. [193] proposed an HD outlier detection (ODHD) accelerator combining SRAM IMC processing element (PE) array with assistant NMC,

which enables high throughput VSA operations (binding, bundling, permutation) in the ODHD algorithm due to parallel computation across the different subarrays. Specifically, the digital SRAM PEs are co-designed with the ODHD algorithm for efficient mapping of encoding, training, threshold calculation, and fine-tuning steps. Gupta et al. [57] presented an HD computing accelerator with a resistive RAM (RRAM) IMC crossbar capable of single-cycle NOR, NOT, NAND, and OR logical functions in memory. This RRAM array efficiently stores pre-trained identity hypervectors and quantized hypervectors within a single memory partition, performing in-memory XOR operations in series. This process maps a feature vector with n elements to high-dimensional space, enabling acceleration in the HD encoding module. In [36], the HDnn-PIM architecture is introduced as a high-dimensional neural network (HDnn) IMC solution. Specifically designed for complex image classifications, it incorporates a few initial stages of convolution-based feature extraction to enhance the effectiveness of HD learning on intricate data. The HDnn-PIM employs a tiled architecture with RRAM crossbars, featuring supertiles that facilitate input reuse and fusion of tiles for handling large input sizes.

Architectures for analog in/near-memory computing leverage physical principles such as Ohm's law or charge sharing/re-distribution. These principles are utilized for two fundamental operations: multiplication and addition. Karunaratne et al. [93] proposed an in-memory HD computing system implemented on two phase-change memory (PCM) crossbar engines alongside peripheral digital CMOS circuits. The first PCM crossbar engine is dedicated to accelerating HD encoding, employing in-memory read logic operations for hypervector binding and near-memory CMOS logic for hypervector bundling. The second PCM crossbar engine is employed for associative memory (AM) search, utilizing in-memory dot-product acceleration to calculate the inverse Hamming distance. An experiment involving 760,000 PCM devices for analog IMC also demonstrates comparable accuracy to software implementations. In [136], an analog RRAM IMC architecture is proposed, capable of supporting dot-product for matrix multiplication in encoding and similarity search during inference. Additionally, it supports addition and subtraction during training and retraining to enhance model robustness. As an example of algorithm-hardware co-designs, Morris et al. [137] introduced a stochastic HD computing system, leveraging the complex task-solving capabilities of VSA models.

A growing trend is that emerging non-volatile memories (NVMs) are being widely explored and adopted in IMC designs for VSA systems [214], including RRAM [29, 36, 57, 136, 137], PCM [93], and 3D NAND Flash [71, 211]. The on-chip NVMs provide fast, energy-efficient access and compute, whereas the off-chip 3D Flash offers even higher storage capacity. Non-volatility benefits edge VSA systems for lifelong learning and inference [113]. NVMs can serve as associative memories in encoding and similarity search operations, which only require infrequent memory writes. The biggest challenge with NVMs in conventional IMC designs targeting DNNs is that the device non-idealities may heavily impact model accuracy. In contrast, given the high-dimensional and holographic nature of VSAs, error bits in hypervectors are not contagious, leading to a computational system inherently resistant to defects and disturbances. In [114], multi-layer 3D vertical RRAM (VRRAM) integrated with FinFETs forms energy-area-efficient MAP kernels, exploiting vertical connectivity of heterogeneous device technologies. In [199], a highly-efficient, end-to-end 3D HD nanosystem is built, leveraging the integration of RRAMs and carbon nanotube field-effect transistors (CNFETs). The inherent variations in RRAMs and CNFETs are collectively harnessed while demonstrating robustness to errors inherent in the underlying hardware.

3.3 System Integration and Scaling

Scaling up memory-centric designs in a cost-effective fashion may further make VSA systems more efficient by providing higher total capacity and exposing additional parallelism. Technology scaling and integration of emerging memories will continue to unlock additional system optimization opportunities. Rapid industry progress has been observed, for example, the back-end-of-line (BEOL) integration of state-of-the-art RRAM technologies in foundry FinFET processes [53, 112]. Blurring the on-chip vs. off-chip boundary and leveraging the system

integration techniques will help amortize the costs of mapping large-scale VSA workloads or processing large databases. Advanced packaging and heterogeneous integration will become increasingly important in enabling flexible designs where VSA hardware fabrics may leverage diverse semiconductor technologies to improve system-level connectivity, energy efficiency, and footprints. Traditional 2.5D and 3D integration involves stacking and bonding separate chips, utilizing through-silicon vias (TSVs) and micro-bumps for vertical connections. Typically, TSVs have a pitch ranging from 20 to 40 μm, with advanced designs achieving pitches as fine as 10 μm, and the pitch for micro-bumps is approximately 40 μm [172]. Further scaling down the micro-bump pitch can be challenging for higher interconnection bandwidth. However, hybrid bonding technology [1] is being actively developed and can be integrated with backend wafer/die packaging, enabling higher interconnection density, energy efficiency, and improved signal/power integrity.

Heterogeneous 3D integration can provide unique functionalities and performance enhancement for VSA systems. For instance, 2.5D and 3D designs with heterogeneous chiplets embedded with computational memories could offer significant improvements in energy efficiency and energy-delay product (EDP) compared to conventional 2D architectures [34]. Furthermore, the on-sensor computation [116] is facilitated by the triple-wafer-stacking technology, where a logic wafer is integrated with dual-layer stacked digital pixel sensors through micro TSVs, and sensor layers are face-to-back hybrid bonded. The integration of sensors and logic dies can further facilitate VSA-based robotic applications, interweaving perception, reasoning, and control modules. In addition, for large-scale VSA architectures that need high data bandwidth and memory capacities, 3D DRAM (e.g., high bandwidth memory (HBM) utilizing DRAM die stacking) may play a crucial role in co-designing future VSA computing systems.

As a complement to heterogeneous integration, monolithic 3D integration interweaves layers of sensors, memories, and logic through monolithic inter-layer vias (MIVs). The diameters of MIVs are much smaller compared to TSVs [54], and the pitch of MIV can reach 100 nm in 14 nm process [165]. Therefore, monolithic 3D integration enables close integration between logic and memory components and facilitates much denser vertical connections than TSVs [120, 121]. In [199], CNFETs and RRAM are deployed for logic circuits and memories, respectively, and are integrated using the monolithic 3D technology, which is feasible due to the low-temperature fabrication of RRAM and CNFET.

VSA system integration may further benefit from alternative chip-to-chip communication. For instance, wireless in-package communication technology was used to interconnect a large number of physically distributed IMC cores, allowing for joint broadcast distribution and computation, and extensive parallelization of the architecture to improve system performance [55, 56].

4 VSA Hardware/Software Co-Design

The increasing complexity and multi-modality of cognitive tasks are driving VSA system design towards more systematic methodologies. To explore hardware-efficient computing, neural architecture search (NAS) is developed in the broader machine learning community to automate the exploration of model architectures and hyperparameters, reducing reliance on manual tuning and expert heuristics [118, 157, 215, 216]. However, traditional NAS frameworks focus on navigating the neural-architecture search space and often under-utilize the substantial hardware design freedom available in the cloud-to-edge continuum. To bridge the gap between high-level model design with hardware constraints, hardware-aware NAS [11, 12, 31, 181, 198] incorporates physical execution metrics, such as latency, energy, and memory footprint, directly into the search objective. Most hardware-aware NAS workflows still assume a fixed hardware template, which limits systematic exploration of the joint hardware-software optimization. For instance, HwAwHDC [204] evaluates the candidate models by rewarding reduced hypervector dimensions and alternative datatypes, but it does not co-optimize dataflows and memory hierarchies, which can be central to VSA system performance and efficiency.

More recently, hardware-software co-exploration has further advanced the Pareto frontiers of system efficiency by simultaneously traversing the model/architecture search space and the hardware design space [22, 83, 84, 184, 205, 206]. A representative two-level co-exploration framework [84] accelerates the search process by first pruning hardware-inefficient candidates during a fast exploration stage, then applying reinforcement-learning-based training in a slower stage to maximize accuracy under hardware-efficiency objectives. CODEBench [184] provides a benchmarking sub-framework that leverages Bayesian optimization with second-order gradients and heteroscedastic surrogate models to efficiently navigate neural architectures, while expanding evaluation fidelity using inlined cycle-accurate accelerator simulations. ASICNAS [206] reduces the ASIC design space through template sets derived from proven dataflows, enabling the simultaneous identification of multiple DNN architectures and their heterogeneous ASIC sub-accelerators to satisfy design specifications. CHaNAS [22] introduces a block-based pre-scheduling methodology to shrink the co-design search space and automatically generates both the network architecture and the scheduling policy. NANDS [205] co-explores the NAS and network-on-chip (NoC) design through a multi-phase manager that guides the system toward optimal accuracy-throughput tradeoffs. NACIM [83] couples neural architecture search with computing-in-memory designs, jointly optimizing across device types, circuit topologies, and architectures while explicitly accounting for device variations to improve robustness.

Despite these advances, these automated design approaches cannot be directly applied to VSA systems because VSA computation is organized around symbolic/algebraic primitives operating on high-dimensional representations. This shift introduces distinct requirements for data movement, memory mapping, and the realization of VSA kernels on concrete hardware primitives. At present, specialized co-exploration frameworks that explicitly capture these VSA structural dynamics remain limited. Consequently, an effective VSA system design methodology must first establish a principled mapping between VSA kernel dynamics and hardware primitives.

To cope with the complexity of next-generation VSA systems, we reason that an effective design process should comprise two essential steps: (1) *kernel formulation*, which aims to formally[1] describe the dynamics of the target function and map this formulation into abstract data and control flows; (2) *feature-aware hardware realization*, which takes the abstract data and control representations along with cross-layer features and map them into suitable hardware components (Figure 1(a)). It is worth noting that the first step provides a technology-agnostic procedure, whereas the second step focuses on incorporating the specifics of the problem domain. Below, we discuss these two steps in detail and present a unified co-design framework, bringing these two steps into a merged design space for VSA-based cognitive systems.

4.1 VSA Kernels

The first step in the design process is to determine the VSA kernel functions that need to be realized in hardware. A major advantage of kernel formulation is its ability to efficiently capture multiple sequences of high-dimensional operations and also present proper control primitives for selecting configurations. The goal here is to represent the various dynamics and interactions between hardware and software while abstracting away low-level details of the hardware [73]. This is typically achieved by formulating the kernel computation as $F(I, P, C)$, where $F(\cdot)$ represents a collective array of all target kernel functions $\{f_1, \ldots, f_m\}$, which together cover the whole domain of high-dimensional operation sequences. The argument I is an array of randomly generated item hypervectors, and P is an array of composed hypervectors. The argument C represents a set of conditional constructs that define the subdomains associated with the computational elements f_i. As such, when mapping the formulation to hardware, the set C conceptualizes multiplexers and other control primitives employed to select HD data paths

[1]The formulation may involve mathematical descriptions, flow charts, or a representation using data structures.

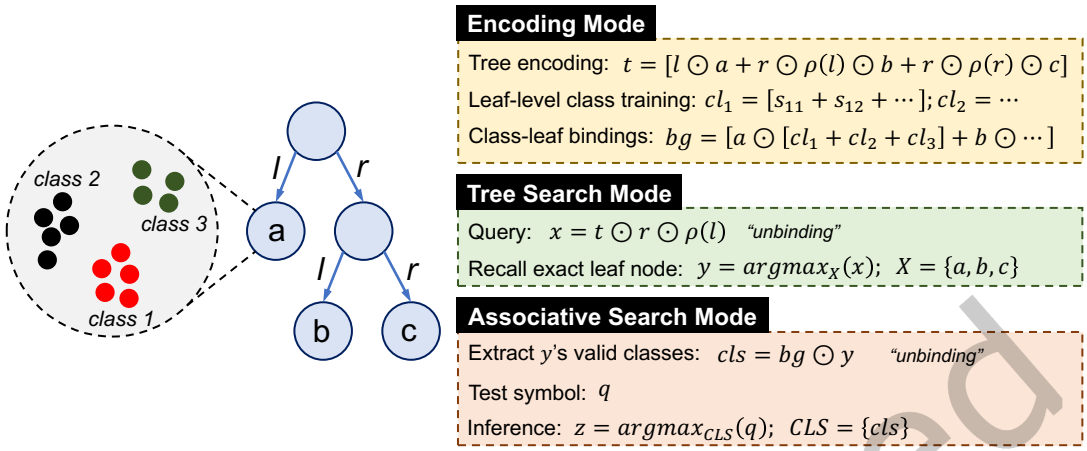

Fig. 3. An example of an arbitrary VSA classification application with its three operating modes: encoding, tree search, and associative search. The encoding mode constructs a hierarchical tree t by binding leaf node identities (a, b, c, etc.) with unique path descriptors (l, r), while mapping features (s_{11}, s_{12}...) to class hypervectors (cl_1, cl_2...) for class training to form the binding guide bg. In tree search mode, a query path hypervector is used to unbind and recall the specific leaf identity y from the global tree t. The associative search mode subsequently employs the bg to extract valid class hypervectors cls corresponding to that leaf identity. These are then compared against a test symbol q for final classification.

according to the execution mode. This methodology offers a high degree of flexibility and is commonly used with domain-specific processors [176, 180].

Example: To illustrate kernel formulation, we consider an arbitrary VSA classification algorithm that performs pattern recognition, which is distributed across a tree structure. Particularly, as illustrated in Figure 3, the algorithm has three modes: *encoding*, *tree search*, and *associative search*. The encoding mode aims to construct a tree of classes, where each leaf node represents a local classification setting. As such, each leaf node receives symbols and encodes them into a unique set of classes. In the tree search mode, the goal is to determine the identity of a leaf node given the path leading to it from the root. The associative search operates at the leaf level and aims to produce a class label given a test symbol.

Given the characteristics described above, we realize that multiple kernel formulations $F(I, P, C)$ can be inferred for such an application. For example, Figure 4(a) shows a spatial kernel formulation where computations for all the individual modes are distributed spatially. This approach does not require any control conditions in order to set up the data paths for execution, i.e., the set C is empty. In other words, this approach of kernel formulation results in a highly parallel hardware design with very limited software reconfigurability.

On the other hand, Figure 4(b) shows a temporal kernel formulation where computations for all the individual modes are centralized, resulting in data paths that are shared among all execution modes. This approach requires a comprehensive set of control constructs C to be incorporated into the data paths. As such, this approach results in an area-efficient design with a high degree of software reconfigurability. However, the resource sharing adopted by this approach may result in a significant increase in latency compared to the approach in Figure 4(a).

It is evident that the choice of a kernel formulation can significantly impact the overall performance and efficiency of the application. The selection of a specific kernel formulation determines how computational resources are utilized, influencing aspects such as parallelism, data flow, and hardware-software interaction. Moreover, the adaptability and reconfigurability of the chosen kernel formulation play a crucial role in addressing dynamic requirements and optimizing resource utilization. A well-chosen kernel formulation aligns with the

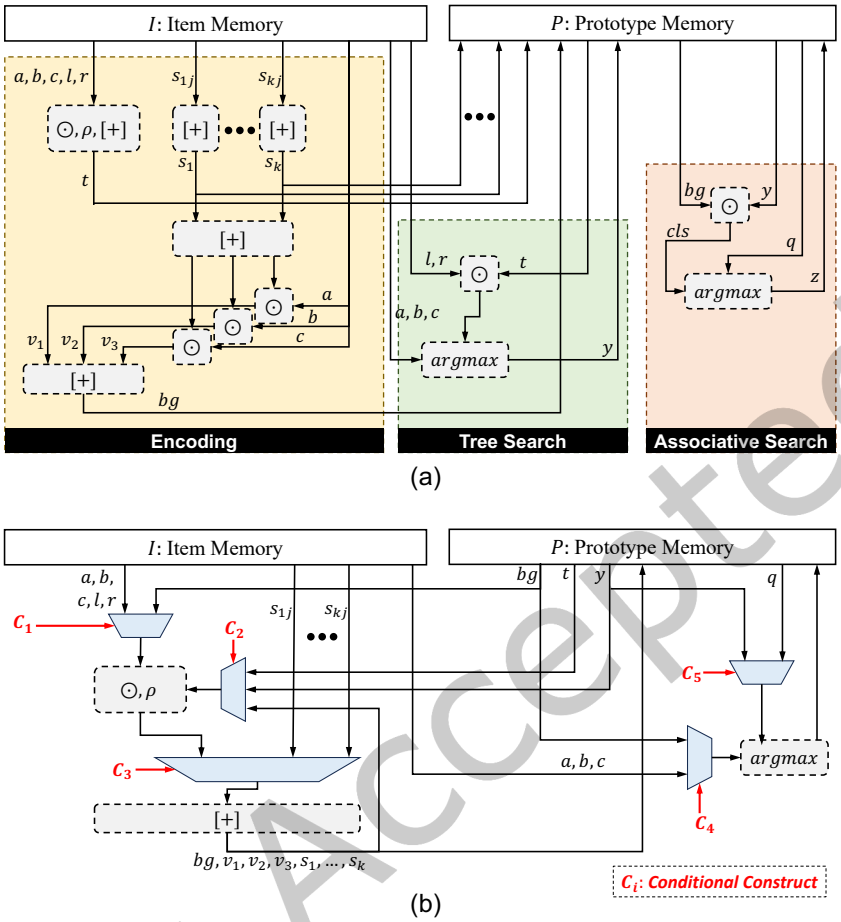

Fig. 4. Visualization of kernel formulations for the example in Figure 3. (a) Spatial kernel formulation, where computations for all the individual modes are distributed spatially across datapaths. (b) Temporal kernel formulation, where a shared datapath is time-multiplexed across execution modes and requires additional control constructs.

specific characteristics and demands of the application, striking a balance between hardware efficiency and flexibility. Similar tradeoffs can also be found when choosing hardware technologies to implement a kernel formulation—some details on this matter will be presented later in Section 4.2.

4.2 Cross-Layer Features and Hardware Realization

Besides kernel formulation, the design process also requires an exploration of essential *domain-specific properties*—a set of key attributes that are primarily inspired by the dynamics of VSA methods. Exploring such properties plays a key role in bridging the gap between theoretical aspects of VSA methods and their counterparts in hardware features. Below, we elaborate on each of the domain-specific properties, with the associated cross-layer features summarized in Table 1.

(1) *Leveraging Stochastic Characteristics:* VSA methods are not inherently classified as stochastic computing architectures; nevertheless, they exhibit stochastic-like characteristics that, when harnessed, can lead to enhanced computational efficiency [153, 187]. An illustrative instance is the utilization of controlled noise-injection techniques, which prove beneficial in breaking limit cycles and enhancing the efficiency of recurrent VSA algorithms, such as resonator networks [42]. Such stochastic techniques are only achievable through synergistic hardware/software integration. In other words, there is a need to align the probabilistic essence of VSAs with hardware primitives that can inject controlled noise, e.g., via ring oscillators, or are inherently stochastic [111].

