Variable Binding for Sparse Distributed Representations: Theory and Applications

Edward Paxon Frady, Denis Kleyko, *Member, IEEE*, and Friedrich T. Sommer

***Abstract*— Variable binding is a cornerstone of symbolic reasoning and cognition. But how binding can be implemented in connectionist models has puzzled neuroscientists, cognitive psychologists, and neural network researchers for many decades. One type of connectionist model that naturally includes a binding operation is vector symbolic architectures (VSAs). In contrast to other proposals for variable binding, the binding operation in VSAs is dimensionality-preserving, which enables representing complex hierarchical data structures, such as trees, while avoiding a combinatoric expansion of dimensionality. Classical VSAs encode symbols by dense randomized vectors, in which information is distributed throughout the entire neuron population. By contrast, in the brain, features are encoded more locally, by the activity of single neurons or small groups of neurons, often forming sparse vectors of neural activation. Following Laiho *et al.* (2015), we explore symbolic reasoning with a special case of sparse distributed representations. Using techniques from compressed sensing, we first show that variable binding in classical VSAs is mathematically equivalent to tensor product binding between sparse feature vectors, another well-known binding operation which increases dimensionality. This theoretical result motivates us to study two dimensionality-preserving binding methods that include a reduction of the tensor matrix into a single sparse vector. One binding method for general sparse vectors uses random projections, the other, block-local circular convolution, is defined for sparse vectors with block structure, sparse block-codes. Our experiments reveal that block-local circular convolution binding has ideal properties, whereas random projection based binding also works, but is lossy. We demonstrate in example applications that a VSA with block-local circular convolution and sparse block-codes reaches similar performance as classical VSAs. Finally, we discuss our results in the context of neuroscience and neural networks.**

***Index Terms*— Classification, cognitive reasoning, compressed sensing (CS), sparse block-codes, sparse distributed representations, tensor product variable binding, vector symbolic architectures (VSAs).**

Manuscript received September 22, 2020; revised March 9, 2021 and June 10, 2021; accepted August 14, 2021. The work of Denis Kleyko was supported in part by the European Union's Horizon 2020 Research and Innovation Program under the Marie Skłodowska-Curie Individual Fellowship Grant Agreement 839179 and in part by the DARPA's Virtual Intelligence Processing (VIP) (Super-HD Project) and Artificial Intelligence Exploration (AIE) (HyDDENN Project) programs. The work of Friedrich T. Sommer was supported by NIH R01-EB026955. *(Corresponding author: Friedrich T. Sommer.)*

Edward Paxon Frady and Friedrich T. Sommer are with Neuromorphic Computing Lab, Intel Labs, Santa Clara, CA 95054 USA, and also with the Redwood Center for Theoretical Neuroscience, University of California, Berkeley, CA 94720 USA (e-mail: fsommer@berkeley.edu).

Denis Kleyko is with the Redwood Center for Theoretical Neuroscience, University of California, Berkeley, CA 94720 USA, and also with Intelligent Systems Lab, Research Institutes of Sweden, 164 40 Kista, Sweden.

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TNNLS.2021.3105949.

Digital Object Identifier 10.1109/TNNLS.2021.3105949

I. Introduction

In a traditional computer, the internal representation of data is organized by data structures. A data structure is a collection of data values with their relationships. For example, a simple data structure is a key-value pair, relating a variable name to its assigned value. Particular variables within data structures can be individually accessed for computations. Data structures are the backbones for computation, and needed for organizing, storing, managing, and manipulating information in computers.

For many tasks that brains have to solve, for instance, analogical inference in cognitive reasoning tasks and invariant pattern recognition, it is essential to represent knowledge in data structures and to query the components of data structures on the fly. It has been a long-standing debate if, and if so how, brains can represent data structures with neural activity and implement algorithms for their manipulation [1]–[4].

Here, we revisit classical connectionist models [5]–[7] that propose encodings of data structures [8]–[11] with distributed representations. Following [12], we will refer to these models as *vector symbolic architectures* (VSAs), synonymously they are also referred to as *hyperdimensional computing* [8]. Typically, VSA models use dense random vectors to represent atomic symbols, such as variable names and feature values. Through two elementary dyadic operations, *bundling* (or summation), and *binding*, atomic symbols can be combined into compound symbols that are represented by vectors that have the same dimension. The encoding of symbols with pseudo-random vectors is a fully distributed code, whose performance rests on the concentration of measure phenomenon [13]—the fact that random vectors become almost orthogonal in high-dimensional vector spaces. For example, the more orthogonal individual elements are in a sum vector, the higher the precision by which a simple projection can detect an individual element [14].

By contrast in conventional neural networks, features are often encoded locally by the activity of a single or of a few neurons and the resulting activity patterns can be sparse, i.e., activation vectors with only a few nonzero elements [15], [16]. Connectionists attempted to use such local feature representations in models describing computations in the brain. However, a critical issue emerged with these representations, known as *the binding problem* in neuroscience. This problem occurs when a representation requires the encoding of sets of feature conjunctions, for example, when representing a red triangle and a blue square [17]. Just representing the color and shape features would lose the

binding information that the triangle is red, not the square. One solution proposed for the binding problem is the tensor product representation (TPR) [3], where a neuron is assigned to each combination of feature conjunctions. However, when expressing hierarchical data structures, the dimensionality of TPRs grows exponentially with hierarchical depth. One proposal to remedy this issue is to form reduced representations of TPRs, whose dimension is the same dimension as for the atomic vectors [18], [19]. Such reduced representations have been the inspiration of VSAs, in which binding is a dyadic operation between dense vectors, an operation which preserves dimensionality. Building on earlier work on sparse VSA [2], [20], we investigate the possibility to build binding operations for sparse patterns that preserve dimensionality and sparsity.

II. Definitions and Foundations

A. Models for Symbolic Reasoning

Many connectionist models for symbolic reasoning with vectors use vector addition (or a thresholded form of it) to express sets of symbols. But there are characteristic differences in how these models encode information and in the operation they provide for variable binding. TPRs [3] use real-valued localist feature vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$ and the outer product $\mathbf{x}\,\mathbf{y}^\top \in \mathbb{R}^{N\times N}$ as the binding operation. This form of tensor product binding encodes compound data structures by representations that have higher dimensions than those of atomic symbols, i.e., the binding operation is not dimensionality-preserving. The deeper a hierarchical data structure, the higher the order of the tensor.

Building on Hinton's concept of reduced representations [21], several VSA models were proposed [5]–[7] in which the vector operations for set formation and binding are dimensionality-preserving and therefore atomic and composed data structures have the same dimension. These models encode atomic symbols by pseudorandom vectors and the vector operations are designed so that representations of compound symbols still resemble random vectors. Specifically, the VSA vector operations are the two dyadic operations *addition* ($+$) and *binding* ($\circ$) that, together with the vector space, form a ring-like algebraic structure. The desired properties for a binding operation are as follows:

1) Associative, i.e., $(\mathbf{a}\circ\mathbf{b})\circ\mathbf{c} = \mathbf{a}\circ(\mathbf{b}\circ\mathbf{c}) = (\mathbf{a}\circ\mathbf{c})\circ\mathbf{b}$.
2) Distributes over addition, i.e., $\sum_i^{D_1}\mathbf{a}^i \circ \sum_j^{D_2}\mathbf{b}^j = \sum_{i,j}^{D_1,D_2}\mathbf{c}^{ij}$ with $\mathbf{c}^{ij} = \mathbf{a}^i \circ \mathbf{b}^j$.
3) Has an inverse operation to perform unbinding.

Holographic reduced representation (HRR) [22], [23] was probably the earliest formalized VSA which uses real-valued Gaussian random vectors and circular convolution as the binding operation. Circular convolution is the standard convolution operation used in the discrete finite Fourier transform which can be used to produce a vector from two input vectors $\mathbf{x}$ and $\mathbf{y}$

$$\text{HRR}: (\mathbf{x}\circ\mathbf{y})_k := (\mathbf{x} * \mathbf{y})_k = \sum_{i=1}^{N} x_{(i-k)_{\bmod N}}\, y_i. \tag{1}$$

Note that (1) is a projection of the TPR matrix. Other VSA models use binding operations based on another projection of the TPR matrix that only samples the matrix diagonal. For example, the binary spatter code (BSC) [6] uses binary random vectors and binding is the XOR operation between components with the same index.

In the following, we focus on the *multiply-add-permute (MAP) model* [7], which uses bipolar atomic vectors whose components are -1 and 1. Atomic features or symbols are represented by random vectors of a matrix $\mathbf{\Phi}$ ("Phi"), called the *codebook*. The columns of $\mathbf{\Phi}$ are normalized i.i.d. random *code vectors*, $\mathbf{\Phi}_i \in \{\pm 1\}^N$. The binding operation is the Hadamard product $\odot$ between the two vectors

$$\text{MAP}: \ \mathbf{x}\circ\mathbf{y} := \mathbf{x}\odot\mathbf{y} = (x_1 y_1, x_2 y_2, \ldots, x_N y_N)^\top. \tag{2}$$

Note that the MAP model is a special case of VSA models in the complex domain, in which atomic symbols are represented by phasor vectors, i.e., dense complex vectors of unit-magnitude components [1].

When the binding involves just a scalar value, the multiplication operation (2) relaxes to ordinary vector-scalar multiplication. A feature with a particular value is simply represented by the vector representing the feature, $\mathbf{\Phi}_i$ (which acts like a "key"), multiplied with the scalar representing the "value" a_i: $\mathbf{x} = \mathbf{\Phi}_i a_i$.

For representing a *set of features*, the generic vector addition is used, and the vector representing a set of features with specific values is then given by

$$\mathbf{x} = \mathbf{\Phi}\mathbf{a}. \tag{3}$$

Here, the nonzero components of $\mathbf{a}$ represent the values of features contained in the set, the zero-components label the features that are absent in the set.