(2) *Controlling Performance-Accuracy Tradeoffs:* VSAs offer a set of intrinsic control knobs that can finely adjust the system's performance and accuracy behaviors in alignment with application requirements. Key control knobs include vector length, sparsity level, binding complexity, and thresholding [106]. Taking vector length as an illustrative example, a shorter vector length enables faster and more energy-efficient processing, rendering it well-suited for real-time applications or scenarios with stringent latency and energy constraints. Conversely, a longer vector leads to enhanced superposition capacity, thus potentially improving the accuracy [103]. However, this advantage comes with increased computational complexity and hardware area. Examples of such techniques are spatial tiling [32] and temporal vector folding [132]. As such, identifying insignificant dimensionality in hyper-vectors has been proposed recently in [217]. With the observation that in VSAs, not all dimensions would have similar impact on the learning task. There are some dimensions with minimal impact during the learning phase, underscoring the possibility of maintaining both hardware performance and software accuracy. Other examples that could maintain both hardware performance and software accuracy include novel quantization scheme [6] and sparsity-aware hardware design [21].

(3) *Adaptability to Multiple Cognitive Tasks:* As detailed in Section 2, the mathematical properties inherent in VSAs impart them with the versatility to tackle a wide array of tasks, encompassing optimization, factoring, classification, as well as learning and reasoning. However, seamless integration of various cognitive tasks demands an interplay between flexible software implementations and adaptable hardware dataflows. On the software front, achieving flexibility involves constructing hierarchical cognitive architectures within VSAs, facilitating the organization and processing of encoded information across different levels of abstraction [146]. Meanwhile, from a hardware perspective, the efficient sharing of memory-search components and high-dimensional arithmetic units becomes crucial, tailored to the specific demands of diverse cognitive tasks [73]. A comprehensive co-design framework for VSAs should consider the joint-tuning of both software flexibility and hardware reconfigurability to exploit the diversity of VSA methods.

(4) *Interoperability with Real-World Interfaces:* The integration of cognitive systems with real-world environments relies heavily on their ability to navigate diverse multi-modal sensory-motor interfaces [169]. The inherent diversity of these interfaces leads to heterogeneous representations, often entangled within a low-dimensional space. The adoption of VSAs in this context allows these representations to be effectively mapped to high-dimensional spaces, thereby achieving linear separability. This transformation, often referred to as *encoding*, leads to an optimized realization of sensory-motor applications, especially since only a few samples need to be processed through VSA operations [134, 166]; refer to Section 2.2. Yet achieving effective VSA encoding necessitates seamless coordination between hardware knobs—specifically, sampling frequency and dataflow capacity—and other software knobs, such as the encoding complexity and supported quantization levels [132, 183].

(5) *Compatibility with Neural Cognitive Systems:* VSAs exhibit a compositional nature through their ability to represent complex symbols by combining simpler symbols in a structured and hierarchical manner. This feature renders VSAs highly compatible with neural network representations, giving rise to a class of

Table 1. Representative cross-layer features that address VSA's domain-specific properties.

Properties	Software Features	Hardware Features	Use Cases
[1]	Random bit-flipping (noise)	Inherent variations in NVMs	Resonator networks [111]
	Stochastic arithmetic	Logic-in-memory	Feature extraction [153]
[2]	Variable vector lengths	Temporal vector folding	Vector generation [132]
	Dropping insignificant dimensions	Noise-energy co-optimization	IoT system [217]
	Controlled sparsity level	Density-thinning logic	Sparse event recognition [66]
	Scheduler for sparse graph	Pipeline-style decoder IP	Graph reasoning [21]
	Non-linear quantization	Learning realistic device noise	Genome sequence matching [6]
[3]	Hierarchical models	Programmable dataflows	Sensory-motor learning [73]
[4]	Variable quantization levels	Data-level parallelism	Online perception [134]
	Stream data processing	On-chip encoding	Seizures detection [49]
[5]	Neuro-vector integration	Hybrid vector/scalar dataflows	Visual reasoning [67]
[6]	Noise-resilience local training	Adaptive bit-width precision	Federated learning [136]

models referred to as *neuro-vector-symbolic architectures* (NVSAs) [67, 209]. These integrated models enable sophisticated and generalized probabilistic reasoning, thus surpassing the limitations of neural networks that operate in isolation [95, 128]. However, the true benefit of realizing NVSA models emerges only when their inherent compositionality is mirrored in the hardware realizations. Specifically, it is necessary to ensure that computational units and memories originally designed for accelerating neural networks can also be shared, or at the very least accessed, by dataflows that perform vector-symbolic operations. This synergy between software and hardware is key for advancing NVSAs [207].

(6) *Architecting for Federated Learning Paradigms:* The algebraic properties of VSA align with key bottlenecks in federated learning, particularly in addressing environmental noise, communication overhead, and privacy protection [14]. VSA's holographic nature ensures that information is distributed across all dimensions. This provides the fault tolerance necessary for robust model updates in wireless transmission scenarios, where even significant bit-flips caused by multi-path fading or interference have minimal impact on overall similarity, preserving the integrity of the global model construction [13]. Moreover, binary/bipolar representations simplify logic and reduce effective bit width, lowering both the energy cost of frequent local iterations on edge devices and the communication bandwidth required for exchanging model parameters [70, 212]. During global aggregation, VSA's bundling fuses updates from multiple clients and provides a native obfuscation mechanism; however, bundling alone does not guarantee privacy due to the reversible nature of VSA primitives. To mitigate feature-extraction and model-inversion attacks, privacy can be further strengthened via encryption, differential privacy protection, local sparse coding, noise injection, and model quantization [61, 75, 97, 99]. From a hardware perspective, prior efforts such as HyDREA [136] primarily target noise-resilient local training by leveraging PIM-based VSA acceleration. A more comprehensive co-design framework remains needed to optimize communication-efficient bundling and enable hardware-native, privacy-preserving aggregation for federated and distributed learning.

It is worth noting that our goal here is not to identify a comprehensive set of hardware features; instead, we aim to outline guidelines or a methodology for deriving such features. As such, we seek to provide a representative set that can be incorporated into the co-design process based on the complexity or performance of the target context,

as illustrated in Figure 5. We map possible implementations in a two-dimensional space, categorizing them by their latency/throughput and energy consumption requirements across various applications. Each data point, derived from related research as referenced in Table 2, is color-coded to match its corresponding application area.

(1) *Classification Applications:* Classification stands out as one of the most compelling applications of VSA computing, with various applications demonstrating a critical sensitivity to factors such as latency and energy consumption. Examples of VSA classification benchmarks include physical activity classification, speech recognition, cardiotocogram classification, and applications targeted towards human-centric Internet of Things (IoT) systems [32]. The encoder and associative search modules are pivotal in the classification process. Recent research efforts have embraced in-memory computing, employing both digital [32, 193] and analog/mixed-signal [137, 199] approaches, to enhance the efficiency of the encoding process and associative memory searches. These innovative designs, however, may encounter limitations in reconfigurability. For instance, some encoders are tailored to specific data types [36] or encoding patterns [32], while others exploit emerging device technologies such as PCM [93] and 3D integration of CNFET and RRAM [199]. Such advancements often require the development of application-specific custom solutions to fully realize their potential.
(2) *Genome Sequencing Applications:* The large-scale databases required for genome sequencing applications present significant throughput and capacity challenges for hardware. To tackle these issues, researchers have employed high-density 3D NAND Flash to efficiently handle the large datasets of class hypervectors [71]. In addition, the integration of reference buffers with parallel computing units has proven effective in alleviating memory access bottlenecks, reducing reliance on large on-chip caches, and enhancing overall system performance [100].
(3) *Graph and Reasoning Applications:* Graphs naturally represent relationships and interactions across domains such as social networks, biological systems, and knowledge graphs. Reasoning over these graphs encompasses tasks like graph reconstruction, node classification, and graph matching. Addressing these tasks has led to the development of domain-specific encoder and decoder hardware [21], with a particular emphasis on exploiting data sparsity to improve computational efficiency [21, 92]. Additionally, emerging hardware technologies like FeFET and PCM are being explored [171]. The inherent device noises [7], stochastic properties [111], and switching dynamics [152] are being studied and utilized to tailor hardware designs specifically for graph-based reasoning tasks, offering new avenues for application-specific optimizations.
(4) *Real-time Learning Applications:* The VSA computing paradigm is being extensively explored for its potential to deliver highly efficient learning capabilities, especially in real-time application scenarios. This effort has driven the development of hardware-friendly encoder [19], and non-MAC-based feature extractor with VSA classifiers, aimed at supporting low-latency retraining at the network edge [87]. To further reduce latency, some researchers have proposed completely bypassing the encoding computation by leveraging memory lookup techniques, offering a promising approach toward efficient real-time learning systems [80].
(5) *General Applications:* General hardware architectures have been proposed for a wide range of VSA applications, emphasizing reconfigurability and task-agnostic designs. These architectures span GPU/CPU [131, 135], FPGA [163], ASIC [98], and IMC [47, 96], providing flexible hardware design frameworks for implementing VSA algorithms.

Summarized in Figure 5, generic architectures are positioned in the lower left corner due to their versatility across various application domains. The trend indicates that CPU/GPU-only, FPGA, and some IMC solutions offer moderate performance, striking a balance between latency and energy consumption. In contrast, a hybrid approach combining configurable ASICs/digital units with IMC can deliver superior performance despite its complexity. Such designs can be tailored for energy efficiency and latency reduction, creating a spectrum for exploring efficiency versus reconfigurability based on application needs. Additionally, the adoption of 2.5D

Table 2. Application-Specific Hardware Features Summarized from VSA Implementations. Reference Indices correspond to the specific implementations mapped in Figure 5.

Application	Reference Index	Hardware Features
Classification	[1] [32]	Flexible digital architecture, ROM + flip flops, DPU
	[2] [199]	3D integration of CNFET and RRAM, RRAM gradual reset
	[3] [36]	RRAM crossbar, Tile-based architecture
	[4] [93]	PCM-based, Partition method for AM search
	[5] [137]	Analog IMC, ADC-less
	[6] [193]	Digital IMC, SRAM tile-based architecture,
	[7] [57]	Single-cycle IMC logic
	[8] [77]	XOR array
	[9] [136]	Adaptive bit-width change
	[10] [79]	Two-stage pipeline, MAC hardware for clustered class vector
	[11] [41]	Electro-photonic accelerator for VSA training and inference
Genome Sequencing	[12] [71]	High density 3D NAND flash memory
	[13] [100]	Multi-platform compatibility, Reference buffer
Graph and Reasoning	[14] [152]	NOR-based, Switching characteristic of NVM
	[15] [21]	Pipelined matrix multiplication, Domain-specific encoder and decoder
	[16] [7]	FeFET-based, Learning realistic device noise
	[17] [92]	FeFET-based, Address sparseness and irregularity
	[18] [111]	PCM stochasticity, In-memory MVM
Real-time Learning	[19] [87]	Shift-ACCumulate instead of MAC, Neural network accelerator
	[20] [80]	Map encoder module to simple memory lookup
	[21] [19]	Hardware-friendly encoder IP, HV chunk fragmentation
Generic Architecture	[22] [135]	Processors cluster architecture
	[23] [47]	Mixed-signal processing unit
	[24] [96]	FeFET-based content-addressable memory
	[25] [98]	Low-power, energy-efficient, low-latency ASIC platform
	[26] [131]	Vector lanes in vector accelerator
	[27] [163]	DSP mapping for PE, Prefetch buffer for BRAM delay hiding

and 3D chiplet integration technologies facilitates the seamless integration of diverse applications or modules, optimizing dataflow and enhancing computational efficiency.

To understand the performance, energy consumption, and resource requirement across diverse VSA application domains, we summarized and characterized the hardware platforms in Figure 6.

VSA hardware implementations exhibit a broad spectrum of performance-energy tradeoffs tailored to specific application requirements (Figure 6(a)). General-purpose platforms, such as CPUs and GPUs, typically occupy the high-latency and high-energy region, reflecting their lack of VSA-specific parallelism and pipeline. In contrast,

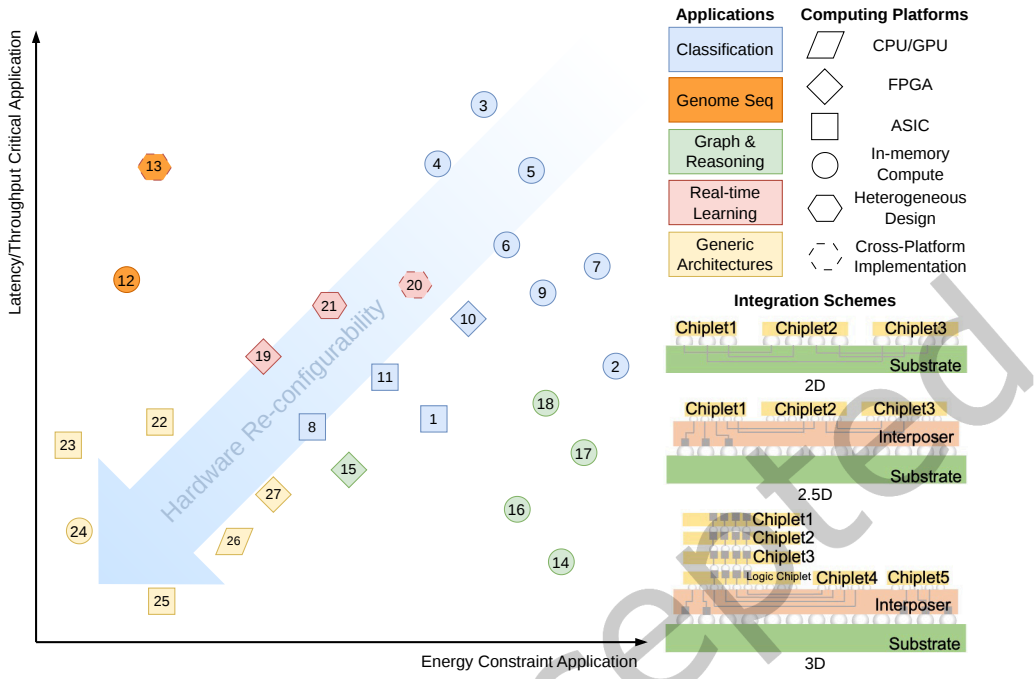

Fig. 5. High-level mapping of VSA compute platforms to application domains based on latency/throughput requirements and energy constraints. Each point corresponds to a prior work listed in Table 2; colors indicate application domains, while marker shapes denote hardware categories. The mapping evaluates the reconfigurability spectrum across platforms and highlights how heterogeneous integration in 2.5D/3D can fuse various computational primitives to meet the system requirements of diverse VSA workloads.

custom ASIC and in-memory computing solutions significantly push the Pareto frontier toward the lower-left corner, achieving several orders of magnitude improvement in both metrics. These trends underline the necessity of cross-layer co-design to align the reconfigurability of the hardware with the computational intensity of the target VSA kernels.

Based on the characteristics of hyperdimensional operations and the mechanism of different kernel implementations, we estimate the memory footprint bound of VSA applications (Figure 6(b)). In ASIC hardware architectures, the memory hierarchy includes both off-chip and on-chip memories that range from IMC units to buffers and register files. In our estimation, the configuration and usage of buffers and register files remain flexible, depending on the underlying dataflow strategy, while off-chip memory accesses can often lead to reduced system efficiency. Our methodology involves mapping operations that are feasible to be implemented in IMC cores. The IMC cores can also be used as pure storage for operands, aligning with prior IMC-based hyperdimensional computing designs that leverage IMC primarily for its high-density storage benefits. The factors that contribute to the difference in memory bound among applications include hypervector dimensionality, encoding methods, datasets, and function kernels on the algorithm side, as well as primitive designs and dataflow pipelining on the hardware side (further discussed in Section 5). These factors differ across applications and require tailored design space exploration. Accordingly, we build on insights from prior hardware implementations to guide our estimation of memory-bound performance.

The lower bound is estimated with a minimum memory requirement. At the algorithm level, the binary hyperdimensional representation and the encoding method that requires less memory capacity (random projection encoding) are used on the smallest dataset feasible for each application. At the hardware level, we adopt the state-of-the-art multi-bit PCM (3 bits per cell) to save the memory footprints. From the perspective of algorithm-hardware co-design, we evaluate the area-efficient accelerator designs while accounting for complexity and performance overhead. To estimate the upper bound, we consider the maximum memory requirement: using multi-bit hyperdimensional representation (4-bit in our case), the encoding method with the highest memory capacity requirement (n-gram encoding) on the largest dataset, and single-bit memory cells. We then analyze performance-efficient accelerator designs with the memory and area overhead factored in. The details of function kernels and their implementation and analysis for each application category are discussed below.

Applications that can be categorized as classification, clustering, outlier detection, and genomics share the same fundamental kernels in the training and inference phase, i.e., encoding and similarity check. The logic implementation difference between these applications results in minor memory discrepancy. The factorization involves primitives of unbinding, similarity, and projection. The reactive robotic reasoning includes training and recall.

The minimum memory requirement for all applications is a few kilobytes, whereas the maximum order of magnitude varies from 100 KB to 1 GB. The clustering and robotic reasoning need less memory due to the simple encoding method in clustering and the smaller number of sensors and actuators used in robot reasoning. In contrast, classification and outlier detection demand more memory to store large-scale class datasets, with genomics requiring the most due to long sequence samples.

It is observed that the memory footprints of some previous works fall outside the estimated range. These deviations are attributed to specific design strategies that necessitated a departure from our generic estimation framework. For instance, Dutta et al. [36] factored in the feature extraction by applying matrix-vector multiplication in the dense neural network as the preprocessing step of the feature encoding in classification. In contrast, Imani et al. [78] designed with cluster parallelism to have higher performance, requiring greater memory resources. The multi-step in-memory permutation was developed in [193]. These designs, while divergent, contribute to a broader understanding of the design space and demonstrate the flexibility of design strategies under varying constraints, prompting us to consider and analyze the arising trade-offs.

Suppose that the memory capacity is below the lower bound. In that case, the IMC cores may need reprogramming to handle changing operands, including inter- or even intra-kernel data traffic, which diminishes the benefits of parallel computing and reduced data movement between memory and compute units offered by IMC, ultimately leading to increased energy consumption and latency. Conversely, memory capacity exceeding the upper bound results in unnecessary area overhead and inefficiencies in energy and latency, particularly due to memory standby power or refresh cycles in the cases of charge-based memories.

4.3 Unified Co-Design Framework

The two co-design steps described above, *kernel formulation* and *feature-aware hardware realization*, can be approached independently. However, it is natural to think of these steps as being part of a unified co-design framework, bringing them into a merged design space for VSA-based cognitive systems. Figure 7 illustrates the co-design framework that integrates both software and hardware perspectives for VSAs. It spans from high-level cognitive tasks, through kernel formulation, down to hardware architectures. The primary objective of this framework is twofold: (1) it allows holistic optimization across the entire design spectrum. A unified co-design framework endeavors to optimize not only the individual steps of kernel formulation and hardware realization but also their collective impact on the VSA system with respect to performance, latency, and energy. (2) It facilitates data-driven refinement of the design space, offering capabilities for dynamic adaptation to evolving requirements

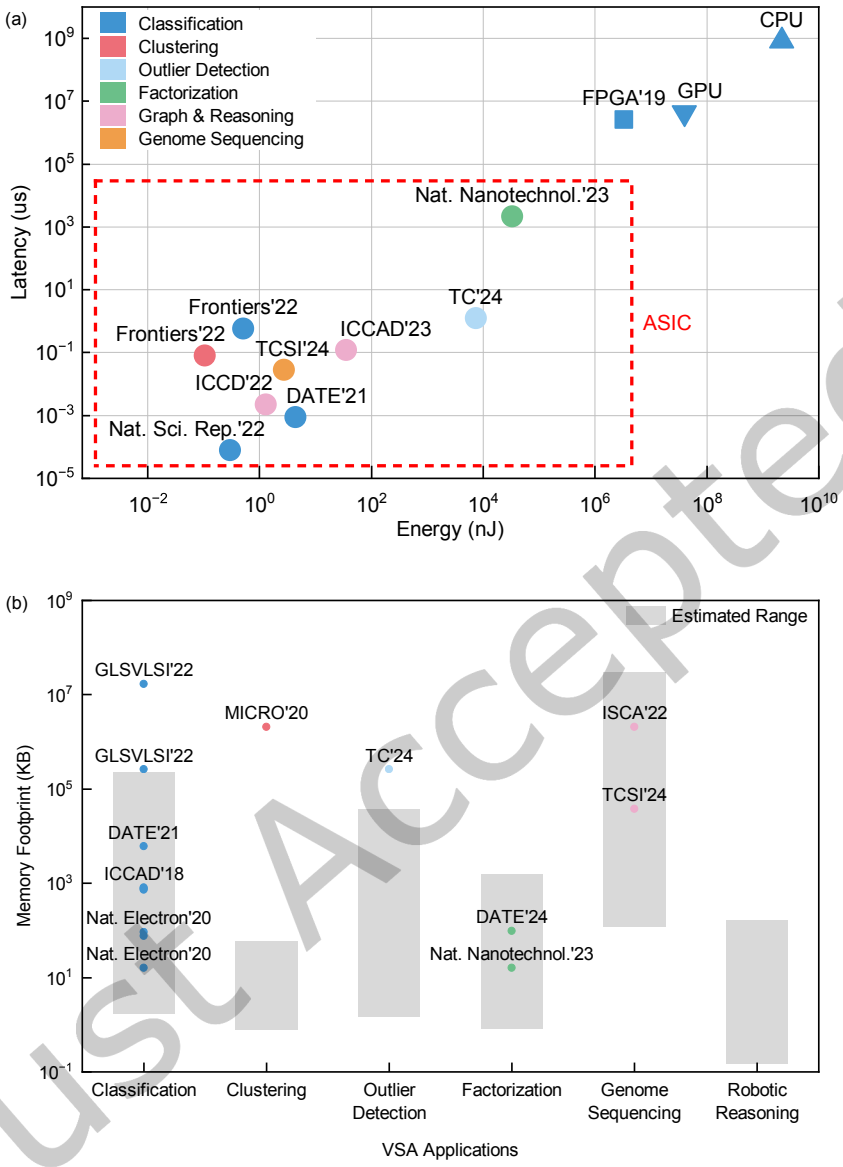

Fig. 6. Performance and resource characterization of VSA application categories across representative hardware implementations. (a) Comparison of average latency and energy per query across hardware platforms and application domains, benchmarked against representative datasets. (b) Application memory-footprint range estimation (bars) together with memory usage reported by prior implementations (points).

and technological landscapes. Through this co-design framework, hardware design methodologies for VSAs can be optimized, steering them towards a more systematic and top-down approach. In the following section, we explore an application case study that conceptualizes such a co-design framework.

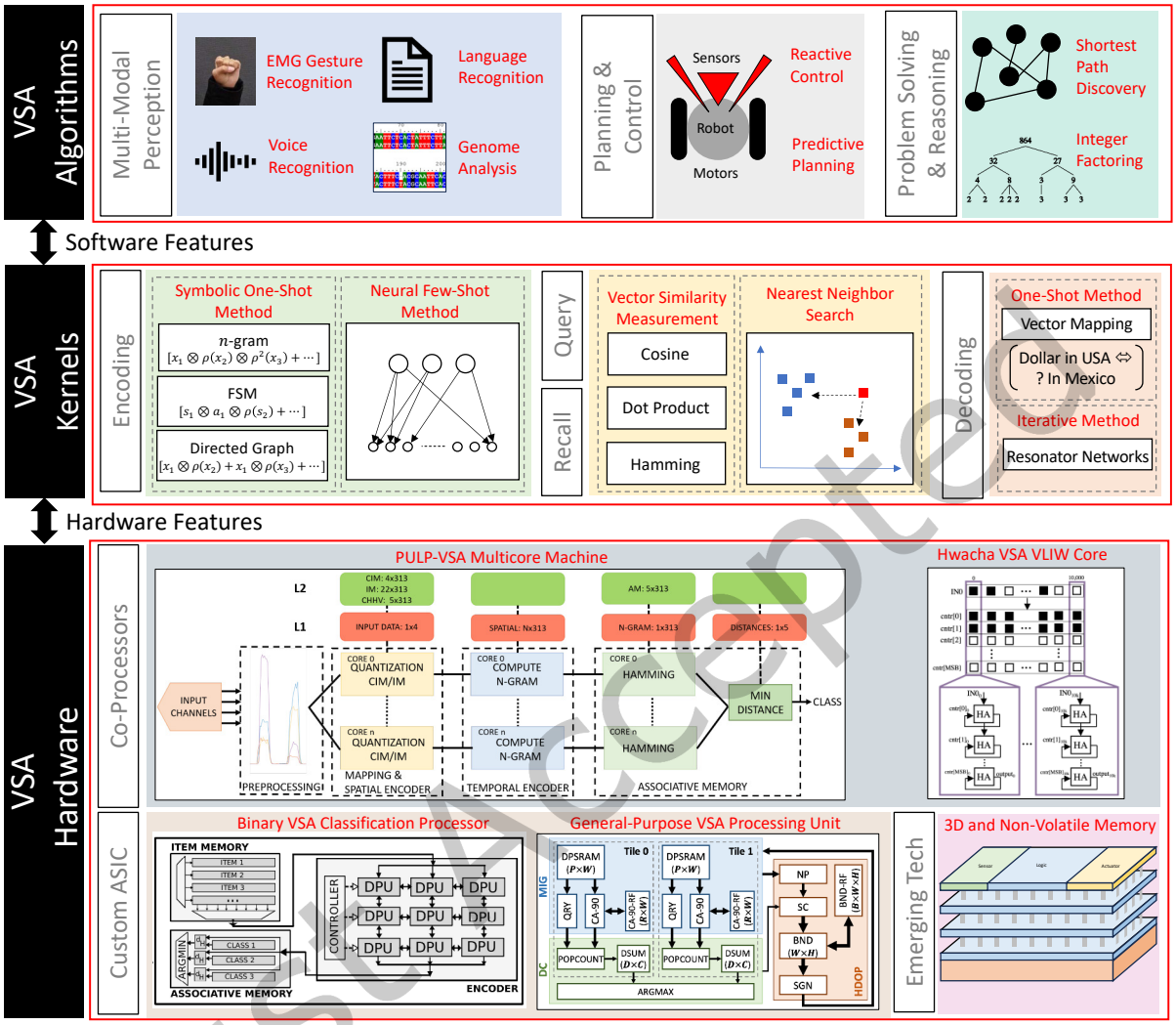

Fig. 7. Unified hardware/software co-design framework for VSAs. The framework bridges high-level cognitive algorithms and kernel formulations with hardware architectures through cross-layer abstractions, illustrating the convergence of software features with hardware realizations.

Given the advantages of in-memory computing architectures in terms of reducing data movement and enhancing MVM computation efficiency, elaborated in Section 3, we develop a generic IMC reconfigurable architecture template (Figure 8) as a synthesis and extension of state-of-the-art IMC-based VSA designs. This template is intended to be applied to complex VSA systems, enabling a flexible and reconfigurable approach that accommodates various precision, sparsity, and quantization requirements. The purpose of the template is to provide a hardware foundation for the versatile framework that can be leveraged to develop and implement VSA systems more effectively. Using classification as the baseline application, the subsequent discussion elaborates on the

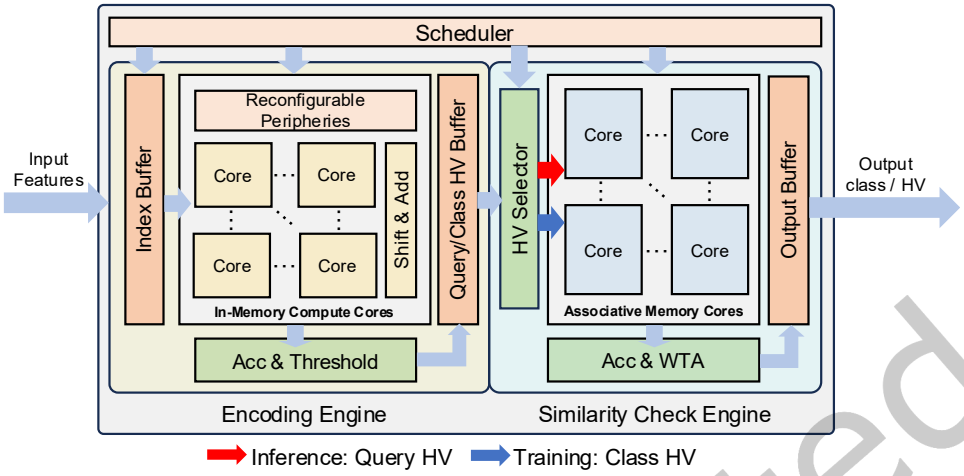

Fig. 8. An example IMC system design for VSA acceleration. The architecture comprises an encoding engine for in-memory binding and bundling, and a similarity check engine for associative memory search. Reconfigurable peripheries handle diverse requirements regarding vector dimension, precision, sparsity, and quantization. Accumulated (Acc) and Winner-Takes-All (WTA) logic enable class labeling by calculating and comparing similarities across all hypervector dimensions.

architectural modules and their functionalities within this proposed template. The modifications and adaptations of the architecture to suit other applications are extended in Section 5.

Our IMC architecture comprises two primary engines: the encoding engine and the similarity check engine. The encoding engine includes IMC cores, reconfigurable peripheries, adder and threshold logic, controller, and buffers, all of which enable high-dimensional computation. The IMC cores implement in-memory binding and bundling. Reconfigurable peripheries handle varying data precision (i.e., floating point or integer values), sparsity levels, and vector lengths, which are crucial for supporting diverse VSA workloads in IMC architectures. In particular, the sparsity scheduler enables efficient processing by skipping zero-valued computations and dynamically allocating resources based on sparsity patterns, an approach that aligns with prior work on sparse-aware CIM architectures [30, 51, 200]. Adder logic circuit aggregates the hypervectors for each class and adds (subtracts) the hypervector to (from) class hypervectors in the training phase. Threshold logic supports various scenarios, such as generating the binary class hypervectors from the summed multiple binary feature hypervectors or generating similarity metrics. Buffers in the encoding engine are partitioned to store item memory for input features, and to store generated query and class hypervectors. IMC peripheries (Figure 9) include sparsity scheduler, FP (floating-point) partitioner that separates mantissa and exponent, and a dimension divider for handling high-dimensional computations. The exponent computing module facilitates exponent and mantissa processing and integrates seamlessly with the IMC cores, enabling efficient floating-point computation acceleration. The floating-point MAC implementation includes exponent alignment and addition, and mantissa multiplication. First, the exponents are read out from the exponent buffer and added in pairs based on the mechanism of MAC implementation, and the maximum sum value among all pairs is determined. The exponent shift value for each pair is calculated by subtracting the maximum sum from all exponent sums. These shift values align the exponent part and adjust the mantissa product generated by the IMC cores. The final floating-point MAC result is obtained by integrating the aligned mantissa sum with the previously determined maximum exponent.

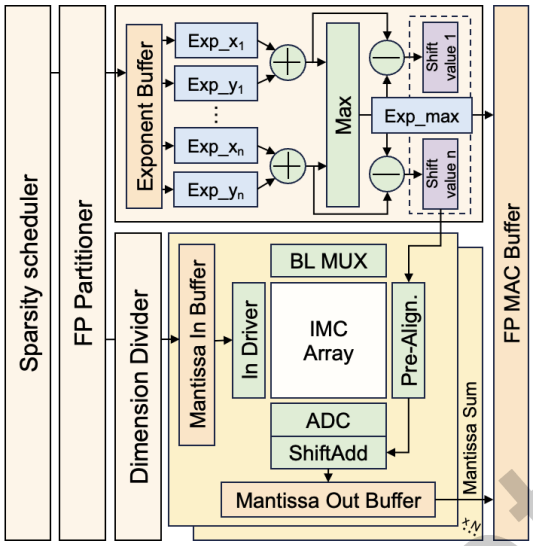

Fig. 9. In-memory compute cores and peripheries for the example VSA system design. The IMC peripheries include a sparsity scheduler for skipping zero-valued computations, a Floating-Point (FP) partitioner for scheduling mantissa and exponent, and a dimension divider for handling high-dimensional computations. Integrated with an exponent computing module for alignment and addition, the IMC cores utilize IMC arrays and accumulation logic for mantissa computation, enabling efficient FP MAC operations.

The similarity check engine includes the following components – associative memory cores, controller, adder and winner-takes-all logic (or comparator logic), and buffers – which enable the class labeling after computing the similarities. The class hypervectors are updated in associative memory cores in the training phase, prestored and fixed for the inference phase. Associative memory cores can be either in-memory computing cores or associative-memory cores depending on the similarity computation method. With the adder and comparator logic, the engine calculates the similarity between the query hypervector and each class hypervector by accumulating matching bit values in all dimensions. The class label of the query hypervector is determined and output based on the comparison result.

Translating this unified and reconfigurable IMC template into practical applications requires navigating inherent semi-quantitative tradeoffs, particularly between hardware flexibility and system latency/energy. For instance, as quantitatively demonstrated in our case study (Section 5.4) for time-critical multi-modal perception tasks, prioritizing reconfigurability by utilizing heterogeneous memory configurations (e.g., combining charge-based dynamic cores with NVM static cores) can yield up to an 86% reduction in latency and a 2.6× improvement in energy efficiency, compared to homogeneous NVM and eDRAM baselines, respectively. Ultimately, these semi-quantitative insights serve as first-order design guidelines; however, exhaustively resolving the multi-dimensional design space remains a quantification challenge, which necessitates the development of unified benchmarking tools as further discussed in Section 6.3.

5 Case Study for Hierarchical Cognition

Insights from the field of cognitive science have suggested that humans build complex models of the world to fulfill the fundamental tasks of cognition [82]. Cognitive workloads, which span from low-level perceptual processes that integrate massive sensory data from diverse sources to high-level problem-solving and reasoning, are inherently

both heterogeneous and hierarchical. These two characteristics motivate the selection of hierarchical cognition as a representative case study in this work. First, hierarchical organization mirrors the intricate layers of cognitive processing, forming a continuum that orchestrates our interactions with the environment. From a computational perspective, the research literature includes many VSA methods that can capture the underlying mechanisms of these cognitive processes. These methods collectively form a unified cognitive framework, which stands to benefit several application domains, in particular assistive technologies and autonomous systems [35, 130, 146]. Figure 7 illustrates the unified co-design framework with the case study of hierarchical cognition. Second, given the diverse nature of the applications and kernel operations involved in the cognition system, the corresponding hardware designs must also be tailored accordingly. Building on this, we present the first in-memory computing hierarchical cognition hardware system as a design example based on our IMC template introduced in Section 4. To demonstrate the practical impact of this methodology, we provide comprehensive quantitative evaluations of the instantiated architecture. As detailed in Section 5.4, by strategically navigating the heterogeneous memory allocation, our co-designed framework achieves substantial system-level benefits.

The following subsections detail the hierarchical cognitive co-design. We begin by exploring lower-level multi-modal perception (Section 5.1) and planning and control (Section 5.2), followed by problem solving and reasoning (Section 5.3). For each cognitive modality, we provide a brief overview of the relevant VSA methods, including their fundamental kernels and their implementations within the IMC template. Section 5.4 presents the key hardware design considerations from functional partitioning, dataflow mapping, to memory type selection, the impact of technology node scaling, and quantitative evaluations.

5.1 Multi-Modal Perception

An increasing number of sensor types (such as gyroscopes, accelerometers, and EEGs) are integrated into one edge device to detect and collect richer data in the environment. This multimodal nature of data presents challenges in processing and learning, specifically, how to exploit multi-modality to facilitate better learning and how to efficiently process large-scale data on resource-constrained embedded devices. VSA-based architecture unlocks potential solutions to efficiently learn from multimodal data at the edge [15, 132, 213]. The multimodal data from different sensor types are first encoded into hypervectors with uniform dimensions by utilizing the permutation-based encoder, and then the hypervectors of different modalities can be either bundled or processed through attention modules to learn intermodality correlations for downstream perceptual tasks.

VSA-based perception methods can also build on data structures such as graphs, inspired by the human brain that clusters data and represents information as a graph structure [8]. The objects and edges in the graph can show the correlation between objects, and memorization provides prior knowledge to keep the context and define confidence for downstream reasoning and decision-making. Specifically, high-dimensional vectors can be utilized to holographically represent the nodes and memorize the graph, which enables the known information (such as graph nodes and their connections) to be well memorized [124, 144, 152]. For a graph with V nodes and E edges, the node i memory is constructed by accumulating all node hypervectors connected to it as $M_i = \sum_j H_j$ where H is a random hypervector to the node and j represents all the neighbors of node i. The bundling of all associated hypervectors then generates a graph memory as $G = \sum_i H_i \oplus M_i$, where the graph memory is a compressed, invertible, and transparent model that can be used for downstream brain-like cognitive learning tasks.