Although the representation $\mathbf{x}$ of this set is lossy, a particular feature value can be approximately decoded by forming the inner product with the corresponding "key" vector

$$a_i \approx \mathbf{\Phi}_i^\top \mathbf{x}/N \tag{4}$$

where N is the dimension of vectors. The crosstalk noise in the decoding (4) decreases with the square root of the dimension of the vectors or by increasing the sparseness in $\mathbf{a}$, for analysis of this decoding procedure, see [14].

To represent a set of sets, one cannot simply form a sum of the compound vectors. This is because the set information on the first level is lost, which is exactly the binding problem described in the introduction. VSAs can solve this issue by combining addition and binding to form a representation of a set of compound objects in which the integrity of individual objects is preserved. This is sometimes called the *protected sum* of L objects

$$\mathbf{s} = \sum_{j}^{L} \mathbf{\Psi}_j \odot \mathbf{x}^{(j)} \tag{5}$$

where $\mathbf{\Psi}_j$ ("Psi") are dense bipolar random vectors that label the different compound objects. Another method for

representing protected sums uses powers of a single random permutation matrix $\mathbf{P}$ [2], [14]

$$\mathbf{s} = \sum_{j}^{L} \mathbf{P}^{j-1}\mathbf{x}^{(j)}. \tag{6}$$

In general, algebraic manipulation in VSAs yields a noisy representation of the result of a symbolic reasoning procedure. To filter out the result, so-called *cleanup memory* is required, which is typically nearest-neighbor search through a content-addressable memory or associative memory [15], [24], [25] storing the codebook(s).

B. Sparse Distributed Representations

Classical VSAs described in Section II-A use dense representations, that is, vectors in which most components are nonzero. In the context of neuroscience and neural networks for unsupervised learning and synaptic memory, another type of representation has been suggested: *sparse representations*. In sparse representations, a large fraction of components are zero, e.g., most neurons are silent. There is evidence that the brain uses sparse representations. The fraction of active neurons at any point in time is only a few percent [26], [27]. Furthermore, learning sparse representations for natural images or natural sounds leads to response properties that capture essential aspects of real sensory neurons in the brain [16], [28], [29].

Here, we will investigate how sparse representations can be used in VSAs. For the cleanup required in VSAs, sparse representations have the advantage that they can be stored more efficiently than dense representations in Hebbian synapses [15], [24], [30]–[32]. A few previous studies [2], [20] have proposed VSA operations on sparse vectors, however, the properties of these operations have not been studied systematically.

A subclass of sparse representations with additional structure has been proposed for symbolic reasoning before that we call *sparse block-codes* [2]. In a K-sparse block-code, the ratio between active components and total number of components is K/N, as usual. But the index set is partitioned into K blocks, each block is of size $L = N/K$ and has only a single active element.[1]

The constraint of a sparse block-code reduces the entropy in a code vector significantly, from $\log\left(\binom{N}{K}\right)$ to $K\log(N/K)$ bits [34]. At the same time, the block constraint can also be exploited to improve the retrieval in Hebbian associative memory. As a result, the information capacity of associative memories with Hebbian synapses for block-coded sparse vectors is almost the same as for unconstrained sparse vectors [35], [36]. Sparse block-codes can be regarded as an extreme version of competitive coding principles observed in the brain, for example, through competition between sensory neurons [37], as seen in orientation hypercolumns in the visual system of certain species [38].

[1]Note that sparse block-codes differ from sparse block signals [33], in the latter the activity within blocks can be nonsparse but the nonzero blocks is K'-sparse, with $K' \ll K$. The resulting N-dimensional vectors have a ratio between active components and total number of components of $K'L/N = K'/K$.

Here, we also consider a variation of the sparse block-code where active entries are phasors. Recent work has shown that sparse phasor vectors can be efficiently stored in associative memories and that this coding scheme can be implemented in spiking neural networks [32].

C. Compressed Sensing

Under certain conditions, there is unique equivalence between sparse and dense vectors, which has been exploited for signal processing under the name *compressed sensing* (CS) [39], [40]. Many types of signals, such as images or sounds, have a sparse underlying structure and CS can be used as a signal compression method in applications, and even for modeling communication in biological brains [41]. For example, if one assumes that the data vectors are K-sparse, that is

$$\mathbf{a} \in \mathcal{A}_K := \{\mathbf{a} \in I\!R^M : ||\mathbf{a}||_0 \leq K\} \tag{7}$$

with $||.||_0$ the L0-norm, then in CS, the following linear transformation creates a dimensionality-compressed dense vector from the sparse data vector:

$$\mathbf{x} = \Xi\mathbf{a} \tag{8}$$

where Ξ ("Xi") is a $N \times M$ random sampling matrix, with $N < M$.

Due to the distribution of sparse random vectors $\mathbf{a}$, the statistics of the dimensionality-compressed dense vectors $\mathbf{x}$ becomes somewhat non-Gaussian. The data vector can be recovered from the compressed vector $\mathbf{x}$ by solving the following sparse inference problem:

$$\hat{\mathbf{a}} = \text{argmax}_{\mathbf{a}}(\mathbf{x} - \Xi\mathbf{a})^2 + \lambda|\mathbf{a}|_1. \tag{9}$$

The condition for K, N, M, and Ξ, under which the recovery (9) is possible, forms the cornerstones of CS [39], [40].

For CS to work, a necessary condition is that the sampling matrix is injective for the sparse data vectors, i.e., that the intersection between the kernel of the sampling matrix, $\text{Ker}(\Xi) = \{\mathbf{a} : \Xi\mathbf{a} = 0\}$, and the set of sparse data vectors, $\mathcal{A}_K$, is empty: $\text{Ker}(\Xi) \cap \mathcal{A}_K = \emptyset$. But, this condition does not guarantee that each data vector has a unique dense representation. In other words, the mapping between data vectors and dense representations must also be bijective. To guarantee uniqueness of the dense representation of K-sparse vectors, the kernel of the sampling matrix must not contain any $(2K+1)$-sparse vector

$$\text{Ker}(\Xi) \cap \mathcal{A}_{2K+1} = \emptyset \tag{10}$$

with $\mathcal{A}_{2K+1}$ being the set of $(2K+1)$-sparse vectors. Intuitively, the condition (10) excludes that any two K-sparse data vectors can have the same dense representation: $\mathbf{a}_1 \neq \mathbf{a}_2$: $\Xi\mathbf{a}_1 - \Xi\mathbf{a}_2 = 0$.

Even with condition (10), it still might not be possible to infer the sparse data vectors from the dense representations (9) in the presence of noise. Another common criterion for CS to work is the s-restricted isometry property (RIP)

$$(1-\delta_s)||\mathbf{a}^{(s)}||_2^2 \leq ||\Xi\mathbf{a}^{(s)}||_2^2 \leq (1+\delta_s)||\mathbf{a}^{(s)}||_2^2 \tag{11}$$

with the vector $\mathbf{a}^{(s)}$ s-sparse, and the RIP constant $\delta_s \in (0, 1)$. The choice $\delta_{2K+1} = 1$ is equivalent to condition (10). With a choice $\delta_{2K+1} = \delta^* < 1$, one can impose a more stringent condition that enables the inference, even in the presence of noise. The minimal dimension of the compression vector that guarantees (11) is typically linear in K but increases only logarithmically with M:

$$N \geq C K \log\left(\frac{M}{K}\right) \tag{12}$$

where C is a constant of order $O(1)$ that depends on δ_{2K+1}. Here, we will use the uniqueness conditions (10) and (11) to assess the equivalence between different models of symbolic reasoning.

D. Connection Between Symbolic Reasoning, Sparse Representations, and CS

The various topics described in this section are not commonly perceived as closely associated. The connection between these topics is one of the original contributions of this article. Specifically, Section III uses CS as an analysis framework to identify the operations between sparse vectors that are mathematically equivalent to the dimensionality-preserving operations for variable binding and protected sum in traditional dense VSAs. Interestingly, it turns out that the equivalent operations in sparse symbolic reasoning models are tensor product binding and vector concatenation, both of which are not dimensionality-preserving. Section IV focuses on dyadic functions as binding operations for symbolic reasoning with sparse representations that are both, sparsity- and dimensionality-preserving. Most existing binding methods, such as Hadamard product, circular convolution [5] and vector-derived transformation binding (VDTB) [42] are dimensionality—but not sparsity-preserving. The sparsity-preserving methods we propose, are either lossy or defined on the subset of block-sparse vectors. In spite of the restriction to a subspace of sparse patterns, our experiments show that VSAs with block-sparse vectors have interesting properties in applications.

III. Equivalent Representations With Sparse Versus Dense Vectors

Here, we consider a setting where sparse and dense symbolic representations can be directly compared. Specifically, we ask what operations between K-sparse vectors are induced by the operations in the MAP VSA. To address this question, we map K-sparse feature vectors to corresponding dense vectors via (3). The column vectors of the codebook in (3) correspond to the atomic dense vectors in the VSA. We choose the dimension N and properties of the codebook(s) and sparse random vectors so that the CS condition (10) is fulfilled.[2] Thus, each sparse vector has a unique dense representation and vice versa.

[2]In CS, the choice of sampling matrices with binary or bipolar random entries is common, e.g., [43].

A. Improved VSA Decoding Based on CS

In our setting, the coefficient vector $\mathbf{a}$ is sparse. The standard decoding method in VSA (4) provides a noisy estimate of the sparse vector (Fig. 1) from the dense representation. However, if the sparse vector and the codebook $\mathbf{\Phi}$ in (3) satisfy the CS conditions, one can do better: decoding à la CS (9) achieves near-perfect accuracy (Fig. 1). Note that sparse inference requires that the entire coefficient vector $\mathbf{a}$ is decoded at once, similar to [44], while with (4) individual values a_i can be decoded separately. If the CS condition is violated, sparse inference (9) abruptly ceases to work, while the VSA decoding with (4) degrades gradually [14], [45], [46].