VSA-based architecture also excels in cognitive perception when confronted with out-of-distribution (OOD) samples - an inevitably encountered scenario in many embodied AI applications, such as robotics and autonomous systems, when objects and scenes were not part of the training data distribution. The powerful concepts of binding, bundling, and projection from VSAs can be applied to the features from multiple layers in a DNN without requiring re-training or any prior knowledge of the OOD data. Hyperdimensional Feature Fusion (HDFF) presented such an example [197]. Specifically, the feature maps from multiple layers can be projected into a

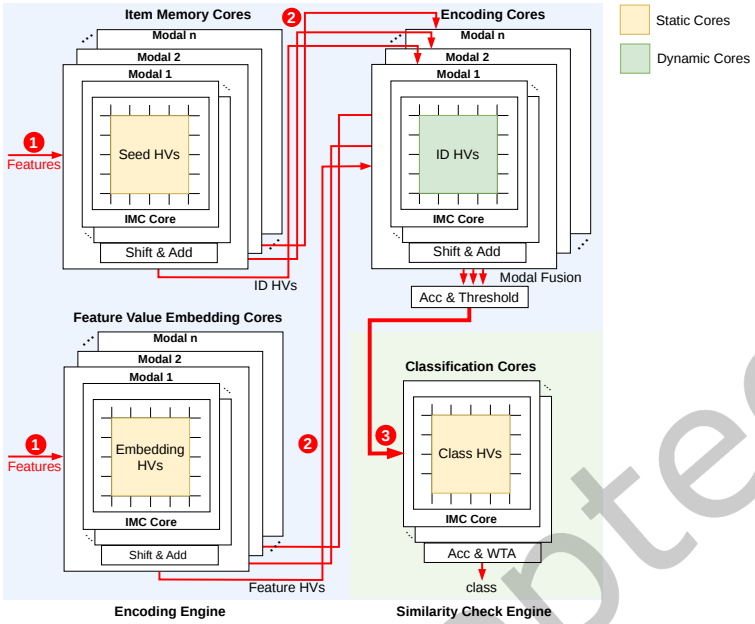

Fig. 10. An illustration of the IMC kernel design for multi-modal perception. Within the Encoding Engine, sensory features and their corresponding IDs are mapped into hypervectors using Feature Value Embedding Cores and Item Memory Cores, respectively. These hypervectors are processed in Encoding Cores to generate modal hypervectors, which are subsequently fused through near-memory bundling. In the Similarity Check Engine, the fused hypervectors are bundled to class prototypes (for training) or compared with class hypervectors (for inference) within the Classification Cores.

common vector space by using similarity-preserving semi-orthogonal projection matrices. During deployment, the projection and bundling operations are applied for a new input image, i.e., $y = h_1 \otimes h_2 \cdots \otimes h_l$ where h_i is the encoded high-dimensional vector from different network layers and y is the resulting single vector that serves as an expressive descriptor for the input image. The cosine similarity is then leveraged by the class representatives to identify OOD samples. The VSA-inspired technique paves the way to potentially address the limitations of DNN models in OOD scenarios, where they tend to fail silently in producing overconfident but erroneous predictions.

IMC-based hardware implementation leverages diverse memory characteristics to optimize different cores. Figure 10 illustrates an IMC multi-modal perception kernel design. The Feature Value Embedding Cores, Item Memory Cores, and Classification Cores are static cores; all memory cells are prestored, eliminating the need for additional memory writes during fusion operations. In contrast, Encoding Cores operate as dynamic cores as they process changing feature value/ID hypervectors as inputs. To illustrate the computation process, first, the features captured by modal sensors are mapped into feature value hypervectors in Feature Value Embedding Cores ①, where in-memory random projection is used in this case as an example. The IDs of the features are also mapped to ID hypervectors in Item Memory Cores ①, which serve as the in-memory look-up table. Each modal's generated feature value hypervectors and feature ID hypervectors are encoded into modal hypervectors through in-memory binding and bundling in Encoding Cores ②. A temporal encoding step might be needed to create a hypervector of each modality according to the features capturing process, which can be realized by the permutation function of shift-and-add logic in the template. As the final step in the Encoding Engine, the modal fusion is implemented through near-memory bundling of each modal hypervector in Encoding Cores. In

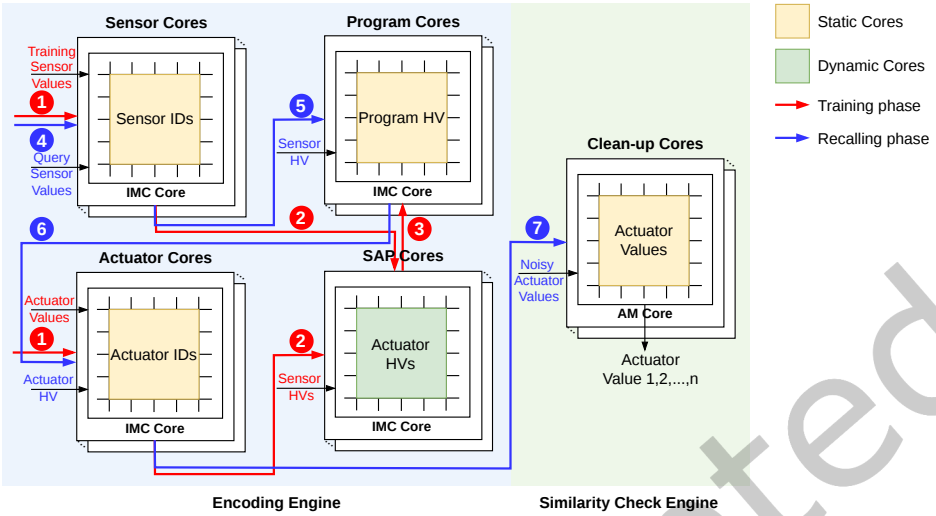

Fig. 11. An illustration of the IMC kernel design for a reactive robot navigation tasks. The dataflow comprises training and recalling phases. During training, each sensor-actuator pairing (SAP) is bundled into the program hypervector. In the recalling phase, query sensor data are bound to the pre-stored program to produce noisy actuator hypervectors, which are then cleaned up via associative memory search within the Clean-up Cores.

Similarity Check Engine, the fused hypervectors are bundled to class prototype in the training phase or compared with class hypervectors in the inference phase in Classification Cores ③.

5.2 Planning and Control

While pattern recognition is the most popular application of VSAs, there is a wide body of research showing that VSAs can also facilitate planning and control applications. Within this domain, VSA methods can be broadly classified into two primary groups: *reactive* and *predictive*. A reactive VSA method allows an agent to run task demonstrations while encoding its interactions as pairs of sensor input and actuator output [134, 141]. The so-called reactive robotic behavior is eventually encoded as a single high-dimensional vector, thus enabling an agent to promptly respond (through a quick memory search) to immediate stimuli during navigation [128, 129]. A predictive method, on the other hand, aims to learn an abstract view or a topology of the environment's state space. Remarkably, VSAs have demonstrated great strengths in representing multiple data structures [103], and hence they can be used to learn structural, most often graph, representations of an environment [142, 177]. Such a VSA representation is often referred to as *a cognitive map* [126]. Additionally, planning and control are usually vulnerable to software or hardware disturbance [69, 190]; thus, the robustness characteristics of VSA computing benefit the safety of autonomous agents.

An IMC kernel design is demonstrated through an example of reactive robot navigation tasks [141] (Figure 11), utilizing static and dynamic cores to optimize different operations. In the 2-D navigation task, each of the sensors and the actuators (for the robot movement direction in this case) is assigned a random ID hypervector, which is bound with input/output feature value hypervectors to form sensor-action pairs, enabling the system to learn for future tasks. The task includes a training phase and a recalling phase. During training, the in-memory hyperdimensional operations are only implemented in the encoding engine, including the Sensor Cores, Actuator

Cores, and Program Cores that handle operations with static memory cores mapping, and the dynamic memory cores, i.e., Sensor-Actuator Pairing (SAP) Cores where both inputs and memories change with training samples. The training phase starts with encoding each sensor value into a sensor hypervector through in-memory binding (XOR) followed by in-memory bundling (addition) operations to generate a single sensor hypervector representing the training sample, where ID hypervectors for all sensors are pre-stored in the Sensor Cores ①. A similar process is applied to actuator data in the Actuator Cores. Thus, the sensor-actuator pairs of each training sample are created. For each pair, the sensor hypervector (input) is bound with the corresponding actuator hypervector (stored in memory). The sensor-actuator pairs from all training samples are bundled into the program hypervector in SAP Cores ②. The generated program hypervectors are stored in Program Cores memory, preparing for the recalling phase ③.

During the recalling phase, both the encoding engine and the similarity check engine are mapped with pre-stored static memories, where some memories are reused from the training phase. The same encoding process is performed on the query input sensor data in Sensor Cores ④. The query sensor hypervector is bound to the trained program hypervector pre-stored in Program Cores ⑤. The generated actuator hypervector goes through the unbinding with actuator ID hypervectors in Actuator Cores, resulting in a noisy actuator hypervector for each actuator ⑥. These actuator hypervectors undergo the clean-up process in the Similarity Check Engine by comparing against the stored hypervector representations of possible robot actuation values in Clean-up Cores ⑦. The actuation with the highest similarity to the noisy actuator hypervector is provided as the action output (move in one of the directions in this case) to the robot.

5.3 Problem Solving and Reasoning

VSAs are key enablers for neuro-symbolic methods, offering highly efficient solutions for a range of reasoning and analogy-making problems [39, 64]. Notably, VSA methods for reasoning can be broadly classified into two groups: *rule-based* and *disentanglement-based*. The objective of rule-based methods is to extract relevant information from a composite vector and transfer it to unfamiliar contexts where the original items may not be known *a priori*. Consider an example that involves the following rules:

$$States = [name \otimes USA + cur \otimes DOL],$$

$$Mexico = [name \otimes MEX + cur \otimes PES],$$

$$F = States \otimes Mexico$$

A typical scenario for rule-based reasoning is when we need to find an answer to the question: "*What is the Dollar of Mexico?*" [90], which is formulated as a function $DOL \otimes F$. This function computes the reverse mapping of F, aiming to find what in *Mexico* corresponds to DOL. The returned vector, however, is a distorted version of the correct answer (PES) because of the cross-talk noise coming from all other elements. The correct answer can be recovered in turn using an implementation of the clean-up memory model. The described approach can be generalized and hence applied to more complex scenarios, such as Raven's progressive matrices [67] and analogical mapping through graph isomorphism [48].

One more challenging scenario arises when reasoning involves multiple unknown factors. For instance, suppose we have a composite vector $f = a \otimes b \otimes c$, which is defined by three unknown factors: a, b, and c. These factors are unknown because we only have access to the codebooks (A, B, and C), which represent the high-dimensional item sub-spaces for these factors. To recover the original items of f using the rule-based approach described earlier, we would need to search through all possible combinations of the factors, which becomes increasingly complex as the number of factors grows. In such scenarios, the disentanglement-based approach (resonator networks) comes into play [42].

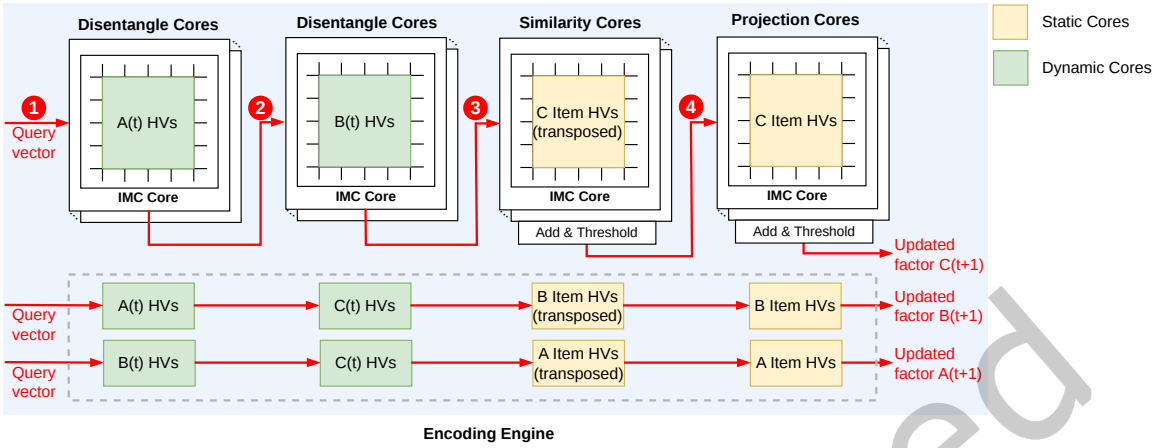

Fig. 12. An illustration of the IMC reasoning kernel design for a three-feature factorization task. This dataflow demonstrates the parallel factoring scenario, where all unknown factors are updated simultaneously: the query vector undergoes iterative unbinding via dynamic Disentangle Cores, followed by similarity and projection steps in static cores to achieve factor convergence. While individual features can also be handled sequentially, the illustrated parallel approach maximizes throughput at the cost of a larger memory footprint.

The disentanglement-based reasoning approach formulates a compositional state-space model, which superposes all possible combinatorial solutions. Given this model, distributed disentanglement computations are performed while searching for the exact feature or item vectors. One could employ this approach to disentangle f, defined earlier, into its factors. The state-space equations that describe this method are given as follows:

$$\hat{a}(t+1) = g\big(AA^{\top}(f \otimes \hat{b}(t) \otimes \hat{c}(t))\big); \quad A = [a_1\ a_2\ \ldots]$$
$$\hat{b}(t+1) = g\big(BB^{\top}(f \otimes \hat{a}(t) \otimes \hat{c}(t))\big); \quad B = [b_1\ b_2\ \ldots]$$
$$\hat{c}(t+1) = g\big(CC^{\top}(f \otimes \hat{a}(t) \otimes \hat{b}(t))\big); \quad C = [c_1\ c_2\ \ldots]$$

Here, $g(\cdot)$ is the sign function; t is a time step; $\hat{a}$, $\hat{b}$, and $\hat{c}$ hold the predicted values of the factors a, b, and c, respectively. The term $AA^{\top}\hat{x}$ represents the update role for $\hat{a}$, which computes the dot product between the reverse-mapping vector $\hat{x}$ and the item vectors of A, scales these item vectors using the dot-product (weight) results, then bundles all the weighted vectors to update the state.

Figure 12 illustrates an IMC reasoning kernel design in an example of a three-feature factorization task. The query hypervector is made up of the characteristics from all features, resulting in the factoring for each feature. Take feature C factorization as an example, the query vector first unbinds the features A and B in sequence by undergoing in-memory binding in Disentangle Cores (①-②). The unbinding result goes through Similarity Cores and Projection Cores to implement the dot product with C feature codebooks and get factor C updated in this iteration (③-④). Multiple iterations are needed until each feature converges to the correct factorization with a high similarity for a code vector. In this task, factorization of each feature can either be handled sequentially or in parallel, where the IMC cores mapping and overall memory resources utilization vary substantially.

In the case of sequential factoring, at each time step, the estimation update begins with the first factor. Subsequently, the second factor is updated using the newly estimated first factor and the most recent estimates of the remaining factors for the following similarity computation and projection process. This update process continues iteratively, with each subsequent factor updated based on the most current estimates available for all

preceding factors. During the factorization, Disentangle Cores, similarity Cores, and Projection Cores are all dynamic cores. The feature types and their feature factors estimation stored in disentangle cores change with the factoring iteration, and the stored feature codebooks for similarity and projection steps vary depending on the feature type in each factoring iteration. In the case of parallel factoring (Figure 12), all features are updated at the same time step and are mapped in cores for in-memory binding for the next iteration. The memory footprint is scaled with the number of features. While the Disentangle Cores are dynamic cores, storing updated feature factors estimation with fixed feature types, the Similarity Cores and Projection Cores are static cores, with each feature codebook written once into their respective static cores.

5.4 Hardware Realization

Developing a hardware platform for the above framework involves careful consideration of the requirements of each layer. This can be achieved by exploring the kernels, computational characteristics, and constraints of each layer and examining the hardware technologies that are aligned with them.

IMC-based architectures offer a natural solution due to their ability to minimize data movement and provide efficient in-memory vector operations. As introduced in Section 5.1, IMC kernel design tailored for multi-modal perception leverages static and dynamic memory cores to optimize different operations: Static cores precompute hypervectors, eliminating the need for additional memory writes during fusion operations; Dynamic cores handle feature value/ID hypervectors as inputs, supporting real-time encoding and temporal processing through in-memory operations. We evaluated the energy, latency, and area overhead of IMC architectures that integrate static and dynamic cores across three cognitive VSA applications (Figure 13) and the impact of technology scaling (Figure 14).

Our analysis considers various memory core options, including charge-based memories [27, 107, 179, 192, 201] (e.g., eDRAM and SRAM) and non-volatile memories [2, 16, 17, 24–26, 52, 58, 72, 81, 123, 138, 139, 145, 160, 170, 174, 194, 195, 203, 210] (e.g., RRAM and MRAM). As shown in Figure 13, diverse memory types can be applied homogeneously across all IMC cores—either dynamic or static—or heterogeneously, with charge-based memories used in dynamic cores to efficiently handle frequent memory writes while mitigating standby energy overhead by minimizing SRAM leakage time and eDRAM refresh cycles, and NVMs in static cores to reduce write overhead and enhance energy-area efficiency. In architectures with heterogeneous memory cores, applying charge-based memories and NVM to dynamic and static IMC cores, respectively, results in relatively lower energy consumption compared to homogeneous memory configurations. Among the evaluated workloads, multi-modal perception and factorization have the worst energy efficiency when using RRAM, primarily due to frequent writes in dynamic cores and the inherently high write overhead of RRAM. Robot navigation performs worst with eDRAM, as the frequent refresh operations and retention time required during training across multiple samples incur significant energy overhead. Heterogeneous memory cores also demonstrate higher area efficiency compared to using charge-based memories alone, owing to the comparable dynamic and static IMC core memory footprint requirements and high-density nature of NVM.

As described in Section 5.1, multi-modal perception involves a computational process of continuous data sampling and HD encoding, with data extracted from a large number of sensory inputs. It is therefore logical to think of multi-modal perception as a time-critical application where data needs to be encoded in a timely manner [132]. In other words, a hardware platform developed to realize multi-modal perception needs to be positioned within the top part of Figure 5. As shown in Figure 13, heterogeneous MRAM/eDRAM and MRAM/SRAM architectures achieve over 86% lower latency compared to RRAM while maintaining latency performance close to eDRAM, which offers the lowest latency. Additionally, these configurations improve energy efficiency by up to 2.6× compared to eDRAM, balancing both speed and power effectively.

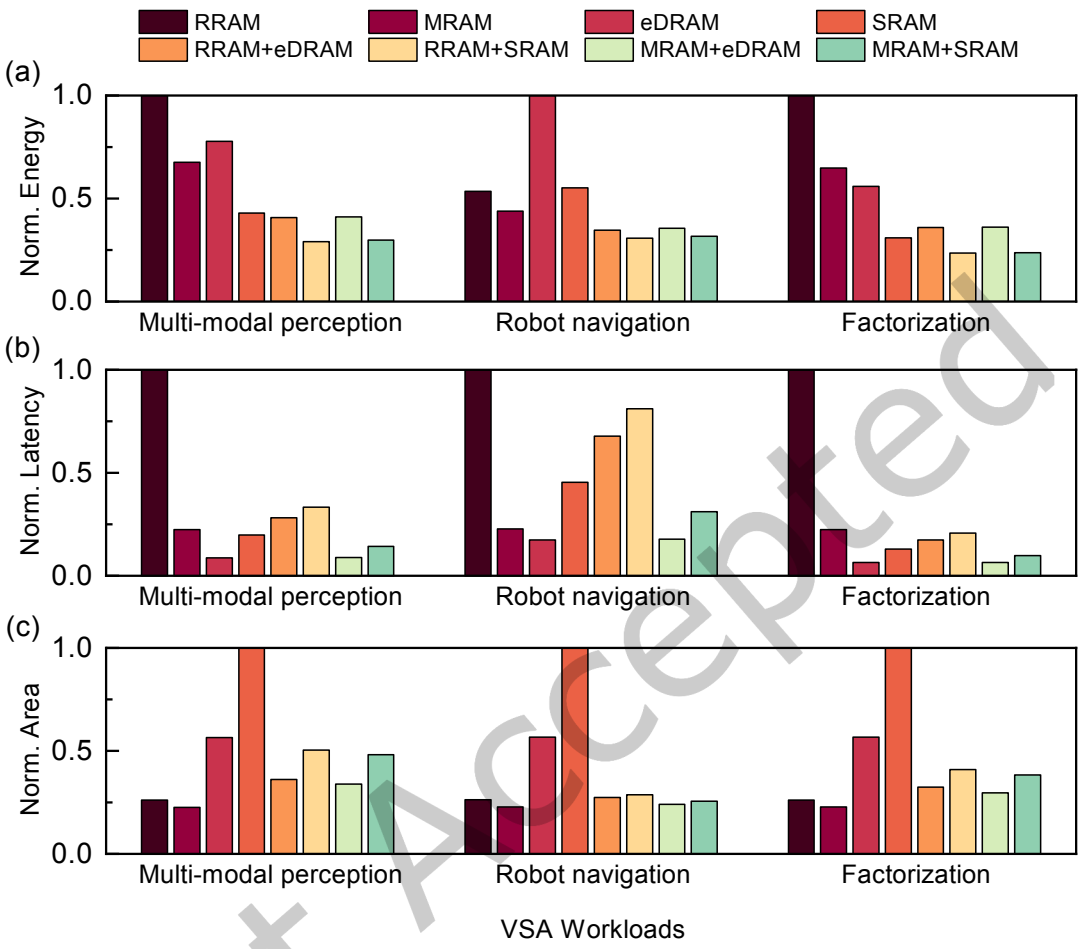

Fig. 13. Normalized energy (a), latency (b), and area (c) benchmark across IMC architectures with different memory technologies for VSA workloads. The heterogeneous design, combining charge-based memory for dynamic cores and NVM for static cores, improves total energy and latency compared to the single-memory IMC baselines.

For the reasoning layer (Section 5.3), NVM technologies play a crucial role in maintaining the persistent representations of knowledge and providing technology-assisted robustness to the resonator networks. These technologies are particularly well-suited for reasoning tasks due to their energy efficiency, non-volatile nature, and fast access times [111]. Moreover, these memory technologies can be organized hierarchically (e.g., via 3-D stacking) to mirror the structure of the reasoning kernels, allowing for quick retrieval of stored information during iterative reasoning processes [187]. IMC architectures offer an intrinsic advantage by reducing data movement and enabling energy-efficient vector operations. Combining NVM and IMC enables parallelized vector operations, low-energy iterative updates, and scalable memory management, ensuring efficient and robust execution of reasoning tasks. Overall, reasoning computations can be realized with hardware platforms that are positioned at the bottom right part of Figure 5, indicating that reasoning kernels are not time-critical, and energy efficiency

remains a key optimization goal. In this context, heterogeneous MRAM/SRAM and RRAM/SRAM architectures are well-suited for low-power requirements, reducing energy consumption by over 76% compared to RRAM. At the same time, these architectures maintain performance comparable to eDRAM, the lowest-latency option.

In Figure 14, energy-delay product (EDP) and area are benchmarked across 22 nm, 40/45 nm, and 65 nm technologies using different memories under the multi-modal perception workload. As the node goes from 65 nm to 22 nm for edge AI hardware and embedded systems, the average EDP and area overhead across the three memory configurations, SRAM alone, NVM alone, and hybrid SRAM/NVM, decrease significantly, aligning with the expectations of technology scaling. Hybrid SRAM/NVM and SRAM-only architectures exhibit improved EDP compared to NVM-only counterparts, primarily due to the mitigation of NVM write overhead. However, this advantage diminishes at scaled nodes (e.g., 22 nm), where the EDP performance of hybrid SRAM/NVM and SRAM-only designs becomes comparable. This trend is likely attributed to the fact that technology scaling primarily benefits silicon-based components, whereas the NVM device characteristics are not directly tied to silicon technology nodes. These observations suggest that adopting advanced technology nodes in VSA kernel implementations can yield system-level energy and performance gains, but the specific NVM device characteristics decoupled from node-specific silicon CMOS performance should be carefully considered and exploited during early-stage design exploration at the architecture level, with awareness of technology constraints. In addition, all configurations involving NVM—including NVM alone and hybrid SRAM/NVM—consistently demonstrate superior density across all technology nodes. This makes hybrid charge-based memories and NVM particularly promising candidates for energy-efficient and compact integration in edge VSA systems, where minimizing footprint and maximizing memory capacity are critical.

However, it is equally important that the developed hardware platform remains reconfigurable to support different encoding schemes for sensory data [32]. A hybrid approach that integrates IMC-based computation with digital FPGA or ASIC implementations would enhance both computational efficiency and adaptability across various applications. At the same time, advancements in memory technologies introduce additional considerations. For instance, state-of-the-art refresh-free eDRAM and ultra-low-power SRAM designs have the potential to further optimize power and performance trade-offs. As memory technologies continue to evolve, the optimal choice of memory for different applications may shift, reinforcing the need for flexible hardware platforms. In this context, digital processors based on FPGA or ASIC implementations would be particularly beneficial in maintaining reconfigurability while adapting to emerging memory innovations.

6 Research Challenges and Opportunities

This section discusses major challenges for VSA system design and highlights opportunities for future research directions.

6.1 Modular Formulation of VSA Kernels and Features

The field of VSA computing is fast-changing, and thereby we foresee a significant increase in the number of newly proposed algorithms in the coming years [104]. Being an algebra, VSA algorithms intrinsically share similar computational features and statistical properties, allowing designers to build libraries or packages of VSA functions [60]. It is therefore natural to seek a modular approach when formulating VSA kernels. Adopting this approach requires an optimization framework that assembles VSA kernels hierarchically in a plug-and-play fashion. This approach opens the door for *hardware-aware* exploration of VSA methods without greatly delving into details of hardware design. Another key benefit of adopting a modular design approach is the capability to address the lack of reconfigurability in existing hardware-constrained scenarios. By defining granular and reusable computational primitives, we can adapt and reconfigure deployed VSA kernels into unseen scenarios and varying precision requirements on the fly. This dynamic adaptability can eventually be transformed into a

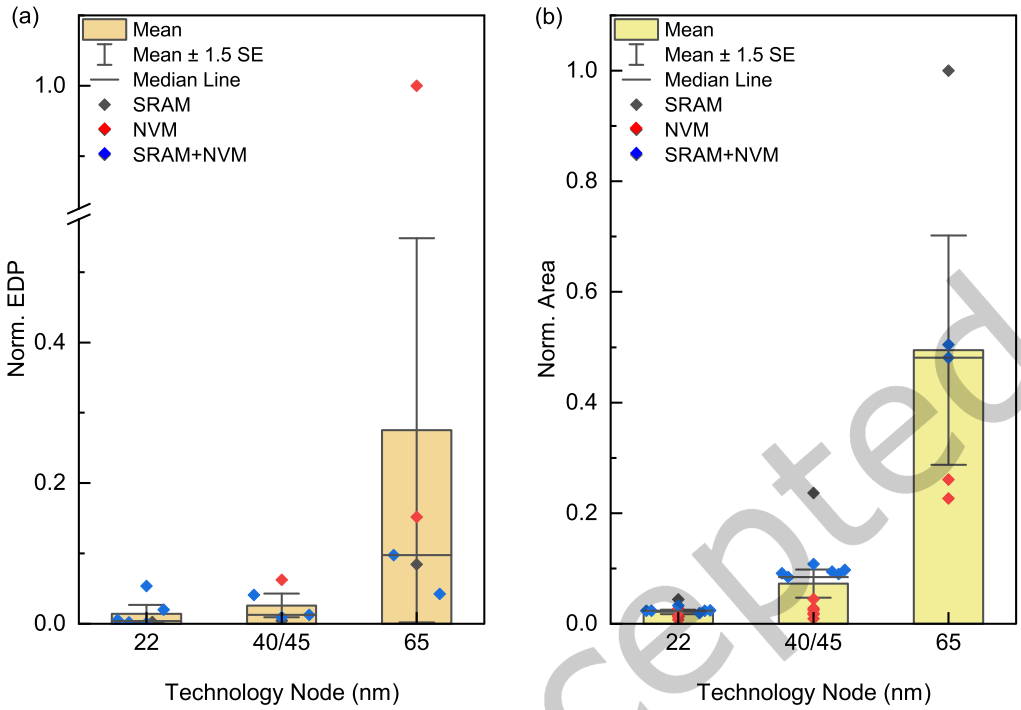

Fig. 14. Normalized energy-delay product (EDP) and area benchmark across sample technology nodes and memory configurations for multi-modal perception workload. While scaling benefits silicon memories (e.g., SRAM), specific NVM device characteristics decoupled from node-specific silicon CMOS performance should be fully exploited in co-optimization to achieve optimal EDP while maintaining the high memory density beneficial for edge VSA systems.

real-time reconfiguration framework—akin to an "operating system" tailored specifically for VSA methods—that orchestrates heterogeneous computational primitives and helps bridge the gap between algorithms and flexible hardware deployment.

6.2 Pareto-Front Exploration of Optimized Hardware Mapping and Compositional VSA Modeling

Compositionality is one of the key characteristics of VSA methods. For instance, attribute vectors resulting from multiple tasks (e.g., perception and reasoning) can be bundled/superposed to generate more complex high-dimensional structures in a hierarchical fashion [108]. This provides potential for a generic hardware design with maximally reusable modules. On the other hand, an efficient hardware design seeks to facilitate seamless data transfer devoid of contention across all abstraction levels, spanning from individual units to the system-on-chip level. Hence, while the computing-in-superposition nature inherent in VSAs creates massive opportunities for theoretical explorations, it may also pose challenges in terms of distributed processing and contention-free utilization of critical memory resources. Therefore, an efficient strategy for hardware mapping of VSA models should determine the granularity of compositionality that can be sustained at varying levels of the system-on-chip. The balance between the level of computational complexity supported at local units and the efficiency of distributed processing is best visualized through a spectrum of Pareto-front mapping solutions [196]. Considering

the synergistic nature of VSA operators, it is also highly desirable to develop frameworks for hardware-aware vector-symbolic architecture search for efficient model-hardware co-development [162].

6.3 Unified Benchmarking of VSA Hardware/Software Co-Design Flows

The myriad combinations of VSA algorithms and hardware technologies have aggravated the demand for a systematic evaluation methodology that enables standardized, fair comparisons of different VSA design flows. This gives rise to the importance of developing unified benchmarking suites, akin to benchmarks in conventional machine-learning systems [158], robotic systems [109, 125], compute-in-memory systems [150], etc. Such benchmarking allows for assessing the performance of various VSA algorithms on the same hardware architecture, and at the same time facilitates the mapping and evaluation of the same algorithm on multiple hardware platforms. A unified benchmark for VSA should include various cognitive tasks, including learning, common-sense reasoning, planning, and decision-making [175] to test the robustness of different VSA workloads.

Central to this benchmarking effort is the need for robust simulation tools that facilitate granular and quantitative tradeoff analysis. To provide insights for future designers, the tools should evaluate the impact of critical design parameters, which include hypervector dimensionality, sparsity levels, and temporal folding factors, on the tension between hardware reconfigurability and latency. By benchmarking algorithmic requirements (e.g., accuracy) against hardware metrics (e.g., energy efficiency and performance), the simulation frameworks enable researchers to systematically explore the design space and investigate tradeoffs across a wide range of hardware designs, from NVM-based to FPGAs. Additionally, given the increasing complexity of VSA algorithms and the cross-stack nature, building a push-button flow with VSA task requirements as input to automatically generate accelerator design is critical [186]. We envision the agile framework will intelligently search the huge design space and automatically choose the optimal algorithm-hardware parameters with the help of modular kernels, benchmarking, and machine learning-assist methods.

6.4 Cross-Layer Integration of Heterogeneous NVSA Workloads

NVSA workloads are heterogeneous because they interleave multiple modes of information processing. As explained in Section 4, this heterogeneity enables capabilities such as spatial-temporal reasoning [67, 209], multi-input processing [127], and robust out-of-distribution data processing [65]. These capabilities are difficult to obtain with a single specialized architecture (e.g., DNN). Realizing these benefits, however, requires systematic integration across the computing stack, from software abstractions and runtime systems to architectures and circuits [74, 188, 191]. Yet, many NVSA models do not map cleanly onto existing hardware; therefore, making progress in this field cannot be achieved unless such a gap is closed—a situation often referred to as *winning the hardware lottery* [68]. To bridge this gap and realize the hardware-software synergy discussed in Section 4.2, we highlight three concrete cross-layer directions. First, the realization of reconfigurable dataflows is essential to enable hardware to switch dynamically between operations in different cognitive modalities (e.g., dense tensor arithmetic in neural and element-wise binding/bundling in symbolic) while achieving the highest compute utilization. Second, the development of a unique memory system is needed to seamlessly interleave different types of cognitive data (e.g., scalars, tensors, and HD vectors). Finally, defining a "cognitive" interconnection protocol with a co-designed heterogeneous chiplet system is critical for coordinating efficient data movement and task partitioning across units.

The above challenges (and several others) provide ample opportunities for research development at all levels. Exploring innovative solutions to these challenges and embracing interdisciplinary approaches can pave the way for even more breakthroughs in VSAs, significantly advancing cognitive systems on a broader scale.

7 Conclusion

This paper surveys the current state of VSA systems and emphasizes the crucial role of hardware/software co-design. We explore diverse VSA algorithms and hardware technologies, and present a new framework bridging the gap between software-level explorations and efficient hardware designs. Discussions on open research challenges and opportunities serve as a call for action on cross-layer collaborations to bring this promising paradigm closer to ubiquitous applications.

References

[1] Rahul Agarwal, Patrick Cheng, Priyal Shah, Brett Wilkerson, Raja Swaminathan, John Wuu, and Chandrasekhar Mandalapu. 2022. 3D packaging for heterogeneous integration. In *2022 IEEE 72nd Electronic Components and Technology Conference (ECTC)*. IEEE, 1103–1107.

[2] Amogh Agrawal, Aayush Ankit, and Kaushik Roy. 2018. SPARE: Spiking neural network acceleration using ROM-embedded RAMs as in-memory-computation primitives. *IEEE Trans. Comput.* 68, 8 (2018), 1190–1200.

[3] Junwhan Ahn, Sungpack Hong, Sungjoo Yoo, Onur Mutlu, and Kiyoung Choi. 2015. A scalable processing-in-memory accelerator for parallel graph processing. In *Proceedings of the 42nd Annual International Symposium on Computer Architecture* (Portland, Oregon) *(ISCA '15)*. Association for Computing Machinery, New York, NY, USA, 105–117. doi:10.1145/2749469.2750386

[4] Kerem Akarvardar and H-S Philip Wong. 2023. Technology prospects for data-intensive computing. *Proc. IEEE* 111, 1 (2023), 92–112.

[5] Tharindu Bandaragoda, Daswin De Silva, Denis Kleyko, Evgeny Osipov, Urban Wiklund, and Damminda Alahakoon. 2019. Trajectory clustering of road traffic in urban environments using incremental machine learning in combination with hyperdimensional computing. In *2019 IEEE intelligent transportation systems conference (ITSC)*. IEEE, 1664–1670.

[6] Hamze Barkam, Sanggeon Yun, Paul R Genssler, Che-Kai Liu, Zhuowen Zou, Hussam Amrouch, and Mohsen Imani. 2024. In-memory acceleration of hyperdimensional genome matching on unreliable emerging technologies. *IEEE Transactions on Circuits and Systems I: Regular Papers* (2024).

[7] Hamza Errahmouni Barkam, Sanggeon Yun, Hanning Chen, Paul Gensler, Albi Mema, Andrew Ding, George Michelogiannakis, Hussam Amrouch, and Mohsen Imani. 2023. Reliable hyperdimensional reasoning on unreliable emerging technologies. In *2023 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 1–9.

[8] Danielle S Bassett and Olaf Sporns. 2017. Network neuroscience. *Nature neuroscience* 20, 3 (2017), 353–364.

[9] Graham Bent, Chris Simpkin, Yuhua Li, and Alun Preece. 2022. Hyperdimensional computing using time-to-spike neuromorphic circuits. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*. 1–8.

[10] Alessio Burrello, Kaspar Schindler, Luca Benini, and Abbas Rahimi. 2019. Hyperdimensional computing with local binary patterns: One-shot learning of seizure onset and identification of ictogenic brain regions using short-time iEEG recordings. *IEEE Transactions on Biomedical Engineering* 67, 2 (2019), 601–613.

[11] Han Cai, Tianyao Chen, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Efficient architecture search by network transformation. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[12] Han Cai, Ligeng Zhu, and Song Han. 2018. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332* (2018).

[13] Rishikanth Chandrasekaran, Kazim Ergun, Jihyun Lee, Dhanush Nanjunda, Jaeyoung Kang, and Tajana Rosing. 2022. Fhdnn: Communication efficient and robust federated learning for aiot networks. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*. 37–42.

[14] Cheng-Yang Chang, Yu-Chuan Chuang, Chi-Tse Huang, and An-Yeu Wu. 2023. Recent progress and development of hyperdimensional computing (HDC) for edge intelligence. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS)* 13, 1 (2023), 119–136.

[15] En-Jui Chang, Abbas Rahimi, Luca Benini, and An-Yeu Andy Wu. 2019. Hyperdimensional computing-based multimodality emotion recognition with physiological signals. In *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 137–141.