B. Variable Binding Operation

The Hadamard product between dense vectors turns out to be a function of the tensor product, i.e., the TPR, of the corresponding sparse vectors

$$\begin{aligned}(\mathbf{x} \odot \mathbf{y})_i &= (\mathbf{\Phi a})_i (\mathbf{\Psi b})_i = \sum_l \Phi_{il} a_l \sum_k \Psi_{ik} b_k \\ &= \sum_{lk} \Phi_{il} \Psi_{ik} a_l b_k = ((\mathbf{\Phi} \boxdot \mathbf{\Psi}) \mathrm{vec}(\mathbf{a}\mathbf{b}^\top))_i. \end{aligned} \tag{13}$$

This linear relationship between the Hadamard product of two vectors and the TPR can be seen as a generalization of the Fourier convolution theorem (see Section I in the Supplementary Materials). The reshaping of the structure on the RHS of (13) shows that there is a relationship to the matrix vector multiplication in CS sampling (8): the raveled tensor product matrix of the sparse vectors, $\mathrm{vec}(\mathbf{a}\ \mathbf{b}^\top)$, becomes a M^2-dimensional vector with K^2 nonzero elements. Furthermore, $(\mathbf{\Phi} \boxdot \mathbf{\Psi})$ is a $N \times M^2$ sampling matrix, formed by pair-wise Hadamard products of vectors in the individual dictionaries $\mathbf{\Phi}$ and $\mathbf{\Psi}$

$$(\mathbf{\Phi} \boxdot \mathbf{\Psi}) := (\mathbf{\Phi}_1 \odot \mathbf{\Psi}_1, \mathbf{\Phi}_1 \odot \mathbf{\Psi}_2, \ldots, \mathbf{\Phi}_M \odot \mathbf{\Psi}_M). \tag{14}$$

One can now ask under what conditions Hadamard product and tensor product become mathematically equivalent, that is, can any sparse tensor product in (13) be uniquely inferred from the Hadamard product using a CS inference procedure (9). The following two lemmas consider a worst case scenario in which there is equivalence between the atomic sparse and dense vectors, which requires that the sparks of the individual codebooks are at least $2K + 1$.

Lemma 1: Let $\mathrm{Spark}(\mathbf{\Phi}) = \mathrm{Spark}(\mathbf{\Psi}) = 2K + 1$. Then the spark of the sampling matrix in (13) is $\mathrm{Spark}((\mathbf{\Phi} \boxdot \mathbf{\Psi})) \leq 2K + 1$.

Proof: Choose a $(2K + 1)$-sparse vector $\mathbf{c}$ in the kernel of $\mathbf{\Phi}$, and choose any cardinal vector $\mathbf{b}^{(j)} := (0, \ldots, 0, 1, 0, \ldots, 0)$ with the nonzero component at index j. Then we have: $0 = \mathbf{\Phi}\boldsymbol{\alpha} = \mathbf{\Phi c} \odot \mathbf{\Psi}_j = \sum_{i \in \boldsymbol{\alpha}} c_i \mathbf{\Phi}_i \odot \mathbf{\Psi}_j = (\mathbf{\Phi} \boxdot \mathbf{\Psi}) \mathrm{vec}(\mathbf{c} \otimes \mathbf{b}^{(j)})$. Thus, the $(2K+1)$-sparse vector $\mathrm{vec}(\mathbf{c} \otimes \mathbf{b}^{(j)})$ lies in the kernel of the sampling matrix in (13). There is also a small probability that the construction of $(\mathbf{\Phi} \boxdot \mathbf{\Psi})$ produces a set of columns with less than $2K + 1$ components that are linearly dependent. □

Lemma 1 reveals that the sampling matrix (14) does certainly not allow the recovery of K^2-sparse patterns in general.

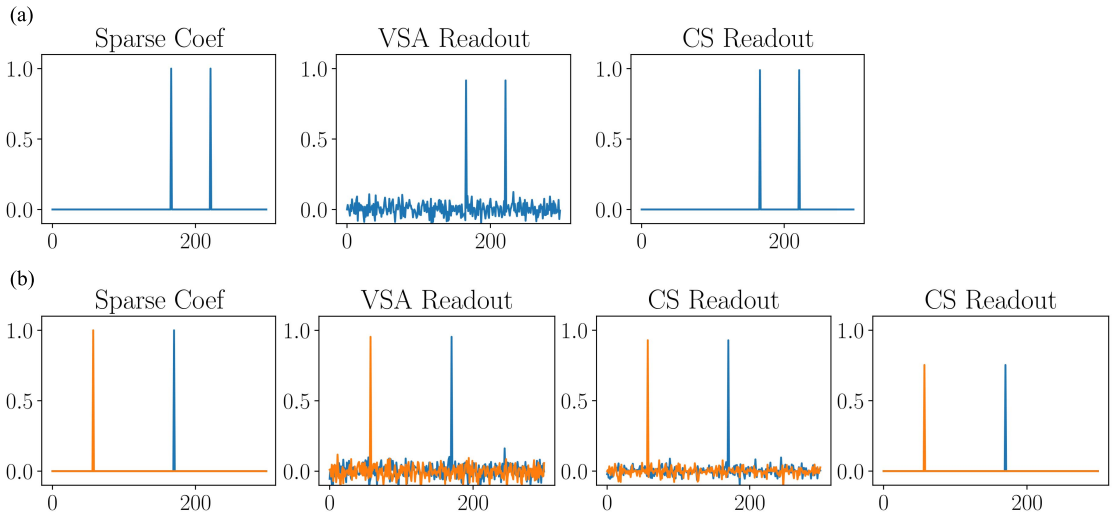

Fig. 1. Readout of sparse coefficients from dense distributed representation. (a) Sparse coefficients (left) are stored as a dense representation using a random codebook. The coefficients are recovered with standard VSA readout (middle) and with sparse inference (right), which reduces the crosstalk noise. (b) Two sparse coefficients are stored as a protected set (left). Readout with sparse inference reduces crosstalk noise, but some noise can remain depending on sparsity penalty (far right panel has increased sparsity penalty).

However, this is not required since the reshaped outer products of K-sparse vectors form a subset of K^2-sparse patterns. The following lemma shows that for this subset recovery can still be possible.

Lemma 2: The difference between the outer-products of pairs of K-sparse vectors cannot fully coincide in support with the $(2K+1)$-sparse vectors in the kernel of the sampling matrix of (13) as identified by Lemma 1. Thus, although $\text{Spark}((\mathbf{\Phi} \boxdot \mathbf{\Psi})) \leq 2K+1$, the recovery of reshaped tensor products from the Hadamard product can still be possible.

Proof: The $(2K+1)$-sparse vectors in the kernel of the sampling matrix $(\mathbf{\Phi} \boxdot \mathbf{\Psi})$ identified in Lemma 1 correspond to an outer product of a $(2K+1)$-sparse vector with a 1-sparse vector. The resulting matrix has $2K+1$ nonzero components in one single column.

The difference of two outer products of K-sparse vectors yields a matrix that can have maximally $2K$ nonzero components in one column. Thus, the sampling matrix should enable the unique inference of the tensor product from the Hadamard product of the dense vectors. □

Lemmas 1 and 2 investigate the equivalence of Hadamard and tensor product binding in the worst case, that is, when the codebooks have the minimum spark that still guarantees the unique equivalence between the sparse and dense atomic vectors. To explore the equivalence in the case of random codebooks, we performed simulation experiments with a large ensemble of randomly generated codebook pairs $(\mathbf{\Phi}, \mathbf{\Psi})$. Fig. 2 shows the averaged worst (i.e., highest) RIP constant amongst the ensembles for inferring the tensor product from the Hadamard product (solid red line).

Compared to the RIP constant for inferring the sparse representations of atomic vectors (black line), the RIP constant for inferring the tensor product (red line) is significantly higher. Thus, tensor product and Hadamard product are not always equivalent even if the atomic sparse and dense vectors are equivalent—in the example, when the dimension of dense

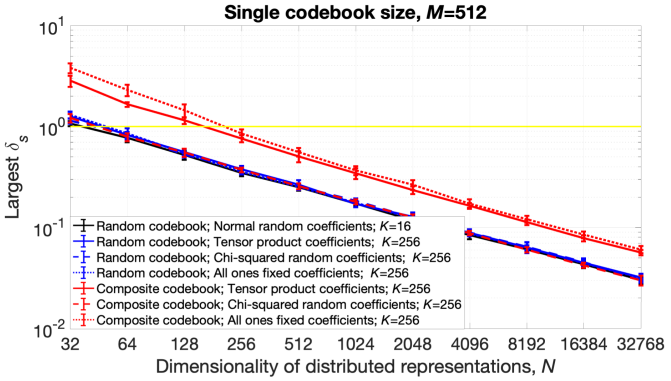

Fig. 2. Worst case RIP constant for inferring sparse tensor products in ensemble of random codebooks. The largest empirical RIP constant (δ_s) in an ensemble of ten pairs of pseudorandom dictionaries $\mathbf{\Phi}, \mathbf{\Psi}$. For each pair, the maximum RIP was determined by compressing 10 000 sparse vectors. For successful inference of the sparse representations, the RIP constant has to be below the $\delta_s = 1$ level (yellow line). The black solid line represents the RIP for inferring atomic sparse vectors from dense vectors formed according to (8). The red solid line represents the RIP for inferring tensor products from dense vectors formed according to (13). Other lines in the diagram are controls. The blue solid line represents the RIP for a $(N \times M^2)$ random dictionary in which all elements are independently sampled rather than constructed by $\mathbf{\Phi} \boxdot \mathbf{\Psi}$ from the smaller dictionaries. Dashed and dotted red lines represent the RIP using the $\mathbf{\Phi} \boxdot \mathbf{\Psi}$ sampling matrix with sparse vectors with independent random components, rather than formed by a tensor product $\text{vec}(\mathbf{a}\ \mathbf{b}^\top)$ of two random vectors. The blue dashed line describes real-valued vectors with elements sampled from a Chi-squared distribution and the blue dotted line for binary random vectors. These dashed and dotted blue lines represent the RIP for the same type of independent random vectors with the independent random sampling matrix.

vectors is between $N = 40$ to $N = 140$. However, with the dimension of dense vectors large enough ($N > 140$), the equivalence holds. Furthermore, the controls in Fig. 2 help to explain the reasons for the gap in equivalence for small dense vectors. The RIP constants are significantly reduced if the tensor product is subsampled with a fully randomized matrix (blue solid line), rather than with the sampling matrix

resulting from (13). In contrast, the requirement to infer outer products of continuous-valued random vectors (solid red line) does not much increase the RIP values over the RIP requirement for the inference of outer products of binary vectors (dotted red line). Thus, we conclude that sampling with matrix $\mathbf{\Phi} \boxdot \mathbf{\Psi}$ (14), which is not i.i.d. random but formed by a deterministic function from the smaller atomic random sampling matrices, requires a somewhat bigger dimension of the dense vectors to be invertible.