[16] Meng-Fan Chang, Che-Wei Wu, Chia-Cheng Kuo, Shin-Jang Shen, Sue-Meng Yang, Ku-Feng Lin, Wen-Chao Shen, Ya-Chin King, Chorng-Jung Lin, and Yu-Der Chih. 2013. A low-voltage bulk-drain-driven read scheme for sub-0.5 V 4 Mb 65 nm logic-process compatible embedded resistive RAM (ReRAM) macro. *IEEE journal of solid-state circuits* 48, 9 (2013), 2250–2259.

[17] Tung-Cheng Chang, Yen-Cheng Chiu, Chun-Ying Lee, Je-Min Hung, Kuang-Tang Chang, Cheng-Xin Xue, Ssu-Yen Wu, Hui-Yao Kao, Peng Chen, Hsiao-Yu Huang, et al. 2020. 13.4 A 22nm 1Mb 1024b-read and near-memory-computing dual-mode STT-MRAM macro with 42.6 GB/s read bandwidth for security-aware mobile devices. In *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 224–226.

[18] Vikas Chaudhary, RS Bhatia, and Anil K Ahlawat. 2014. A novel Self-Organizing Map (SOM) learning algorithm with nearest and farthest neurons. *Alexandria Engineering Journal* 53, 4 (2014), 827–831.

[19] Hanning Chen, Mariam Issa, Yang Ni, and Mohsen Imani. 2022. DARL: Distributed Reconfigurable Accelerator for Hyperdimensional Reinforcement Learning. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*. 1–9.

[20] Hanning Chen, M Hassan Najafi, Elaheh Sadredini, and Mohsen Imani. 2022. Full stack parallel online hyperdimensional regression on FPGA. In *Proceedings of the International Conference on Computer Design (ICCD)*. 517–524.

[21] Hanning Chen, Ali Zakeri, Fei Wen, Hamza Errahmouni Barkam, and Mohsen Imani. 2023. HyperGRAF: Hyperdimensional Graph-Based Reasoning Acceleration on FPGA. In *2023 33rd International Conference on Field-Programmable Logic and Applications (FPL)*. IEEE, 34–41.

[22] Weiwei Chen, Ying Wang, Ying Xu, Chengsi Gao, Cheng Liu, and Lei Zhang. 2022. A framework for neural network architecture and compile co-optimization. *ACM Transactions on Embedded Computing Systems* 22, 1 (2022), 1–24.

[23] C-Y Cher et al. 2015. Active memory cube: A processing-in-memory architecture for exascale systems. *IBM Journal of Research and Development* 59, 2/3 (2015), 17–1.

[24] Yu-Der Chih, Yi-Chun Shih, Chia-Fu Lee, Yen-An Chang, Po-Hao Lee, Hon-Jarn Lin, Yu-Lin Chen, Chieh-Pu Lo, Meng-Chun Shih, Kuei-Hung Shen, et al. 2020. 13.3 a 22nm 32mb embedded stt-mram with 10ns read speed, 1m cycle write endurance, 10 years retention at 150 c and high immunity to magnetic field interference. In *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 222–224.

[25] Yen-Cheng Chiu, Win-San Khwa, Chung-Yuan Li, Fang-Ling Hsieh, Yu-An Chien, Guan-Yi Lin, Po-Jung Chen, Tsen-Hsiang Pan, De-Qi You, Fang-Yi Chen, et al. 2023. A 22nm 8Mb STT-MRAM near-memory-computing macro with 8b-precision and 46.4-160.1 TOPS/W for edge-AI devices. In *2023 IEEE International Solid-State Circuits Conference (ISSCC)*. IEEE, 496–498.

[26] Jeongdong Choe. 2023. Recent technology insights on STT-MRAM: Structure, materials, and process integration. In *2023 IEEE International Memory Workshop (IMW)*. IEEE, 1–4.

[27] Woong Choi, Gyuseong Kang, and Jongsun Park. 2015. A refresh-less eDRAM macro with embedded voltage reference and selective read for an area and power efficient Viterbi decoder. *IEEE Journal of Solid-State Circuits* 50, 10 (2015), 2451–2462.

[28] Mark M Churchland, John P Cunningham, Matthew T Kaufman, Justin D Foster, Paul Nuyujukian, Stephen I Ryu, and Krishna V Shenoy. 2012. Neural population dynamics during reaching. *Nature* 487, 7405 (2012), 51–56.

[29] Brian Crafton, Zishen Wan, Samuel Spetalnick, Jong-Hyeok Yoon, Wei Wu, Carlos Tokunaga, Vivek De, and Arijit Raychowdhury. 2022. Improving compute in-memory ECC reliability with successive correction. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*. 745–750.

[30] Guohao Dai, Tianhao Huang, Yu Wang, Huazhong Yang, and John Wawrzynek. 2019. GraphSAR: A sparsity-aware processing-in-memory architecture for large-scale graph processing on ReRAMs. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference*. 120–126.

[31] Xiaoliang Dai, Peizhao Zhang, Bichen Wu, Hongxu Yin, Fei Sun, Yanghan Wang, Marat Dukhan, Yunqing Hu, Yiming Wu, Yangqing Jia, et al. 2019. Chamnet: Towards efficient network design through platform-aware model adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11398–11407.

[32] Sohum Datta, Ryan AG Antonio, Aldrin RS Ison, and Jan M Rabaey. 2019. A programmable hyper-dimensional processor architecture for human-centric IoT. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9, 3 (2019), 439–452.

[33] Sohum Datta, Brian Richards, Harrison Liew, Youbin Kim, Daniel Sun, and Jan M Rabaey. 2023. HDBinaryCore: A 28nm 2048-bit Hyper-Dimensional biosignal classifier achieving 25 nJ/prediction for EMG hand-gesture recognition. In *Proceedings of the European Solid State Circuits Conference (ESSCIRC)*. 229–232.

[34] Shuting Du, Luqi Zheng, Aradhana Mohan Parvathy, Feifan Xie, Tiwei Wei, Anand Raghunathan, and Haitong Li. 2025. 3D-CIMlet: A Chiplet Co-Design Framework for Heterogeneous In-Memory Acceleration of Edge LLM Inference and Continual Learning. In *2025 62nd ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–7.

[35] Nicole Sandra-Yaffa Dumont, P Michael Furlong, Jeff Orchard, and Chris Eliasmith. 2023. Exploiting semantic information in a spiking neural SLAM system. *Frontiers in Neuroscience* 17 (2023).

[36] Arpan Dutta, Saransh Gupta, Behnam Khaleghi, Rishikanth Chandrasekaran, Weihong Xu, and Tajana Rosing. 2022. Hdnn-pim: Efficient in memory design of hyperdimensional computing with feature extraction. In *Proceedings of the Great Lakes Symposium on VLSI 2022*. 281–286.

[37] Manuel Eggimann, Abbas Rahimi, and Luca Benini. 2021. A 5 μW standard cell memory-based configurable hyperdimensional computing accelerator for always-on smart sensing. *IEEE Transactions on Circuits and Systems I: Regular Papers* 68, 10 (2021), 4116–4128.

[38] Chris Eliasmith, Terrence C Stewart, Xuan Choo, Trevor Bekolay, Travis DeWolf, Yichuan Tang, and Daniel Rasmussen. 2012. A large-scale model of the functioning brain. *Science* 338, 6111 (2012), 1202–1205.

[39] Blerim Emruli, Ross W Gayler, and Fredrik Sandin. 2013. Analogical mapping and inference with binary spatter codes and sparse distributed memory. In *International Joint Conference on Neural Networks (IJCNN)*. 1–8.

[40] Amin Farmahini-Farahani, Jung Ho Ahn, Katherine Morrow, and Nam Sung Kim. 2015. NDA: Near-DRAM acceleration architecture leveraging commodity DRAM devices and standard memory modules. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. 283–295. doi:10.1109/HPCA.2015.7056040

[41] Farbin Fayza, Cansu Demirkiran, Hanning Chen, Che-Kai Liu, Avi Mohan, Hamza Errahmouni, Sanggeon Yun, Mohsen Imani, David Zhang, Darius Bunandar, et al. 2023. Towards Efficient Hyperdimensional Computing Using Photonics. *arXiv preprint arXiv:2311.17801* (2023).
[42] E Paxon Frady, Spencer J Kent, Bruno A Olshausen, and Friedrich T Sommer. 2020. Resonator networks, 1: An efficient solution for factoring high-dimensional, distributed representations of data structures. *Neural Computation* 32, 12 (2020), 2311–2331.
[43] Gene A Frantz and Richard H Wiggins. 1982. Design case history: Speak & Spell learns to talk. *IEEE Spectrum* 19, 2 (1982), 45–49.
[44] Charlotte Frenkel, David Bol, and Giacomo Indiveri. 2023. Bottom-Up and Top-Down Approaches for the Design of Neuromorphic Processing Systems: Tradeoffs and Synergies Between Natural and Artificial Intelligence. *Proc. IEEE* (2023).
[45] P Michael Furlong and Chris Eliasmith. 2023. Bridging Cognitive Architectures and Generative Models with Vector Symbolic Algebras. In *Proceedings of the AAAI Symposium Series*, Vol. 2. 262–271.
[46] Stephen I Gallant and T Wendy Okaywe. 2013. Representing Objects, Relations, and Sequences. *Neural Computation* 25, 8 (2013), 2038–2078.
[47] D Garcia-Lesta, F Pardo, O Pereira-Rial, VM Brea, and P Lopez. 2022. HDC8192: A General Purpose Mixed-Signal CMOS Architecture for Massively Parallel Hyperdimensional Computing. In *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 496–500.
[48] Ross W Gayler and Simon D Levy. 2009. A distributed basis for analogical mapping. In *New Frontiers in Analogy Research: Proc. of 2nd Intern. Analogy Conf*, Vol. 9.
[49] Lulu Ge and Keshab K Parhi. 2022. Applicability of hyperdimensional computing to seizure detection. *IEEE Open Journal of Circuits and Systems* 3 (2022), 59–71.
[50] Lulu Ge and Keshab K Parhi. 2024. Robust clustering using hyperdimensional computing. *IEEE Open Journal of Circuits and Systems* 5 (2024), 102–116.
[51] Christina Giannoula, Ivan Fernandez, Juan Gómez Luna, Nectarios Koziris, Georgios Goumas, and Onur Mutlu. 2022. Sparsep: Towards efficient sparse matrix vector multiplication on real processing-in-memory architectures. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 6, 1 (2022), 1–49.
[52] Massimo Giordano, Kartik Prabhu, Kalhan Koul, Robert M Radway, Albert Gural, Rohan Doshi, Zainab F Khan, John W Kustin, Timothy Liu, Gregorio B Lopes, et al. 2021. CHIMERA: A 0.92 TOPS, 2.2 TOPS/W edge AI accelerator with 2 MByte on-chip foundry resistive RAM for efficient training and inference. In *2021 symposium on VLSI circuits*. IEEE, 1–2.
[53] Oleg Golonzka, U Arslan, P Bai, M Bohr, O Baykan, Y Chang, A Chaudhari, A Chen, J Clarke, C Connor, et al. 2019. Non-volatile RRAM embedded into 22FFL FinFET technology. In *2019 Symposium on VLSI Technology*. IEEE, T230–T231.
[54] Bhargava Gopireddy and Josep Torrellas. 2019. Designing vertical processors in monolithic 3D. In *Proceedings of the 46th International Symposium on Computer Architecture*. 643–656.
[55] Robert Guirado, Abbas Rahimi, Geethan Karunaratne, Eduard Alarcón, Abu Sebastian, and Sergi Abadal. 2022. Wireless on-chip communications for scalable in-memory hyperdimensional computing. In *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.
[56] Robert Guirado, Abbas Rahimi, Geethan Karunaratne, Eduard Alarcón, Abu Sebastian, and Sergi Abadal. 2023. WHYPE: A Scale-Out Architecture With Wireless Over-the-Air Majority for Scalable In-Memory Hyperdimensional Computing. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 13, 1 (2023), 137–149.
[57] Saransh Gupta, Mohsen Imani, and Tajana Rosing. 2018. Felix: Fast and energy-efficient logic in memory. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 1–7.
[58] Zhezhi He, Shaahin Angizi, and Deliang Fan. 2017. Exploring STT-MRAM based in-memory computing paradigm with application of image edge extraction. In *2017 IEEE International Conference on Computer Design (ICCD)*. IEEE, 439–446.
[59] Mike Heddes, Igor Nunes, Tony Givargis, Alexandru Nicolau, and Alex Veidenbaum. 2024. Hyperdimensional computing: a framework for stochastic computation and symbolic AI. *Journal of Big Data* 11, 1 (2024), 145.
[60] Mike Heddes, Igor Nunes, Pere Vergés, Denis Kleyko, Danny Abraham, Tony Givargis, Alexandru Nicolau, and Alexander Veidenbaum. 2023. Torchhd: An open source Python library to support research on hyperdimensional computing and vector symbolic architectures. *Journal of Machine Learning Research* 24, 255 (2023), 1–10.
[61] Alejandro Hernández-Cano, Rosario Cammarota, and Mohsen Imani. 2021. PRID: Model Inversion Privacy Attacks in Hyperdimensional Learning Systems. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 553–558.
[62] Alejandro Hernández-Cano, Yeseong Kim, and Mohsen Imani. 2021. A framework for efficient and binary clustering in high-dimensional space. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1859–1864.
[63] Alejandro Hernández-Cano, Cheng Zhuo, Xunzhao Yin, and Mohsen Imani. 2021. Reghd: Robust and efficient regression in hyperdimensional learning system. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 7–12.
[64] Michael Hersche, Francesco di Stefano, Thomas Hofmann, Abu Sebastian, and Abbas Rahimi. 2023. Probabilistic Abduction for Visual Abstract Reasoning via Learning Rules in Vector-symbolic Architectures. In *The 3rd Workshop on Mathematical Reasoning and AI at NeurIPS'23*.

[65] Michael Hersche, Francesco Di Stefano, Thomas Hofmann, Abu Sebastian, and Abbas Rahimi. 2024. Probabilistic abduction for visual abstract reasoning via learning rules in vector-symbolic architectures. *arXiv preprint arXiv:2401.16024* (2024).

[66] Michael Hersche, Edoardo Mello Rella, Alfio Di Mauro, Luca Benini, and Abbas Rahimi. 2020. Integrating event-based dynamic vision sensors with sparse hyperdimensional computing: A low-power accelerator with online learning capability. In *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*. 169–174.

[67] Michael Hersche, Mustafa Zeqiri, Luca Benini, Abu Sebastian, and Abbas Rahimi. 2023. A neuro-vector-symbolic architecture for solving Raven's progressive matrices. *Nature Machine Intelligence* 5, 4 (2023), 363–375.

[68] Sara Hooker. 2021. The hardware lottery. *Commun. ACM* 64, 12 (2021), 58–65.

[69] Yu-Shun Hsiao, Zishen Wan, Tianyu Jia, Radhika Ghosal, Abdulrahman Mahmoud, Arijit Raychowdhury, David Brooks, Gu-Yeon Wei, and Vijay Janapa Reddi. 2023. Silent Data Corruption in Robot Operating System: A Case for End-to-End System-Level Fault Analysis Using Autonomous UAVs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2023).

[70] Cheng-Yen Hsieh, Yu-Chuan Chuang, and An-Yeu Andy Wu. 2021. Fl-hdc: Hyperdimensional computing design for the application of federated learning. In *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 1–5.

[71] Po-Kai Hsu and Shimeng Yu. 2022. In-memory 3d nand flash hyperdimensional computing engine for energy-efficient sars-cov-2 genome sequencing. In *2022 IEEE International Memory Workshop (IMW)*. IEEE, 1–4.

[72] Je-Min Hung, Cheng-Xin Xue, Hui-Yao Kao, Yen-Hsiang Huang, Fu-Chun Chang, Sheng-Po Huang, Ta-Wei Liu, Chuan-Jia Jhang, Chin-I Su, Win-San Khwa, et al. 2021. A four-megabit compute-in-memory macro with eight-bit precision based on CMOS and resistive random-access memory for AI edge devices. *Nature Electronics* 4, 12 (2021), 921–930.

[73] Mohamed Ibrahim, Youbin Kim, and Jan M. Rabaey. 2024. Efficient Design of a Hyperdimensional Processing Unit for Multi-Layer Cognition. In *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE.

[74] Mohamed Ibrahim, Zishen Wan, Haitong Li, Priyadarshini Panda, Tushar Krishna, Pentti Kanerva, Yiran Chen, and Arijit Raychowdhury. 2024. Special Session: Neuro-Symbolic Architecture Meets Large Language Models: A Memory-Centric Perspective. In *2024 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS)*. IEEE, 11–20.

[75] Mohsen Imani, Yeseong Kim, Sadegh Riazi, John Messerly, Patric Liu, Farinaz Koushanfar, and Tajana Rosing. 2019. A framework for collaborative learning in secure high-dimensional space. In *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*. IEEE, 435–446.

[76] Mohsen Imani, Yeseong Kim, Thomas Worley, Saransh Gupta, and Tajana Rosing. 2019. Hdcluster: An accurate clustering using brain-inspired high-dimensional computing. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1591–1594.

[77] Mohsen Imani, John Messerly, Fan Wu, Wang Pi, and Tajana Rosing. 2019. A binary learning framework for hyperdimensional computing. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 126–131.

[78] Mohsen Imani, Saikishan Pampana, Saransh Gupta, Minxuan Zhou, Yeseong Kim, and Tajana Rosing. 2020. Dual: Acceleration of clustering algorithms using digital-based processing in-memory. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 356–371.

[79] Mohsen Imani, Sahand Salamat, Saransh Gupta, Jiani Huang, and Tajana Rosing. 2019. Fach: Fpga-based acceleration of hyperdimensional computing by reducing computational complexity. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference*. 493–498.

[80] Mohsen Imani, Zhuowen Zou, Samuel Bosch, Sanjay Anantha Rao, Sahand Salamat, Venkatesh Kumar, Yeseong Kim, and Tajana Rosing. 2021. Revisiting hyperdimensional learning for fpga and low-power architectures. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. 221–234.

[81] Pulkit Jain, Umut Arslan, Meenakshi Sekhar, Blake C Lin, Liqiong Wei, Tanaya Sahu, Juan Alzate-Vinasco, Ajay Vangapaty, Mesut Meterelliyoz, Nathan Strutt, et al. 2019. 13.2 A 3.6 Mb 10.1 Mb/mm 2 embedded non-volatile ReRAM macro in 22nm FinFET technology with adaptive forming/set/reset schemes yielding down to 0.5 V with sensing time of 5ns at 0.7 V. In *2019 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 212–214.

[82] Hyeon-Ae Jeon. 2014. Hierarchical processing in the prefrontal cortex in a variety of cognitive domains. *Frontiers in Systems Neuroscience* 8 (2014), 223.

[83] Weiwen Jiang, Qiuwen Lou, Zheyu Yan, Lei Yang, Jingtong Hu, Xiaobo Sharon Hu, and Yiyu Shi. 2020. Device-circuit-architecture co-exploration for computing-in-memory neural accelerators. *IEEE Trans. Comput.* 70, 4 (2020), 595–605.

[84] Weiwen Jiang, Lei Yang, Edwin Hsing-Mean Sha, Qingfeng Zhuge, Shouzhen Gu, Sakyasingha Dasgupta, Yiyu Shi, and Jingtong Hu. 2020. Hardware/software co-exploration of neural architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 12 (2020), 4805–4815.

[85] Aditya Joshi, Johan T Halseth, and Pentti Kanerva. 2017. Language Geometry Using Random Indexing. In *Proceedings of International Conference on Quantum Interaction*. 265–274.

[86] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. 2017. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the International*

Symposium on Computer Architecture (ISCA). 1–12.
[87] Indhumathi Kandaswamy, Saurabh Farkya, Zachary Daniels, Gooitzen van der Wal, Aswin Raghavan, Yuzheng Zhang, Jun Hu, Michael Lomnitz, Michael Isnardi, David Zhang, et al. 2022. Real-time Hyper-Dimensional Reconfiguration at the Edge using Hardware Accelerators. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3610–3618.
[88] Pentti Kanerva. 1988. *Sparse distributed memory*. MIT press.
[89] Pentti Kanerva. 2009. Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors. *Cognitive Computation* 1 (2009), 139–159.
[90] Pentti Kanerva. 2010. What we mean when we say "What's the dollar of Mexico?": Prototypes and mapping in concept space. In *2010 AAAI fall symposium series*.
[91] Jaeyoung Kang, Behnam Khaleghi, Tajana Rosing, and Yeseong Kim. 2022. OpenHD: A GPU-powered framework for hyperdimensional computing. *IEEE Trans. Comput.* 71, 11 (2022), 2753–2765.
[92] Jaeyoung Kang, Minxuan Zhou, Abhinav Bhansali, Weihong Xu, Anthony Thomas, and Tajana Rosing. 2022. RelHD: A Graph-based Learning on FeFET with Hyperdimensional Computing. In *2022 IEEE 40th International Conference on Computer Design (ICCD)*. IEEE, 553–560.
[93] Geethan Karunaratne, Manuel Le Gallo, Giovanni Cherubini, Luca Benini, Abbas Rahimi, and Abu Sebastian. 2020. In-memory hyperdimensional computing. *Nature Electronics* 3, 6 (2020), 327–337.
[94] Geethan Karunaratne, Manuel Le Gallo, Michael Hersche, Giovanni Cherubini, Luca Benini, Abu Sebastian, and Abbas Rahimi. 2021. Energy Efficient In-Memory Hyperdimensional Encoding for Spatio-Temporal Signal Processing. *IEEE Transactions on Circuits and Systems II: Express Briefs (TCAS-II)* 68, 5 (2021), 1725–1729.
[95] Geethan Karunaratne, Manuel Schmuck, Manuel Le Gallo, Giovanni Cherubini, Luca Benini, Abu Sebastian, and Abbas Rahimi. 2021. Robust high-dimensional memory-augmented neural networks. *Nature communications* 12, 1 (2021), 2468.
[96] Arman Kazemi, Franz Müller, Mohammad Mehdi Sharifi, Hamza Errahmouni, Gerald Gerlach, Thomas Kämpfe, Mohsen Imani, Xiaobo Sharon Hu, and Michael Niemier. 2022. Achieving software-equivalent accuracy for hyperdimensional computing with ferroelectric-based in-memory computing. *Scientific reports* 12, 1 (2022), 19201.
[97] Behnam Khaleghi, Mohsen Imani, and Tajana Rosing. 2020. Prive-hd: Privacy-preserved hyperdimensional computing. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.
[98] Behnam Khaleghi, Hanyang Xu, Justin Morris, and Tajana Šimunić Rosing. 2021. tiny-HD: Ultra-efficient hyperdimensional computing engine for IoT applications. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 408–413.
[99] Behnam Khaleghi, Xiaofan Yu, Jaeyoung Kang, Xuan Wang, and Tajana Rosing. 2024. Private and efficient learning with hyperdimensional computing. *IEEE Transactions on Circuits and Systems for Artificial Intelligence* 1, 2 (2024), 204–219.
[100] Yeseong Kim, Mohsen Imani, Niema Moshiri, and Tajana Rosing. 2020. Geniehd: Efficient dna pattern matching accelerator using hyperdimensional computing. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 115–120.
[101] Daniil Kirilenko, Alexey K Kovalev, Yaroslav Solomentsev, Alexander Melekhin, Dmitry A Yudin, and Aleksandr I Panov. 2022. Vector symbolic scene representation for semantic place recognition. In *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.
[102] Denis Kleyko, Connor Bybee, Ping-Chen Huang, Christopher J Kymn, Bruno A Olshausen, E Paxon Frady, and Friedrich T Sommer. 2023. Efficient Decoding of Compositional Structure in Holistic Representations. *Neural Computation* 35, 7 (2023), 1159–1186.
[103] Denis Kleyko, Mike Davies, Edward Paxon Frady, Pentti Kanerva, Spencer J Kent, Bruno A Olshausen, Evgeny Osipov, Jan M Rabaey, Dmitri A Rachkovskij, Abbas Rahimi, et al. 2022. Vector symbolic architectures as a computing framework for emerging hardware. *Proc. IEEE* 110, 10 (2022), 1538–1571.
[104] Denis Kleyko, Dmitri Rachkovskij, Evgeny Osipov, and Abbas Rahimi. 2023. A survey on hyperdimensional computing aka vector symbolic architectures, part ii: Applications, cognitive models, and challenges. *Comput. Surveys* 55, 9 (2023), 1–52.
[105] Denis Kleyko, Dmitri A Rachkovskij, Evgeny Osipov, and Abbas Rahimi. 2022. A survey on hyperdimensional computing aka vector symbolic architectures, part i: Models and data transformations. *Comput. Surveys* 55, 6 (2022), 1–40.
[106] Denis Kleyko, Abbas Rahimi, Dmitri A Rachkovskij, Evgeny Osipov, and Jan M Rabaey. 2018. Classification and recall with binary hyperdimensional computing: Tradeoffs in choice of density and mapping characteristics. *IEEE Transactions on Neural Networks and Learning Systems* 29, 12 (2018), 5880–5898.
[107] Shigenobu Komatsu, Masanao Yamaoka, Masao Morimoto, Noriaki Maeda, Yasuhisa Shimazaki, and Kenichi Osada. 2009. A 40-nm low-power SRAM with multi-stage replica-bitline technique for reducing timing variation. In *2009 IEEE Custom Integrated Circuits Conference*. 701–704. doi:10.1109/CICC.2009.5280731
[108] Adam Kortylewski, Aleksander Wieczorek, Mario Wieser, Clemens Blumer, Sonali Parbhoo, Andreas Morel-Forster, Volker Roth, and Thomas Vetter. 2019. Greedy structure learning of hierarchical compositional models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11612–11621.
[109] Srivatsan Krishnan, Zishen Wan, Kshitij Bhardwaj, Ninad Jadhav, Aleksandra Faust, and Vijay Janapa Reddi. 2022. Roofline model for uavs: A bottleneck analysis tool for onboard compute characterization of autonomous unmanned aerial vehicles. In *2022 IEEE*

International Symposium on Performance Analysis of Systems and Software (ISPASS). IEEE, 162–174.
[110] HT Kung. 1982. Why Systolic Architectures? *Computer* 15, 01 (1982), 37–46.
[111] Jovin Langenegger, Geethan Karunaratne, Michael Hersche, Luca Benini, Abu Sebastian, and Abbas Rahimi. 2023. In-memory factorization of holographic perceptual representations. *Nature Nanotechnology* 18, 5 (2023), 479–485.
[112] Chieh Lee, Yue-Der Chih, Jonathan Chang, Chrong Jung Lin, and Ya-Chin King. 2020. Memory-logic hybrid gate with 3-D stackable complementary latches. *IEEE Transactions on Electron Devices* 67, 8 (2020), 3109–3114.
[113] Haitong Li, Wei-Chen Chen, Akash Levy, Ching-Hua Wang, Hongjie Wang, Po-Han Chen, Weier Wan, Win-San Khwa, Harry Chuang, Y-D Chih, et al. 2021. SAPIENS: A 64-kb RRAM-based non-volatile associative memory for one-shot learning and inference at the edge. *IEEE Transactions on Electron Devices* 68, 12 (2021), 6637–6643.
[114] Haitong Li, Tony F Wu, Abbas Rahimi, Kai-Shin Li, Miles Rusch, Chang-Hsien Lin, Juo-Luen Hsu, Mohamed M Sabry, S Burc Eryilmaz, Joon Sohn, et al. 2016. Hyperdimensional computing with 3D VRRAM in-memory kernels: Device-architecture co-design for energy-efficient, error-resilient language recognition. In *2016 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 16–1.
[115] Harrison Liew, Daniel Grubb, John Wright, Colin Schmidt, Nayiri Krzysztofowicz, Adam Izraelevitz, Edward Wang, Krste Asanović, Jonathan Bachrach, and Borivoje Nikolić. 2022. Hammer: a modular and reusable physical design flow tool. In *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*. 1335–1338.
[116] Chiao Liu, Song Chen, Tsung-Hsun Tsai, Barbara De Salvo, and Jorge Gomez. 2022. Augmented Reality-The Next Frontier of Image Sensors and Compute Systems. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, Vol. 65. IEEE, 426–428.
[117] Che-Kai Liu, Haobang Chen, Mohsen Imani, Kai Ni, Arman Kazemi, Ann Franchesca Laguna, Michael Niemier, Xiaobo Sharon Hu, Liang Zhao, Cheng Zhuo, et al. 2022. Cosime: FeFET Based Associative Memory for In-Memory Cosine Similarity Search. In *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1–9.
[118] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055* (2018).
[119] Qiang Liu, Zishen Wan, Bo Yu, Weizhuang Liu, Shaoshan Liu, and Arijit Raychowdhury. 2022. An energy-efficient and runtime-reconfigurable fpga-based accelerator for robotic localization systems. In *2022 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE, 01–02.
[120] Shuhan Liu, Robert M. Radway, Xinxin Wang, Filippo Moro, Jean-Francois Nodin, Koustav Jana, Shuting Du, Luke R. Upton, Wei-Chen Chen, Jian Chen, Haitong Li, Francois Andrieu, Elisa Vianello, Priyanka Raina, Subhasish Mitra, and H.-S. Philip Wong. 2024. Edge Continual Training and Inference with RRAM-Gain Cell Memory Integrated on Si CMOS. In *2024 IEEE International Electron Devices Meeting (IEDM)*. 1–4. doi:10.1109/IEDM50854.2024.10873546
[121] Shuhan Liu, Robert M. Radway, Xinxin Wang, Filippo Moro, Jean-Francois Nodin, Koustav Jana, Lixian Yan, Shuting Du, Luke R. Upton, Wei-Chen Chen, Jimin Kang, Jian Chen, Haitong Li, Francois Andrieu, Elisa Vianello, Priyanka Raina, Subhasish Mitra, and H.-S. Philip Wong. 2025. Monolithic 3-D Integration of Diverse Memories: Resistive Switching (RRAM) and Gain Cell (GC) Memory Integrated on Si CMOS. *IEEE Transactions on Electron Devices* 72, 5 (2025), 2685–2690. doi:10.1109/TED.2025.3556113
[122] Shaoshan Liu, Zishen Wan, Bo Yu, and Yu Wang. 2021. *Robotic computing on fpgas*. Springer.
[123] Yu Lu, Tom Zhong, W Hsu, S Kim, X Lu, JJ Kan, C Park, WC Chen, X Li, X Zhu, et al. 2015. Fully functional perpendicular STT-MRAM macro embedded in 40 nm logic for energy-efficient IOT applications. In *2015 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 26–1.
[124] Yunpu Ma, Marcel Hildebrandt, Volker Tresp, and Stephan Baier. 2018. Holistic Representations for Memorization and Inference.. In *UAI*. 403–413.
[125] Víctor Mayoral-Vilches, Jason Jabbour, Yu-Shun Hsiao, Zishen Wan, Alejandra Martínez-Fariña, Martino Crespo-Alvarez, Matthew Stewart, Juan Manuel Reina-Munoz, Prateek Nagras, Gaurav Vikhe, et al. 2023. RobotPerf: An Open-Source, Vendor-Agnostic, Benchmarking Suite for Evaluating Robotics Computing System Performance. *arXiv preprint arXiv:2309.09212* (2023).
[126] Nathan McDonald. 2023. Modularizing and Assembling Cognitive Map Learners via Hyperdimensional Computing. *arXiv preprint arXiv:2304.04734* (2023).
[127] Nicolas Menet, Michael Hersche, Geethan Karunaratne, Luca Benini, Abu Sebastian, and Abbas Rahimi. 2023. Mimonets: Multiple-input-multiple-output neural networks exploiting computation in superposition. *Advances in Neural Information Processing Systems* 36 (2023), 39553–39565.
[128] Alisha Menon, Anirudh Natarajan, Laura I. Galindez Olascoaga, Youbin Kim, Braeden Benedict, and Jan M. Rabaey. 2022. On the Role of Hyperdimensional Computing for Behavioral Prioritization in Reactive Robot Navigation Tasks. In *2022 International Conference on Robotics and Automation (ICRA)*. 7335–7341. doi:10.1109/ICRA46639.2022.9811939
[129] Alisha Menon, Laura I Galindez Olascoaga, Vamshi Balanaga, Anirudh Natarajan, Jennifer Ruffing, Ryan Ardalan, and Jan M Rabaey. 2023. Shared Control of Assistive Robots through User-intent Prediction and Hyperdimensional Recall of Reactive Behavior. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 12638–12644.
[130] Alisha Menon, Laura I Galindez Olascoaga, Niki Shakouri, Jennifer Ruffing, Vamshi Balanaga, and Jan M Rabaey. 2022. Brain-inspired multi-level control of an assistive prosthetic hand through EMG task recognition. In *IEEE Biomedical Circuits and Systems Conference (BioCAS)*. IEEE, 384–388.

[131] Alisha Menon, Meek Simbule, Harrison Liew, Adriel Tan, Daniel Sun, and Jan M Rabaey. 2023. Accelerating Hyperdimensional Computing with Vector Machines. In *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1–5.

[132] Alisha Menon, Daniel Sun, Melvin Aristio, Harrison Liew, Kyoungtae Lee, and Jan M Rabaey. 2021. A highly energy-efficient hyperdimensional computing processor for wearable multi-modal classification. In *2021 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. IEEE, 1–4.

[133] Alisha Menon, Daniel Sun, Sarina Sabouri, Kyoungtae Lee, Melvin Aristio, Harrison Liew, and Jan M Rabaey. 2022. A highly energy-efficient hyperdimensional computing processor for biosignal classification. *IEEE Transactions on Biomedical Circuits and Systems (TBioCAS)* 16, 4 (2022), 524–534.

[134] Anton Mitrokhin, P Sutor, Cornelia Fermüller, and Yiannis Aloimonos. 2019. Learning sensorimotor control with neuromorphic sensors: Toward hyperdimensional active perception. *Science Robotics* 4, 30 (2019), eaaw6736.

[135] Fabio Montagna, Abbas Rahimi, Simone Benatti, Davide Rossi, and Luca Benini. 2018. PULP-HD: Accelerating brain-inspired high-dimensional computing on a parallel ultra-low power platform. In *Proceedings of the 55th Annual Design Automation Conference*. 1–6.

[136] Justin Morris, Kazim Ergun, Behnam Khaleghi, Mohsen Imani, Baris Aksanli, and Tajana Rosing. 2021. Hydrea: Towards more robust and efficient machine learning systems with hyperdimensional computing. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 723–728.

[137] Justin Morris, Yilun Hao, Saransh Gupta, Behnam Khaleghi, Baris Aksanli, and Tajana Rosing. 2022. Stochastic-HD: Leveraging Stochastic Computing on the Hyper-Dimensional Computing Pipeline. *Frontiers in Neuroscience* 16 (2022), 867192.

[138] Junjie Mu, Lu Lu, Ju Eon Kim, Byungkwon An, Vishal Sharma, Arya Jagath Lekshmi, Putu Andhita Dananjaya, Weng Hong Lai, Wen Siang Lew, and Tony Tae-Hyoung Kim. 2024. A 1Mb RRAM Macro With 9.8 ns Read Access Time Utilizing Dynamic Reference Voltage for Reliable Sensing Operation. *IEEE Transactions on Circuits and Systems II: Express Briefs* (2024).

[139] VB Naik, K Yamane, TY Lee, J Kwon, R Chao, JH Lim, NL Chung, B Behin-Aein, LY Hau, D Zeng, et al. 2020. JEDEC-qualified highly reliable 22nm FD-SOI embedded MRAM for low-power industrial-grade, and extended performance towards automotive-grade-1 applications. In *2020 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 11–3.

[140] Peer Neubert and Stefan Schubert. 2021. Hyperdimensional Computing as a Framework for Systematic Aggregation of Image Descriptors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 16938–16947.

[141] Peer Neubert, Stefan Schubert, and Peter Protzel. 2019. An introduction to hyperdimensional computing for robotics. *KI-Künstliche Intelligenz* 33 (2019), 319–330.

[142] Yang Ni, Mariam Issa, Danny Abraham, Mahdi Imani, Xunzhao Yin, and Mohsen Imani. 2022. Hdpg: hyperdimensional policy-based reinforcement learning for continuous control. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*. 1141–1146.

[143] Yang Ni, Nicholas Lesica, Fan-Gang Zeng, and Mohsen Imani. 2022. Neurally-Inspired Hyperdimensional Classification for Efficient and Robust Biosignal Processing. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*. 1–9.

[144] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 30.

[145] M Oka, Y Namba, Y Sato, H Uchida, T Doi, T Tatsuno, M Nakazawa, A Tamura, R Haga, M Kuroda, et al. 2021. 3D stacked CIS compatible 40nm embedded STT-MRAM for buffer memory. In *2021 Symposium on VLSI Technology*. IEEE, 1–2.

[146] Laura Isabel Galindez Olascoaga, Alisha Menon, Mohamed Ibrahim, and Jan Rabaey. 2022. A brain-inspired hierarchical reasoning framework for cognition-augmented prosthetic grasping. In *Combining Learning and Reasoning: Programming Languages, Formalisms, and Representations*.

[147] Jeff Orchard and Russell Jarvis. 2023. Hyperdimensional Computing with Spiking-Phasor Neurons. In *Proceedings of the International Conference on Neuromorphic Systems (ICNS)*. 1–7.