Here, we have shown that under certain circumstances the binding operation between dense vectors in the MAP VSA is mathematically equivalent to the tensor product between the corresponding sparse vectors. This equivalence reveals a natural link between two prominent proposals for symbolic binding in the literature, the dimensionality preserving binding operations in VSA models, with the tensor product in the TPR model [3], [47]. In other VSA models, such as HRR [1], atomic symbols are represented by dense Gaussian vectors and the binding operation is circular convolution. Our treatment can be extended to these models by simply noting that by the Fourier convolution theorem (see (4) in Section I.B in the Supplementary Materials) circular convolution is equivalent to the Hadamard product in the Fourier domain, i.e., $\mathbf{x} * \mathbf{y} = \mathcal{F}^{-1}(\mathcal{F}(\mathbf{x}) \odot \mathcal{F}(\mathbf{y}))$.

C. Set Operations

Summing dense vectors corresponds to summing the sparse vectors

$$\mathbf{x} + \mathbf{y} = \mathbf{\Phi}(\mathbf{a} + \mathbf{b}). \tag{15}$$

Thus, the sum operation represents a bag of features from all objects, but the grouping information on how these features were configured in the individual objects is lost. The inability to recover the individual compound objects from the sum representation has been referred to as the binding problem in neuroscience [17].

The protected sum of set vectors (5) can resolve the binding problem. This relies on binding the dense representations of the individual objects to a set of random vectors that act as keys, stored in the codebook $\mathbf{\Psi}$ (5)

$$\sum_j^L \mathbf{\Psi}_j \odot \mathbf{x}^j = \sum_j^L \mathbf{\Psi}_j \odot \sum_i^M \mathbf{\Phi}_i a_i^j = (\mathbf{\Phi} \boxdot \mathbf{\Psi})(\mathbf{a}^1, \mathbf{a}^2, \ldots, \mathbf{a}^L). \tag{16}$$

This shows that the protected sum can be computed from the concatenation of sparse vectors. The concatenation of sparse vectors is a representation that fully contains the binding information, but again leads to an increase in dimensionality. Similar to (13), (16) describes linear sampling of a sparse vector like in CS. The sampling matrix $(\mathbf{\Phi} \boxdot \mathbf{\Psi})$ is a $N \times ML$ sampling matrix formed by each pair of vectors in $\mathbf{\Phi}$ and $\mathbf{\Psi}$, as in (14), and the sparse vector is the ML-dimensional concatenation vector.

We again ask under what conditions the sparse concatenation vector can be uniquely inferred given the dense representation of the protected sum, which makes the dense and sparse representations equivalent. Like in Section III-B, we first look at the worst case scenario, and then perform an experiment with codebooks composed of random vectors. The worst case scenario assumes the spark of $\mathbf{\Phi}$ to be $2K + 1$, just big enough that atomic vectors can be inferred uniquely. By Lemma 1, the spark of the sampling matrix is smaller or equal to $2K + 1$, smaller than the sparsity KL of vectors to be inferred. Again, the vectors to be inferred are a subset of KL-sparse vectors, the vectors that have K nonzero components in each of the L M-sized compartments. Thus, as in Lemma 2 for the Hadamard product, the difference formed by two of these vectors can maximally produce $2K$ nonzero components in each compartment, and therefore never coincide with a kernel vector of the sampling matrix.

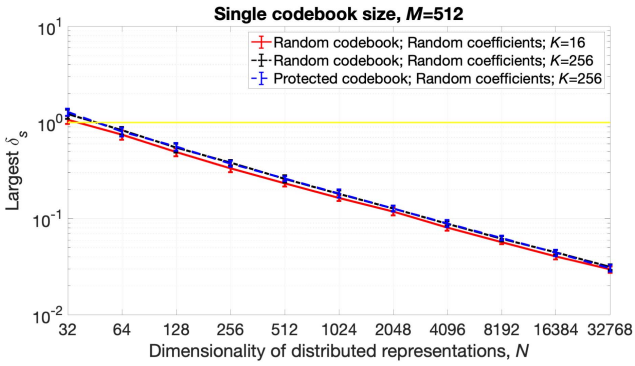

Fig. 3. Worst case RIP constant for inferring sparse representations of protected sums in ensemble of random codebooks. The largest empirical RIP constant (δ_s) in ensemble of ten pairs of pseudorandom dictionaries $\mathbf{\Phi}, \mathbf{\Psi}$. For each pair the maximum RIP was determined by compressing 10 000 sparse vectors. For successful inference of the sparse representations, the RIP constant has to be below the $\delta_s = 1$ level (yellow line). Red solid line represents RIP values for inferring atomic sparse vectors from dense vectors formed according to (8). Blue dashed line represents RIP values for inferring protected sum from dense vectors formed according to (16). For comparison, black dotted line represents RIP values for inferring protected sum when instead of $\mathbf{\Phi} \boxdot \mathbf{\Psi}$ the dictionary is random.

Fig. 3 shows the results of simulation experiments with an ensemble of random codebooks. For the protected sum, the worst RIP values of the inference of individual sparse vectors versus the list of sparse vectors composing the protected sum do coincide. Thus, the dense protected sum vector and the list of sparse feature vectors are equivalent.

The alternative method of forming a protected sum (6) using powers of a permutation matrix $\mathbf{P}$, corresponds equally to a sampling of the concatenation of the sparse vectors. As long as the sampling matrices $(\mathbf{\Phi}, \mathbf{P\Phi}, \mathbf{P}^2\mathbf{\Phi}, \ldots, \mathbf{P}^{(L-1)}\mathbf{\Phi})$ and $(\mathbf{\Phi} \boxdot \mathbf{\Psi})$ have similar properties, the conditions for equivalence between protected sum and concatenated sparse vectors hold.

IV. Dimensionality- and Sparsity-Preserving VSA Operations

The results from Section III reveal that variable binding and the protected set representation in classical VSA models induce equivalent operations between sparse vectors that are not dimensionality preserving. Thus, dimensionality-preserving operations for binding and protected sum involve potentially lossy transformations of the higher dimensional

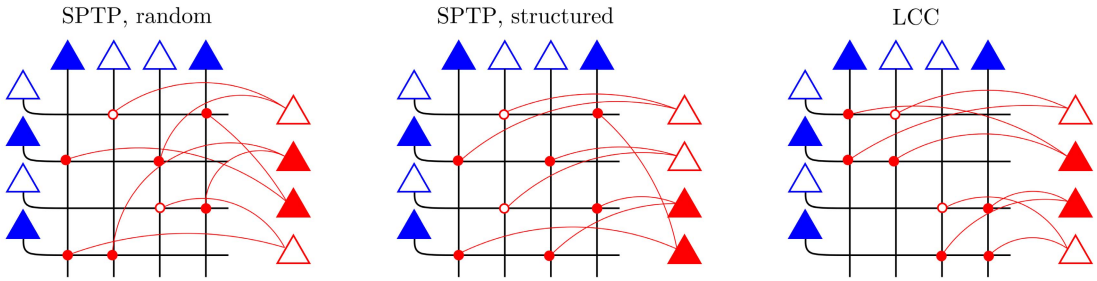

Fig. 4. Circuits for sparsity-preserving binding: three pools of neurons (blue: two inputs, red: output) represent the sparse neural activity patterns **a**, **b** and **c**. The dendritic tree of the output neurons contains coincidence detectors that detect pairs of co-active axons (red circles), and the soma (red triangles) sums up several coincidence detectors based on the required fan-in. Each neuron samples only a subset of the outer product depending on the desired sparsity and threshold settings. The subsampling pattern of neurons is described by a binary tensor $W^l_{ij} \in \{0, 1\}$, where i, j indexes the coincidence point and l the postsynaptic neuron. We examine three different sampling strategies random sampling, structured sampling, and block code.

data structure into a single vector. In the following, we investigate binding operations on sparse VSA representations that are both dimensionality- and sparsity-preserving, one for general sparse vectors and one for sparse vectors with block structure.

A. Sparsity-Preserving Binding for General K-Sparse Vectors

Binding operations in classical VSAs can all be described as a projection of the tensor product to a vector, including Hadamard product, circular convolution binding [1] and VDTB [42] (see (2) in Section I.A in the Supplementary Materials). However, when applied to sparse atomic vectors, these operations do not preserve sparsity—circular convolution produces a vector with reduced sparsity, while the Hadamard product increases sparsity.

Ideally, a sparsity-preserving VSA binding operation operates on two atomic vectors that are K-sparse and produces a K-sparse vector that has the correct algebraic properties. To preserve sparsity, we developed a binding operation that uses a random projection to subsample the tensor product. We refer to this operation as *sparsity-preserving tensor projection (SPTP)*. Given two K-sparse binary vectors **a** and **b**, SPTP variable binding is given by

$$(\mathbf{a} \underline{\circ} \mathbf{b})_l = H\left(\sum_{ij} W^l_{ij} a_i b_j - \theta\right). \tag{17}$$

Here, $H(x)$ is the Heaviside function, θ is a threshold. For a pair of K-sparse complex phasor vectors, SPTP binding is defined as

$$(\mathbf{a} \underline{\circ} \mathbf{b})_l = \frac{z_l}{|z_l|} H(|z_l| - \theta)$$
$$z_l = \sum_{ij} W^l_{ij} a_i b_j. \tag{18}$$

The computation of (17) resembles a circuit of threshold neurons with coincidence detectors in their dendritic trees, see Fig. 4. The synaptic tensor $\mathbf{W} \in \{0, 1\}^{M \times M \times M}$ is a binary third-order tensor that indicates how each output neuron samples from the outer-product. We examined two types of random sampling tensors, one with the 1-entries chosen i.i.d. (without repetition), and one with 1-entries aligned along truncated diagonals of the tensor (left and middle panel in Fig. 4).

The sparsity of the output in (17) is controlled by the threshold and by the density of this sampling tensor. To achieve a target sparsity of K/N for a threshold $\theta = 1$, the fan-in to each neuron has to be N/K (see analysis in Section II.A in the Supplementary Materials). Thus, the minimal fan-in of the sampling tensor $\mathbf{W}$ increases with sparsity. If the pattern activity is linear in the dimension, $K = \beta N$ with $\beta \ll 1$, the minimal fan-in is $\alpha^* = 1/\beta$. In this case, the computational cost of SPTP binding is order N. If the pattern activity goes with the square root of the dimension, $K = \beta\sqrt{N}$, the minimal fan-in is $\alpha^* = \sqrt{N}/\beta$. If the pattern activity goes with the logarithm of the dimension, $K = \beta \ln(N)$, the minimal fan-in is $\alpha^* = N/(\beta \ln(N))$. Furthermore, for optimizing the unbinding performance, the sampling tensor should fulfill the symmetry condition $W^i_{jl} = W^l_{ij}$ (see analysis in Section II.B in the Supplementary Materials).

B. Sparsity-Preserving Binding for Sparse Block-Codes

We next consider sparse vector representations that are constrained to be block codes [35]. Here, we extend a previous VSA model with sparse binary block-code [2] to a VSA model with complex phasor block-codes. In a sparse block-code, a vector of length N is divided into K equally-sized blocks, each with a one-hot component. In the complex domain, the hot component is a phasor with unit amplitude and arbitrary phase.

The binding operation [2] proposed operates on each block individually. For each block, the indices of the two active elements of the input are summed modulo block size to produce the index for the active element of the output. In essence, this is circular convolution [5] performed locally between individual pairs of blocks. This binding operation, *local circular convolution* (LCC), denoted by $*_b$, produces a sparse block-code when the input vectors are sparse block-codes (Fig. 4). LCC variable binding can be implemented by forming the outer product and sampling as in Fig. 4, with circuitry in which each neuron has a fan-in of $\alpha = N/K$ and samples along truncated diagonals of the tensor product. LCC has a computational complexity of αN, which is order N if K is proportional to N. An alternative implementation (that is more efficient on a CPU) uses the Fourier convolution theorem (see (4) in Section I.B in the Supplementary Materials) to

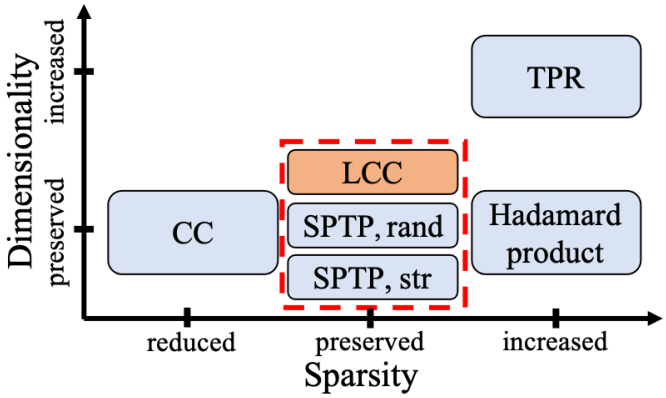

Fig. 5. Methods for variable binding in terms of dimensionality and sparsity preservation. The red dashed line marks the three binding methods that are dimensionality- and sparsity-preserving.

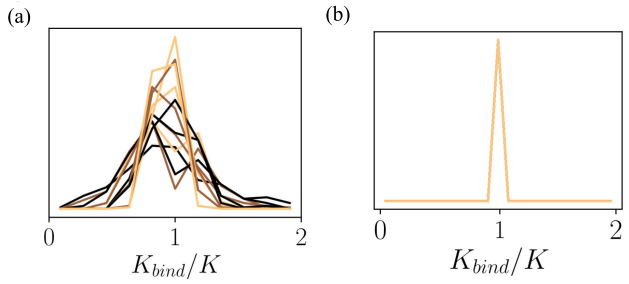

Fig. 6. Preservation of sparsity with a binding operation. (a) Output sparsity K_{bind} is compared to the sparsity of the base vectors K. Binding with SPTP results in an output vector that has the correct expected sparsity, but there is some random variance. This variance reduces with more active components ($K = [20, 50, 100, 200]$ black to orange lines). This result is similar for both random and structured SPTP. (b) Output sparsity of binding sparse block-codes with LCC deterministically results in a vector that maintains the sparsity of the inputs.

replace convolution by the Hadamard product

$$(\mathbf{a} *_b \mathbf{b})_{\text{block}_i} = \mathbf{a}_{\text{block}_i} * \mathbf{b}_{\text{block}_i} = \mathcal{F}^{-1}(\mathcal{F}(\mathbf{a}_{\text{block}_i}) \odot \mathcal{F}(\mathbf{b}_{\text{block}_i})) \quad (19)$$

where $\mathcal{F}$ is the Fourier transform.

The LCC unbinding of a block can be performed by computing the inverse of the input vector to unbind. This is the inverse with respect to circular convolution, which is computed for each block

$$\mathbf{a}^{-1}_{\text{block}_i} = \mathcal{F}^{-1}(\mathcal{F}(\mathbf{a}_{\text{block}_i})^*) \quad (20)$$

where * is the complex conjugate. The inverse is used when unbinding, for instance, if $\mathbf{c} = \mathbf{a} *_b \mathbf{b}$, then $\mathbf{a} = \mathbf{b}^{-1} *_b \mathbf{c}$.

Experiments With Sparsity-Preserving Binding: We have discussed and introduced different proposals for variable binding. Their properties regarding the preservation of dimension and sparsity are summarized in Fig. 5.

Here we perform experiments to evaluate for the dimensionality-preserving methods how well sparsity is preserved, and how much information of a bound object can be retrieved by unbinding. We investigated how precisely sparsity is preserved with LCC and SPTP binding (Fig. 6). Circular convolution and Hadamard product were excluded from this test because they systematically alter sparsity, the Hadamard product by increasing, and circular convolution by decreasing it (Fig. 5). We find that LCC binding preserves sparsity perfectly, and SPTP binding preserves sparsity on average (statistically), but with some variance.

We next measure how much information is retained when first binding and then unbinding a vector using the proposed binding operations (Fig. 7). The Hadamard product binding achieves the highest correlation values for dense vectors but performs very poorly for sparse vectors. The other three binding methods perform equally across sparsity levels. Circular convolution and SPTP binding are somewhat lossy for all sparsity levels. The LCC variable binding between block-codes achieves the highest correlation values, outperforming circular convolution, and SPTP binding. Each diagram in Fig. 7 contains six curves, corresponding to different levels of additive superposition in the bound vectors.

SPTP binding works for general K-sparse vectors. It has decent properties but is somewhat lossy. The information loss is due to the fact that not all active input components contribute to the generation of active outputs, which means that some active input components cannot be inferred during unbinding and information is lost. The loss can be kept at a minimum by using a synaptic weight tensor that fulfills the symmetry condition $W^i_{jl} = W^l_{ij}$. This information loss persisted regardless of SPTP being structured or random, or the threshold and fan-in settings.

These experiments identify LCC binding as an ideal sparsity-preserving binding operation. With sparse block-codes and LCC applied separately to each block, the unbinding is loss-less. The block structure guarantees that each active input component participates in the formation of an active output component, which cannot be guaranteed for general K-sparse vectors. Of course, there is a price to pay, LCC binding requires the atomic vectors to be sparse block-codes. The coding entropy of block-codes is significantly smaller than general K-sparse patterns.

V. Application and Performance of VSAs With Sparse Block-Codes

A. Solving Symbolic Reasoning Problems

As a basic illustration of symbolic reasoning with sparse block-codes, we implement the solution to the cognitive reasoning problem [48]: "What's the dollar of Mexico?" in the supplemental Jupyter notebook. We show how reasoning in VSA can answer such queries, given a set of VSA vectors that represent data records with trivia information about individual countries. Note that this is a toy problem, meant for illustrating how such reasoning problems can be implemented with the sparse VSA. While we do not compare the VSA solution with any other baseline solution, we assess quantitatively how the accuracy of the VSA solution depends on the vector dimension and the number of items in a data record, and how this compares to the theoretical prediction.