[148] Evgeny Osipov, Sachin Kahawala, Dilantha Haputhanthri, Thimal Kempitiya, Daswin De Silva, Damminda Alahakoon, and Denis Kleyko. 2022. Hyperseed: Unsupervised learning with vector symbolic architectures. *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[149] Una Pale, Tomas Teijeiro, and David Atienza. 2022. Multi-centroid hyperdimensional computing approach for epileptic seizure detection. *Frontiers in Neurology* 13 (2022), 816294.

[150] Xiaochen Peng, Shanshi Huang, Yandong Luo, Xiaoyu Sun, and Shimeng Yu. 2019. DNN+ NeuroSim: An end-to-end benchmarking framework for compute-in-memory accelerators with versatile device technologies. In *2019 IEEE international electron devices meeting (IEDM)*. IEEE, 32–5.

[151] Tony A Plate. 1994. *Distributed Representations and Nested Compositional Structure*. Ph. D. Dissertation. University of Toronto.

[152] Prathyush Poduval, Haleh Alimohamadi, Ali Zakeri, Farhad Imani, M Hassan Najafi, Tony Givargis, and Mohsen Imani. 2022. Graphd: Graph-based hyperdimensional memorization for brain-like cognitive learning. *Frontiers in Neuroscience* 16 (2022), 757125.

[153] Prathyush Poduval, Zhuowen Zou, Hassan Najafi, Houman Homayoun, and Mohsen Imani. 2021. Stochd: Stochastic hyperdimensional system for efficient and robust learning from raw data. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1195–1200.

[154] Dmitri A Rachkovskij and Denis Kleyko. 2022. Recursive Binding for Similarity-Preserving Hypervector Representations of Sequences. In *Proceedings of International Joint Conference on Neural Networks (IJCNN)*. 1–8.
[155] Abbas Rahimi, Sohum Datta, Denis Kleyko, Edward Paxon Frady, Bruno Olshausen, Pentti Kanerva, and Jan M Rabaey. 2017. High-dimensional computing as a nanoscalable paradigm. *IEEE Transactions on Circuits and Systems I: Regular Papers* 64, 9 (2017), 2508–2521.
[156] Abbas Rahimi, Pentti Kanerva, and Jan M Rabaey. 2016. A robust and energy-efficient classifier using brain-inspired hyperdimensional computing. In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED)*. 64–69.
[157] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. 2017. Large-scale evolution of image classifiers. In *International conference on machine learning*. PMLR, 2902–2911.
[158] Vijay Janapa Reddi, Christine Cheng, David Kanter, Peter Mattson, Guenther Schmuelling, Carole-Jean Wu, Brian Anderson, Maximilien Breughe, Mark Charlebois, William Chou, et al. 2020. MLPerf inference benchmark. In *Proceedings of ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. 446–459.
[159] Alpha Renner, Yulia Sandamirskaya, Friedrich Sommer, and E Paxon Frady. 2022. Sparse vector binding on spiking neuromorphic hardware using synaptic delays. In *Proceedings of the International Conference on Neuromorphic Systems (ICNS)*. 1–5.
[160] Davide Rossi, Francesco Conti, Manuel Eggiman, Alfio Di Mauro, Giuseppe Tagliavini, Stefan Mach, Marco Guermandi, Antonio Pullini, Igor Loi, Jie Chen, et al. 2021. Vega: A ten-core SoC for IoT endnodes with DNN acceleration and cognitive wake-up from MRAM-based state-retentive sleep mode. *IEEE Journal of Solid-State Circuits* 57, 1 (2021), 127–139.
[161] Walter Rudin. 2017. *Fourier analysis on groups*. Courier Dover Publications.
[162] Swapnil Sayan Saha, Sandeep Singh Sandha, Mohit Aggarwal, Brian Wang, Liying Han, Julian de Gortari Briseno, and Mani Srivastava. 2023. TinyNS: Platform-Aware Neurosymbolic Auto Tiny Machine Learning. *ACM Transactions on Embedded Computing Systems* (2023).
[163] Sahand Salamat, Mohsen Imani, Behnam Khaleghi, and Tajana Rosing. 2019. F5-hd: Fast flexible fpga-based framework for refreshing hyperdimensional computing. In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. 53–62.
[164] Sahand Salamat, Mohsen Imani, and Tajana Rosing. 2020. Accelerating hyperdimensional computing on FPGAs by exploiting computational reuse. *IEEE Trans. Comput.* 69, 8 (2020), 1159–1171.
[165] Sandeep Kumar Samal, Deepak Nayak, Motoi Ichihashi, Srinivasa Banna, and Sung Kyu Lim. 2016. Monolithic 3D IC vs. TSV-based 3D IC in 14nm FinFET technology. In *2016 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*. IEEE, 1–2.
[166] Kaspar A Schindler and Abbas Rahimi. 2021. A primer on hyperdimensional computing for iEEG seizure detection. *Frontiers in neurology* 12 (2021), 701791.
[167] Kenny Schlegel, Peer Neubert, and Peter Protzel. 2022. A Comparison of Vector Symbolic Architectures. *Artificial Intelligence Review* 55, 6 (2022), 4523–4555.
[168] Manuel Schmuck, Luca Benini, and Abbas Rahimi. 2019. Hardware optimizations of dense binary hyperdimensional computing: Rematerialization of hypervectors, binarized bundling, and combinational associative memory. *ACM Journal on Emerging Technologies in Computing Systems (JETC)* 15, 4 (2019), 1–25.
[169] Jonathon W Sensinger and Strahinja Dosen. 2020. A review of sensory feedback in upper-limb prostheses from the perspective of human motor control. *Frontiers in Neuroscience* 14 (2020), 345.
[170] Takahiro Shimoi, Ken Matsubara, Tomoya Saito, Tomoya Ogawa, Yasuhiko Taito, Yoshinobu Kaneda, Masayuki Izuna, Koichi Takeda, Hidenori Mitani, Takashi Ito, et al. 2023. A 22-nm 32-Mb Embedded STT-MRAM Macro Achieving 5.9-ns Random Read Access and 7.4-MB/s Write Throughput at up to 150° C. *IEEE Journal of Solid-State Circuits* 59, 4 (2023), 1283–1292.
[171] Shengxi Shou, Che-Kai Liu, Sanggeon Yun, Zishen Wan, Kai Ni, Mohsen Imani, X Sharon Hu, Jianyi Yang, Cheng Zhuo, and Xunzhao Yin. 2023. See-mcam: Scalable multi-bit fefet content addressable memories for energy efficient associative search. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE, 1–9.
[172] Max M Shulaker, Gage Hills, Rebecca S Park, Roger T Howe, Krishna Saraswat, H-S Philip Wong, and Subhasish Mitra. 2017. Three-dimensional integration of nanotechnologies for computing and data storage on a single chip. *Nature* 547, 7661 (2017), 74–78.
[173] William Andrew Simon, Una Pale, Tomas Teijeiro, and David Atienza. 2022. HDTorch: Accelerating hyperdimensional computing with GP-GPUs for design space exploration. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. 1–8.
[174] Samuel D Spetalnick, Ashwin Sanjay Lele, Brian Crafton, Muya Chang, Sigang Ryu, Jong-Hyeok Yoon, Zhijian Hao, Azadeh Ansari, Win-San Khwa, Yu-Der Chih, et al. 2024. An Edge Accelerator With 5 MB of 0.256-pJ/bit Embedded RRAM and a Localization Solver for Bristle Robot Surveillance. *IEEE Journal of Solid-State Circuits* (2024).
[175] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2023. Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research* (2023).
[176] Prakalp Srivastava, Mingu Kang, Sujan K Gonugondla, Sungmin Lim, Jungwook Choi, Vikram Adve, Nam Sung Kim, and Naresh Shanbhag. 2018. PROMISE: An end-to-end design of a programmable mixed-signal accelerator for machine-learning algorithms. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 43–56.

[177] Christoph Stöckl, Yukun Yang, and Wolfgang Maass. 2024. Local prediction-learning in high-dimensional spaces enables neural networks to plan. *Nature Communications* 15, 1 (2024), 2344.
[178] Carsen Stringer, Marius Pachitariu, Nicholas Steinmetz, Matteo Carandini, and Kenneth D Harris. 2019. High-dimensional geometry of population responses in visual cortex. *Nature* 571, 7765 (2019), 361–365.
[179] Guangyu Sun, Xiangyu Dong, Yuan Xie, Jian Li, and Yiran Chen. 2009. A novel architecture of the 3D stacked MRAM L2 cache for CMPs. In *2009 IEEE 15th International Symposium on High Performance Computer Architecture*. IEEE, 239–249.
[180] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. 2017. Efficient processing of deep neural networks: A tutorial and survey. *Proc. IEEE* 105, 12 (2017), 2295–2329.
[181] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. 2019. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2820–2828.
[182] Jeffrey L Teeters, Denis Kleyko, Pentti Kanerva, and Bruno A Olshausen. 2023. On separating long-and short-term memories in hyperdimensional computing. *Frontiers in Neuroscience* 16 (2023), 867568.
[183] Anthony Thomas, Sanjoy Dasgupta, and Tajana Rosing. 2021. A theoretical perspective on hyperdimensional computing. *Journal of Artificial Intelligence Research* 72 (2021), 215–249.
[184] Shikhar Tuli, Chia-Hao Li, Ritvik Sharma, and Niraj K Jha. 2023. CODEBench: A neural architecture and hardware accelerator co-design framework. *ACM Transactions on Embedded Computing Systems* 22, 3 (2023), 1–30.
[185] Roman Vershynin. 2018. *High-dimensional probability: An introduction with applications in data science*. Vol. 47. Cambridge University Press.
[186] Zishen Wan, Ashwin Lele, Bo Yu, Shaoshan Liu, Yu Wang, Vijay Janapa Reddi, Cong Hao, and Arijit Raychowdhury. 2022. Robotic computing on fpgas: Current progress, research challenges, and opportunities. In *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 291–295.
[187] Zishen Wan, Che-Kai Liu, Mohamed Ibrahim, Hanchen Yang, Samuel Spetalnick, Tushar Krishna, and Arijit Raychowdhury. 2024. H3DFact: Heterogeneous 3D Integrated CIM for Factorization with Holographic Perceptual Representations. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2024*. IEEE, 1–6.
[188] Zishen Wan, Che-Kai Liu, Hanchen Yang, Chaojian Li, Haoran You, Yonggan Fu, Cheng Wan, Tushar Krishna, Yingyan Lin, and Arijit Raychowdhury. 2024. Towards Cognitive AI Systems: a Survey and Prospective on Neuro-Symbolic AI. *arXiv preprint arXiv:2401.01040* (2024).
[189] Zishen Wan, Che-Kai Liu, Hanchen Yang, Ritik Raj, Chaojian Li, Haoran You, Yonggan Fu, Cheng Wan, Sixu Li, Youbin Kim, et al. 2024. Towards Efficient Neuro-Symbolic AI: From Workload Characterization to Hardware Architecture. *IEEE Transactions on Circuits and Systems for Artificial Intelligence* (2024).
[190] Zishen Wan, Karthik Swaminathan, Pin-Yu Chen, Nandhini Chandramoorthy, and Arijit Raychowdhury. 2022. Analyzing and Improving Resilience and Robustness of Autonomous Systems. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*. 1–9.
[191] Zishen Wan, Hanchen Yang, Ritik Raj, Che-Kai Liu, Ananda Samajdar, Arijit Raychowdhury, and Tushar Krishna. 2025. CogSys: Efficient and Scalable Neurosymbolic Cognition System via Algorithm-Hardware Co-Design. *arXiv preprint arXiv:2503.01162* (2025).
[192] Jingcheng Wang, Hyochan An, Qirui Zhang, Hun Seok Kim, David Blaauw, and Dennis Sylvester. 2020. A 40-nm ultra-low leakage voltage-stacked SRAM for intelligent IoT sensors. *IEEE Solid-State Circuits Letters* 4 (2020), 14–17.
[193] Ruixuan Wang, Sabrina Hassan Moon, Xiaobo Sharon Hu, Xun Jiao, and Dayane Reis. 2023. A Computing-in-Memory-based One-Class Hyperdimensional Computing Model for Outlier Detection. *arXiv preprint arXiv:2311.17852* (2023).
[194] Z Wang, X Hao, L Hu, D Jung, W Kim, Z Wei, L Wang, K Satoh, J Zhang, and Y Huai. 2021. 22 nm Embedded STT-MRAM Macro with 10 ns Switching and> 10 14 Endurance for Last Level Cache Applications. In *2021 Symposium on VLSI Technology*. IEEE, 1–2.
[195] Liqiong Wei, Juan G Alzate, Umut Arslan, Justin Brockman, Nilanjan Das, Kevin Fischer, Tahir Ghani, Oleg Golonzka, Patrick Hentges, Rawshan Jahan, et al. 2019. 13.3 A 7Mb STT-MRAM in 22FFL FinFET technology with 4ns read sensing time at 0.9 V using write-verify-write scheme and offset-cancellation sensing technique. In *2019 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 214–216.
[196] Jian Weng, Sihao Liu, Vidushi Dadu, Zhengrong Wang, Preyas Shah, and Tony Nowatzki. 2020. Dsagen: Synthesizing programmable spatial accelerators. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 268–281.
[197] Samuel Wilson, Tobias Fischer, Niko Sünderhauf, and Feras Dayoub. 2023. Hyperdimensional feature fusion for out-of-distribution detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2644–2654.
[198] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. 2019. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10734–10742.
[199] Tony F Wu, Haitong Li, Ping-Chen Huang, Abbas Rahimi, Gage Hills, Bryce Hodson, William Hwang, Jan M Rabaey, H-S Philip Wong, Max M Shulaker, et al. 2018. Hyperdimensional computing exploiting carbon nanotube FETs, resistive RAM, and their monolithic 3D

integration. *IEEE Journal of Solid-State Circuits* 53, 11 (2018), 3183–3196.
[200] Xinfeng Xie, Zheng Liang, Peng Gu, Abanti Basak, Lei Deng, Ling Liang, Xing Hu, and Yuan Xie. 2021. SpaceA: Sparse matrix vector multiplication on processing-in-memory accelerator. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 570–583.
[201] Cong Xu, Yang Zheng, Dimin Niu, Xiaochun Zhu, Seung H Kang, and Yuan Xie. 2015. Impact of write pulse and process variation on 22 nm FinFET-based STT-RAM design: A device-architecture co-optimization approach. *IEEE Transactions on Multi-Scale Computing Systems* 1, 4 (2015), 195–206.
[202] Weihong Xu, Jaeyoung Kang, Wout Bittremieux, Niema Moshiri, and Tajana Rosing. 2023. HyperSpec: Ultrafast Mass Spectra Clustering in Hyperdimensional Space. *Journal of Proteome Research* (2023).
[203] Cheng-Xin Xue, Yen-Cheng Chiu, Ta-Wei Liu, Tsung-Yuan Huang, Je-Syu Liu, Ting-Wei Chang, Hui-Yao Kao, Jing-Hong Wang, Shih-Ying Wei, Chun-Ying Lee, et al. 2021. A CMOS-integrated compute-in-memory macro based on resistive random-access memory for AI edge devices. *Nature Electronics* 4, 1 (2021), 81–90.
[204] Junhuan Yang, Venkat Kalyan Reddy Yasa, Yi Sheng, Dayane Reis, Xun Jiao, Weiwen Jiang, and Lei Yang. 2022. Hardware-aware automated architecture search for brain-inspired hyperdimensional computing. In *2022 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 352–357.
[205] Lei Yang, Weiwen Jiang, Weichen Liu, Edwin HM Sha, Yiyu Shi, and Jingtong Hu. 2020. Co-exploring neural architecture and network-on-chip design for real-time artificial intelligence. In *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 85–90.
[206] Lei Yang, Zheyu Yan, Meng Li, Hyoukjun Kwon, Liangzhen Lai, Tushar Krishna, Vikas Chandra, Weiwen Jiang, and Yiyu Shi. 2020. Co-exploration of neural architectures and heterogeneous asic accelerator designs targeting multiple tasks. In *2020 57th ACM/IEEE design automation conference (DAC)*. IEEE, 1–6.
[207] Xiaoxuan Yang, Zhangyang Wang, X Sharon Hu, Chris H Kim, Shimeng Yu, Miroslav Pajic, Rajit Manohar, Yiran Chen, and Hai Helen Li. 2023. Neuro-Symbolic Computing: Advancements and Challenges in Hardware-Software Co-Design. *IEEE Transactions on Circuits and Systems II: Express Briefs* (2023).
[208] Tianyang Yu, Bi Wu, Ke Chen, Gong Zhang, and Weiqiang Liu. 2024. Fully Learnable Hyperdimensional Computing Framework with Ultra-tiny Accelerator for Edge-side Applications. *IEEE Trans. Comput.* 73, 2 (2024), 574–585.
[209] Chi Zhang, Baoxiong Jia, Song-Chun Zhu, and Yixin Zhu. 2021. Abstract spatial-temporal reasoning via probabilistic abduction and execution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9736–9746.
[210] Qirui Zhang, Hyochan An, Zichen Fan, Zhehong Wang, Ziyun Li, Guanru Wang, Hun-Seok Kim, David Blaauw, and Dennis Sylvester. 2022. A 22nm 3.5 TOPS/W flexible micro-robotic vision SoC with 2MB eMRAM for fully-on-chip intelligence. In *2022 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*. IEEE, 72–73.
[211] Quanling Zhao, Yanru Chen, Runyang Tian, Sumukh Pinge, Weihong Xu, Augusto Vega, Steven Holmes, Saransh Gupta, and Tajana Rosing. 2025. HDDB: Efficient In-Storage SQL Database Search Using Hyperdimensional Computing on Ferroelectric NAND Flash. *arXiv preprint arXiv:2511.18234* (2025).
[212] Quanling Zhao, Kai Lee, Jeffrey Liu, Muhammad Huzaifa, Xiaofan Yu, and Tajana Rosing. 2022. Fedhd: federated learning with hyperdimensional computing. In *Proceedings of the 28th annual international conference on mobile computing and networking*. 791–793.
[213] Quanling Zhao, Xiaofan Yu, and Tajana Rosing. 2023. Attentive Multimodal Learning on Sensor Data using Hyperdimensional Computing. In *Proceedings of the 22nd International Conference on Information Processing in Sensor Networks*. 312–313.
[214] Luqi Zheng and Haitong Li. 2024. CMOS+ X Technologies for Neuro-Vector-Symbolic Computing. In *2024 Device Research Conference (DRC)*. IEEE, 1–2.
[215] Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578* (2016).
[216] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. 2018. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 8697–8710.
[217] Zhuowen Zou, Yeseong Kim, Farhad Imani, Haleh Alimohamadi, Rosario Cammarota, and Mohsen Imani. 2021. Scalable edge-based hyperdimensional learning system with brain-like neural adaptation. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–15.

Received 20 November 2025; revised 4 March 2026; accepted 30 March 2026