A data record of a country is a table of key-value pairs. For example, to answer the specific query, the relevant

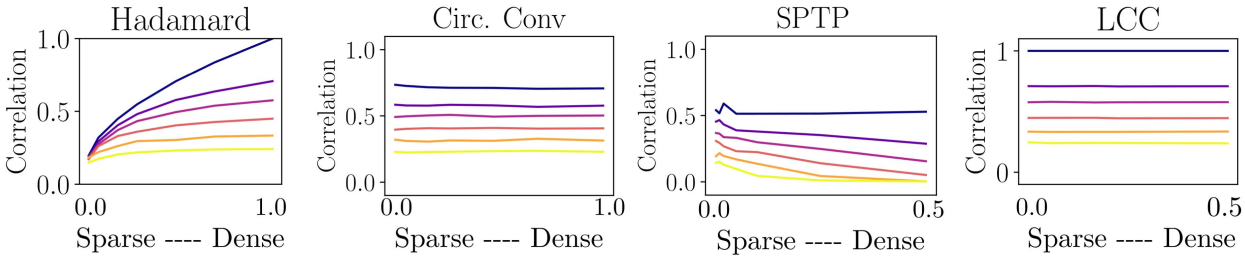

Fig. 7. Comparison of binding operations. The unbinding performance was measured as the correlation between ground truth and output of unbinding. Different levels of sparsity (x-axis) and superposition were examined (colored lines: [0, 1, 2, 4, 8 16] items in superposition).

records are

$$\mathbf{ustates} = \mathbf{nam} *_b \mathbf{usa} + \mathbf{cap} *_b \mathbf{wdc} + \mathbf{cur} *_b \mathbf{dol}$$
$$\mathbf{mexico} = \mathbf{nam} *_b \mathbf{mex} + \mathbf{cap} *_b \mathbf{mxc} + \mathbf{cur} *_b \mathbf{pes}.$$

The keys of the fields *country name*, *capital* and *currency* are represented by random sparse block-code vectors **nam**, **cap** and **cur**. The corresponding values *USA*, *Washington D.C.*, *Dollar*, *Mexico*, *Mexico City*, and *Peso* are also represented by sparse block-code vectors **usa**, **wdc**, **dol**, **mex**, **mxc**, **pes**. All the vectors are stored in the codebook $\mathbf{\Phi}$. The vectors **ustates** and **mexico** represent the complete data records—they are a representation of key-value pairs that can be manipulated to answer queries. These record vectors have several terms added together, which reduces the sparsity.

To perform the reasoning operations required to answer the query, first the two relevant records have to be retrieved in the database. While **mexico** can be found by simple pattern matching between terms in the query and stored data record vectors, the retrieval of **ustates** is not trivial. The original work [48] does not deal with the challenge of inferring that the **ustates** record is needed. Rather, the problem is formally expressed as analogical reasoning, where the query is given as: *Dollar*:*USA*::?:*Mexico*. Thus, the pair of records needed for reasoning are given by the query.

Once the pair of records is identified, the following transformation vector is created:

$$\mathbf{t}_{\mathrm{UM}} = \mathbf{mexico} *_b \mathbf{ustates}^{-1}.$$

Note that unbinding with LCC is to bind with the inverse vector (20), whereas in the MAP VSA used in the original work [48] the binding and unbinding operations are the same. The transformation vector will also contain many summed terms, leading to less sparsity. The transformation vector then contains the relationships between the different concepts

$$\mathbf{t}_{\mathrm{UM}} = \mathbf{mex} *_b \mathbf{usa}^{-1} + \mathbf{mxc} *_b \mathbf{wdc}^{-1} + \mathbf{pes} *_b \mathbf{dol}^{-1} + \text{noise}$$

where all of the cross-terms can be ignored and act as small amounts of crosstalk noise.

The correspondence to dollar can be computed by binding **dol** to the transformation vector

$$\mathbf{ans} = \mathbf{dol} *_b \mathbf{t}_{\mathrm{UM}} = \mathbf{pes} + \text{noise}.$$

The vector **ans** is then compared to each vector in the codebook $\mathbf{\Phi}$. The codebook entry with the highest similarity represents the answer to the query. This will be *Peso* with a high probability for large N. The probability of the correct answer can be understood through the capacity theory of distributed representations described in [14], which we next apply to this context.

In general, a vector like $\mathbf{t}_{\mathrm{UM}}$ can be considered as a mapping between the fields in the two tables. The number of entries will determine the amount of crosstalk noise, but all of the entries that are nonsensible also are considered crosstalk noise.

Specifically, we consider general data records of key-value pairs, similar in form to **ustates** and **mexico**. These data records will contain R "role" vectors that act as keys. Each one has corresponding M_r potential "filler" values. The role vectors are stored in a codebook $\mathbf{\Psi} \in \mathbb{C}^{N \times R}$. For simplicity, we assume that all R roles are present in a data record, each with one of the M_r fillers attached. The fillers for each role are stored in the codebook $\mathbf{\Phi}^{(r)} \in \mathbb{C}^{N \times M_r}$. This yields a generic key-value data record

$$\mathbf{rec} = \sum_{r}^{R} \mathbf{\Psi}_r *_b \mathbf{\Phi}^{(r)}_{i^*} \tag{21}$$

where the index i^* indicates one filler vector from the codebook for a particular role.

Next, we form the transformation vector, which is used to map one data record to another. This is done generically by binding two record vectors: $\mathbf{t}_{ij} = \mathbf{rec}_j *_b \mathbf{rec}_i^{-1}$.

As discussed, the terms in each record will distribute, and the values that share the same roles will be associated with each other. But, there are many cross-terms that are also present in the transformation vector that is not useful for any analogical reasoning query. The crosstalk noise is dependent on how many terms are present in the sum, and this includes the cross-terms. Thus, the total number of terms in the transformation vector $\mathbf{t}_{ij}$ will be R^2.

In the next step, a particular filler is queried and the result is decoded by comparison to the codebook $\mathbf{\Phi}$, which contains the sparse block-code of each possible filler

$$\mathbf{a}_r = \mathbf{\Phi}^{(r)}\left(\mathbf{t}_{ij} *_b \mathbf{\Phi}^{(r)}_{j^*}\right) \tag{22}$$

where j^* indicates the index of the filler in the query (e.g., the index of *Dollar*). The entry with the largest amplitude in the vector $\mathbf{a}_r$ is considered the output.

The probability that this inference finds the correct relationship can be predicted by the VSA capacity analysis [14] (Fig. 8). The probability is a function of the signal-to-noise ratio, given in this case by $s^2 = N/R^2$. Fig. 8 shows that

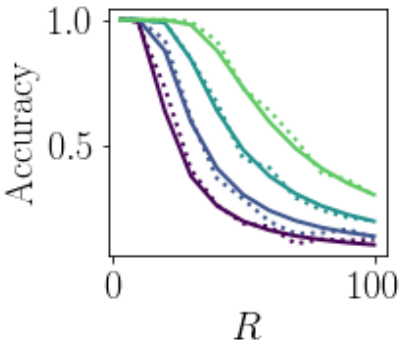

Fig. 8. Performance of analogic reasoning tasks with sparse block-codes. We empirically simulated analogic reasoning tasks with data records containing R key-value pairs, and measured the performance (dashed lines, $N = [1000, 2000, 4000, 8000]$ dark to light). This performance can be predicted based on the VSA capacity theory reported in [14] (solid lines).

the performance of the sparse block-code VSA matches the predictions of the general VSA capacity theory, meaning that sparse block-codes are equivalent to standard VSA representations in representation capacity.

B. Solving Classification Problems

Although VSAs originated as models for symbolic reasoning, recent applications of VSA in machine learning solve classification problems, for example, in language identification [49], [50], gesture recognition [51], [52], prediction of mobile phone use patterns [53], and fault identification [54]. Furthermore, it was pointed out in [55], that VSAs can describe classifiers using randomly connected feed-forward neural networks [56], referred to as random vector functional link (RVFL) [57] or extreme learning machines (ELM) [58]. Specifically, RVFL/ELM can be expressed by VSA operations in the MAP VSA model [55]. Leveraging these insights, we implemented a classification model using a VSA with sparse block-codes.

The model proposed in [55] forms a dense distributed representation $\mathbf{x}$ of a set of features $\mathbf{a}$. Each feature is assigned a random "key" vector $\mathbf{\Phi}_i \in \{\pm 1\}^N$. The collection of "key" vectors constitutes the codebook $\mathbf{\Phi}$. However, in contrast to (3) the set of features is represented differently. The proposed approach requires the mapping of a feature value a_i to distributed representation $\mathbf{F}_i$ ("value") which preserve the similarity between nearby scalars. [55] used thermometric encoding [59] to create such similarity preserving distributed representations. The feature set is represented as the sum of "key"-"value" pairs using the binding operation

$$\mathbf{x} = f_\kappa\left(\sum_i^M \mathbf{\Phi}_i \odot \mathbf{F}_i\right) \tag{23}$$

where f_κ denotes the clipping function which is used as a nonlinear activation function

$$f_\kappa(x_i) = \begin{cases} -\kappa & x_i \le -\kappa \\ x_i & -\kappa < x_i < \kappa \\ \kappa & x_i \ge \kappa. \end{cases} \tag{24}$$

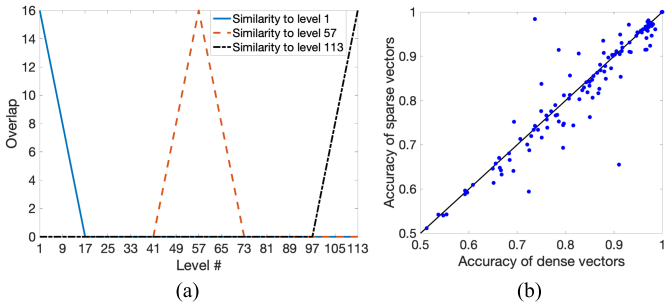

Fig. 9. Solving classification problems with sparse block-codes. (a) Similarity preserving representation of scalars with sparse block-codes: $K = 16$, $N = 128$. Shown are similarity (overlap) between the representations of the levels 1, 57, and 113 and the vectors representing other signal levels. (b) Cross-validation accuracy of the VSA with dense distributed representations against the VSA with sparse distributed representations. A point corresponds to a dataset.

The clipping function is characterized by the configurable threshold parameter κ regulating nonlinear behavior of the neurons and limiting the range of activation values.

The predicted class $\hat{y}$ is read out from $\mathbf{x}$ using the trainable readout matrix as

$$\hat{y} = \text{argmax}\mathbf{W}^{\text{out}}\mathbf{x} \tag{25}$$

where $\mathbf{W}^{\text{out}}$ is obtained via the ridge regression applied to a training dataset.

Thermometric codes are nonsparse and their mean activity is variable across different values. Building on earlier efforts in the design of similarity-preserving sparse coding [60], [61], we design a similarity-preserving encoding scheme with sparse block-codes. In this scheme, the lowest signal level has all hot components in the first positions of each block. The second signal level is encoded by the same pattern except that the hot component of the first block is shifted to the second position. The third signal level is encoded by the code of the second level with the hot component of the second block shifted to the second position, and so on. This feature encoding scheme can represent $N - K + 1$ signal levels uniquely. The similarity between vectors drops of gradually as a function of distance [Fig. 9(a)]. Each pattern has the highest similarity with itself (overlap $= K$). For the range of distances between 1 and K, the overlap decreases linearly until it reaches 0 and then stays at this level for larger distances.

The data vectors in a classification problem are essentially encoded as protected sums (5). First, labels of the different features (data dimensions) are encoded by random sparse block-code vectors. Key-value pairs are then formed by binding feature labels with corresponding values, using the similarity preserving sparse block-code scheme described above. A data vector is then represented by the sum of all the key-value pairs. In addition, we apply a clipping function to the resulting input.

The described representation of the data can be computed in a sparse block-code VSA, the last step can be represented by the activation of a hidden layer with nonlinear neurons. To perform classification, the hidden representation is pattern-matched to prototypes of the different classes. To optimize this pattern matching, in cases where the prototypes are correlated, we train a perceptron network with ridge

regression, similar as previously proposed for in a sequence memory with VSAs [14].

The 121 datasets from the UCI Machine Learning Repository [62] used in the experiments have been initially analyzed in a large-scale comparison study of different classifiers [63]. The only preprocessing step we introduced was to normalize features in the range $[0, 1]$ and quantize the values into $N - K + 1$ levels. The hyperparameters for both dense and sparse models were optimized through grid search over N (for dense representations N varied in the range $[50, 1500]$ with step 50), λ (ridge regression regularization parameter; varied in the range $2^{[-10,5]}$ with step 1), and κ (varied between $\{1, 3, 7, 15\}$). The search additionally considered K for sparse block-codes (K/N varied in the range $2^{[2,5]}$ with step 1 while K varied in the range $2^{[4,7]}$ with step 1).

Interestingly, the cross-validated accuracies for the VSA with sparse block-codes and the VSA with dense representations [55] on the UCI collection are quite similar [Fig. 9(b)], with a correlation coefficient of 0.88 and both reaching average accuracy of 0.80.[3] Importantly, the number of neurons used by both approaches was also of the same order of magnitude: about 500 for sparse block-codes and about 750 for dense representations. Thus, we conclude that sparse block-codes can be used as substitutes for dense representations for practical problems such as classifications tasks.

VI. Discussion

In this article, we investigated methods for variable binding for symbolic reasoning with sparse distributed representations. The motivation for this study was twofold. First, we believe that such methods of variable binding could be key for combining the universal reasoning properties of VSAs [12] with advantages of neural networks. Second, these methods will enable implementations of symbolic reasoning that can leverage efficient sparse Hebbian associative memories [15], [24], [36] and low-power neuromorphic hardware [64].

A. Theoretical Results

Using the framework of CS, we investigated a setting in which there is a unique equivalence between sparse feature vectors and dense random vectors. We find the following.

1) With this setting, CS inference outperforms the classical VSA readout of set representations.
2) Classical vector symbolic binding between dense vectors with the Hadamard product [1], [7], [8] is under certain conditions mathematically equivalent to tensor product binding [3] of the corresponding sparse vectors.
3) For representing sets of objects, vector addition of dense vectors (15) is equivalent to the addition of the corresponding sparse vectors.
4) The protected sum of dense vectors (16) is equivalent to the concatenation of the sparse vectors.
5) The dimensionality preserving operations between dense vectors for variable binding and protected set representations mathematically correspond to operations between sparse vectors, tensor product, and vector concatenation, which are not dimensionality preserving.

[3]In the Supplementary Materials (Section III), we additionally compare the results for the VSA with sparse block-codes to the results of another model with sparse representations—binarized random projections.

B. Experimental Results

Our theory result 5) implies that in order to construct dimensionality and sparsity-preserving variable binding between sparse vectors, an additional reduction step is required for mapping the outer product to a sparse vector. Existing reduction schemes of the outer product proposed in the literature, circular convolution [1] and VDTB [42], are not sparsity-preserving when applied to sparse vectors.

For binding pairs of general K-sparse vectors, we designed a strategy of subsampling from the outer-product with additional thresholding to maintain sparsity. Such a computation can be implemented in neural circuitry where dendrites of neurons detect firing coincidences between pairs of input neurons. The necessary connection density increases with sparsity of the code vectors. Still, the computational complexity is order N when $K = \beta N$, which favorably compares to other binding operations which can have an order of N^2 or $N \log N$. However, the sampling in the circuit always misses components of the tensor product, making the unbinding operation lossy.

Another direction we investigated extends previous work [2] developing VSAs for sparse representations of restricted type, sparse block-codes. We propose block-LCC as a variable binding method that is sparsity and dimensionality preserving. Interestingly, for sparse block-codes, the unbinding given the reduced tensor and one of the factors is lossless. As our experiments show, it has the desired properties required for VSA manipulations, outperforming the other methods. Independent other work has proposed efficient Hebbian associative memory models [35], [36], [65] that could be applied for cleanup steps required in VSAs with block-codes.

VSAs with block-codes are demonstrated in two applications. In a symbolic reasoning application, we show that the accuracy as a function of the dimension of sparse block-codes reaches the full performance of dense VSAs and can be described by the same theory [14]. On 121 classification datasets from the UCI Machine Learning Repository we show that the block-code VSA reaches the same performance as dense VSAs [55]. Moreover, the average accuracy of 0.80 of VSAs models is comparable to the state-of-the-art performance of 0.82 achieved by Random Forest [63].

C. Relationship to Earlier Work

The papers [20], [66] were to our knowledge the first to propose dimensionality- and sparsity-preserving variable binding. For binary representations they proposed methods that involve componentwise Boolean operations and deletion (thinning) based on random permutations. These methods of variable binding are also lossy, similar to our method of SPTP.

The variable binding with block-codes, which our experiments identify as the best, can be done with real-valued binary or complex-valued phasor block-codes. For binary block-codes our binding method is the same as in [2], who demonstrated it in a task processing symbolic sequences. For protecting

individual elements in a sum representation, they use random permutations between blocks, rather than variable binding as we do in Section III-C.

D. Implications for Neural Networks and Machine Learning

In the deep network literature, concatenation is often used in neural network models as a variable binding operation [67]. However, our result 4) suggests that concatenation is fundamentally different from a binding operation. This might be a reason why deep learning methods have limited capabilities to represent and manipulate data structures [68].

Several recent studies have applied VSAs to classification problems [69], [70]. Here, we demonstrated classification in a block-code VSA. The block-code VSA exhibited the same average classification accuracy as earlier VSA solutions with dense codes. This result suggests that sparse block-code VSAs can be a promising basis for developing classification algorithms for low-power neuromorphic hardware platforms [64]. Finally, it is worth mentioning that unrestricted sparse distributed representations in particular and sparsity in general have been applied in many contexts within neural networks and machine learning such as associative memories [15], [24], [31], [34], [71], [72], phasor networks [32], CS [39], [40], approximate nearest neighbor search [73]–[76], training of sparse models [77]–[79] to name a few. Studying the potential use of sparse block-code VSA in these scenarios is an interesting future direction.

E. Implications for Neuroscience

We have investigated variable binding operations between sparse patterns regarding their computational properties in symbolic reasoning. It is interesting that this form of variable binding requires multiplication or coincidence detection, computations which can be implemented by active dendritic mechanisms of biological neurons [80]. Although this computation is beyond the capabilities of standard neural networks, it can be implemented with formal models of neurons, such as sigma-pi neurons [81].

We found that the most efficient form of variable binding with sparse vectors relies on block-code structure. Although block-codes were engineered independent of neurobiology, they compatible with some experimental observations, such as divisive normalization [37], and functional modularity. Specifically, in sensory cortices of carnivores neurons within small cortical columns [82] respond to the same stimulus features, such as the orientation of local edges in the image [38], [83]. Furthermore, groups of nearby orientation columns form so-called macro columns, tiling all possible edge orientations at a specific image location [84], [85]. A macro column may correspond to a block in a block-code.

While binary block-codes are not biologically plausible, complex-valued block-codes in which active elements are complex phasors with unit magnitude, can be represented as timing patterns in networks of spiking neurons [32]. Furthermore, it seems possible to extend LCC binding to soft block-codes, in which localized bumps with graded neural activities represented by spike rate, e.g., [86].

F. Future Directions

One important future direction is to investigate how to combine the advantages of VSA and traditional neural networks to build more powerful tools for artificial intelligence. The challenge is how to design neural networks for learning sparse representations that can be processed in sparse VSAs. Such combined systems could potentially overcome some of the severe limitations of current neural networks, such as the demand for large amounts of data, limited abilities to generalize learned knowledge, etc.

Another interesting research direction is to design VSAs operating with spatio-temporal spike patterns that can be implemented in neuromorphic hardware, potentially also making use spike timing and efficient associative memory for spike timing patterns [32].

Furthermore, it will be interesting to study how binding in sparse VSAs can be used to form similarity-preserving sparse codes [60], [61] for continuous manifolds. For example, binding can be used to create index patterns for representing locations in space, which could be useful for navigation in normative modeling of hippocampus [87].

ACKNOWLEDGMENT

The authors thank Charles Garfinkle and other members of the Redwood Center for Theoretical Neuroscience for stimulating discussions.

REFERENCES

[1] T. A. Plate, *Holographic Reduced Representation: Distributed Representation for Cognitive Structures*. Stanford, CA, USA: CSLI Publications, 2003.

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

[3] P. Smolensky, "Tensor product variable binding and the representation of symbolic structures in connectionist systems," *Artif. Intell.*, vol. 46, nos. 1–2, pp. 159–216, Nov. 1990.

[4] J. A. Fodor and Z. W. Pylyshyn, "Connectionism and cognitive architecture: A critical analysis," *Cognition*, vol. 28, nos. 1–2, pp. 3–71, Mar. 1988.

[5] T. A. Plate, "Distributed representations and nested compositional structure," Ph.D. dissertation, Graduate Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, 1994.

[6] P. Kanerva, "Binary spatter-coding of ordered K-tuples," in *Artificial Neural Networks* (Lecture Notes in Computer Science), vol. 1112. Berlin, Germany: Springer, 1996, pp. 869–873.

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

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

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

[10] T. Yerxa, A. Anderson, and E. Weiss, "The hyperdimensional stack machine," in *Proc. Cogn. Comput., Merging Concepts Hardware*, Hannover, Germany, Dec. 2018, pp. 1–2.

[11] D. Kleyko *et al.*, "Vector symbolic architectures as a computing framework for nanoscale hardware," 2021, *arXiv:2106.05268*. [Online]. Available: https://arxiv.org/abs/2106.05268

[12] R. W. Gayler, "Vector symbolic architectures answer Jackendoff's challenges for cognitive neuroscience," in *Proc. ICCS/ASCS Int. Conf. Cognit. Sci.*, 2003, p. 6.

[13] M. Ledoux, *The Concentration of Measure Phenomenon*, no. 89. American Mathematical Society, 2001,

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

[15] D. J. Willshaw, O. P. Buneman, and H. C. Longuet-Higgins, "Non-holographic associative memory," *Nature*, vol. 222, no. 5197, pp. 960–962, Jun. 1969.

[16] B. A. Olshausen and D. J. Field, "Natural image statistics and efficient coding," *Netw., Comput. Neural Syst.*, vol. 7, no. 2, p. 333, May 1996.

[17] A. Treisman, "Feature binding, attention and object perception," *Phil. Trans. Roy. Soc. London. B, Biol. Sci.*, vol. 353, no. 1373, pp. 1295–1306, Aug. 1998.

[18] G. E. Hinton, J. L. McClelland, and D. E. Rumelhart, "Distributed representations," in *The Philosophy of Artificial Intelligence*. New York, NY, USA: Oxford Univ. Press, 1990, pp. 248–280.

[19] T. A. Plate, "Holographic recurrent networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 5, no. 1, 1993, pp. 34–41.

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

[21] G. E. Hinton, "Mapping part-whole hierarchies into connectionist networks," *Artif. Intell.*, vol. 46, pp. 47–76, Nov. 1990.

[22] T. A. Plate, "Holographic reduced representations: Convolution algebra for compositional distributed representations," in *Proc. 12th Int. Joint Conf. Artif. Intell.*, 1991, pp. 30–35.

[23] T. A. Plate, "Holographic reduced representations," *IEEE Trans. Neural Netw.*, vol. 6, no. 3, pp. 623–641, May 1995.

[24] G. Palm, "On associative memory," *Biol. Cybern.*, vol. 36, no. 1, pp. 19–31, 1980.

[25] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci. USA*, vol. 79, no. 8, pp. 2554–2558, Apr. 1982.

[26] P. Lennie, "The cost of cortical computation," *Current Biol.*, vol. 13, no. 6, pp. 493–497, Mar. 2003.

[27] S. B. Laughlin and T. J. Sejnowski, "Communication in neuronal networks," *Science*, vol. 301, no. 5641, pp. 1870–1874, Sep. 2003.

[28] A. J. Bell and T. J. Sejnowski, "The 'independent components' of natural scenes are edge filters," *Vis. Res.*, vol. 37, no. 23, pp. 3327–3338, Dec. 1997.

[29] M. Rehn and F. T. Sommer, "A network that uses few active neurones to code visual input predicts the diverse shapes of cortical receptive fields," *J. Comput. Neurosci.*, vol. 22, no. 2, pp. 135–146, Feb. 2007.

[30] M. V. Tsodyks and M. V. Feigel'man, "The enhanced storage capacity in neural networks with low activity level," *Europhys. Lett.*, vol. 6, no. 2, pp. 101–105, May 1988.

[31] G. Palm and F. T. Sommer, "Information capacity in recurrent McCulloch–Pitts networks with sparsely coded memory states," *Netw., Comput. Neural Syst.*, vol. 3, no. 2, pp. 177–186, Jan. 1992.

[32] E. P. Frady and F. T. Sommer, "Robust computation with rhythmic spike patterns," *Proc. Nat. Acad. Sci. USA*, vol. 116, no. 36, pp. 18050–18059, Sep. 2019.

[33] Y. C. Eldar, P. Kuppinger, and H. Bolcskei, "Block-sparse signals: Uncertainty relations and efficient recovery," *IEEE Trans. Signal Process.*, vol. 58, no. 6, pp. 3042–3054, Jun. 2010.

[34] V. I. Gritsenko, D. A. Rachkovskij, A. A. Frolov, R. Gayler, D. Kleyko, and E. Osipov, "Neural distributed autoassociative memories: A survey," *Cybern. Comput. Eng.*, vol. 2, no. 188, pp. 5–35, 2017.

[35] V. Gripon and C. Berrou, "Sparse neural networks with large learning diversity," *IEEE Trans. Neural Netw.*, vol. 22, no. 7, pp. 1087–1096, Jul. 2011.

[36] A. Knoblauch and G. Palm, "Iterative retrieval and block coding in autoassociative and heteroassociative memory," *Neural Comput.*, vol. 32, no. 1, pp. 205–260, Jan. 2020.

[37] D. J. Heeger, "Normalization of cell responses in cat striate cortex," *J. Neurosci.*, vol. 9, no. 2, pp. 181–197, 1992.

[38] D. H. Hubel and T. N. Wiesel, "Ferrier lecture-functional architecture of macaque monkey visual cortex," *Proc. Roy. Soc. London. B, Biol. Sci.*, vol. 198, no. 1130, pp. 1–59, 1977.

[39] E. J. Candés, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Commun. Pure Appl. Math.*, vol. 59, no. 8, pp. 1207–1223, 2006.

[40] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[41] C. J. Hillar and F. T. Sommer, "When can dictionary learning uniquely recover sparse data from subsamples?" *IEEE Trans. Inf. Theory*, vol. 61, no. 11, pp. 6290–6297, Nov. 2015.

[42] J. Gosmann and C. Eliasmith, "Vector-derived transformation binding: An improved binding operation for deep symbol-like processing in neural networks," *Neural Comput.*, vol. 31, no. 5, pp. 849–869, May 2019.

[43] A. Amini and F. Marvasti, "Deterministic construction of binary, bipolar, and ternary compressed sensing matrices," *IEEE Trans. Inf. Theory*, vol. 57, no. 4, pp. 2360–2370, Apr. 2011.

[44] S. Ganguli and H. Sompolinsky, "Statistical mechanics of compressed sensing," *Phys. Rev. Lett.*, vol. 104, no. 18, pp. 1–4, 2010.

[45] D. Kleyko, A. Rosato, E. P. Frady, M. Panella, and F. T. Sommer, "Perceptron theory for predicting the accuracy of neural networks," 2020, *arXiv:2012.07881*. [Online]. Available: https://arxiv.org/abs/2012.07881

[46] A. Thomas, S. Dasgupta, and T. Rosing, "Theoretical foundations of hyperdimensional computing," 2020, *arXiv:2010.07426*. [Online]. Available: https://arxiv.org/abs/2010.07426

[47] P. Smolensky, M. Lee, X. He, W.-T. Yih, J. Gao, and L. Deng, "Basic reasoning with tensor product representations," 2016, *arXiv:1601.02745*. [Online]. Available: https://arxiv.org/abs/1601.02745

[48] P. Kanerva, "What we mean when we say, 'what's the dollar of Mexico?': Prototypes and mapping in concept space," in *Proc. AAAI Fall Symp., Quantum Informat. Cognit., Social, Semantic Processes*, 2010, pp. 2–6.

[49] A. Joshi, J. T. Halseth, and P. Kanerva, "Language geometry using random indexing," in *Proc. Int. Symp. Quantum Interact.* Cham, Switzerland: Springer, 2016, pp. 265–274.

[50] D. Kleyko, E. Osipov, D. D. Silva, U. Wiklund, V. Vyatkin, and D. Alahakoon, "Distributed representation of n-gram statistics for boosting self-organizing maps with hyperdimensional computing," in *Proc. Int. Andrei Ershov Memorial Conf. Perspect. Syst. Informat. (PSI)* (Lecture Notes in Computer Science), vol. 11964. Cham, Switzerland: Springer, 2019, pp. 64–79.

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

[52] D. Kleyko, A. Rahimi, D. A. Rachkovskij, E. Osipov, and J. M. Rabaey, "Classification and recall with binary hyperdimensional computing: Tradeoffs in choice of density and mapping characteristics," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 12, pp. 5880–5898, Dec. 2018.

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

[54] D. Kleyko, E. Osipov, N. Papakonstantinou, and V. Vyatkin, "Hyperdimensional computing in industrial systems: The use-case of distributed fault isolation in a power plant," *IEEE Access*, vol. 6, pp. 30766–30777, 2018.

[55] D. Kleyko, M. Kheffache, E. P. Frady, U. Wiklund, and E. Osipov, "Density encoding enables resource-efficient randomly connected neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 8, pp. 3777–3783, Aug. 2021.

[56] S. Scardapane and D. Wang, "Randomness in neural networks: An overview," *Data Mining Know. Discovery*, vol. 7, pp. 1–18, Mar. 2017.

[57] B. Igelnik and Y.-H. Pao, "Stochastic choice of basis functions in adaptive function approximation and the functional-link net," *IEEE Trans. Neural Netw.*, vol. 6, no. 6, pp. 1320–1329, Nov. 1995.

[58] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, nos. 1–3, pp. 489–501, Dec. 2006.

[59] D. A. Rachkovskij, S. V. Slipchenko, E. M. Kussul, and T. N. Baidyk, "Sparse binary distributed encoding of scalars," *J. Autom. Inf. Sci.*, vol. 37, no. 6, pp. 12–23, 2005.

[60] G. Palm, F. Schwenker, and F. T. Sommer, "Associative memory networks and sparse similarity preserving codes," in *From Statistics to Neural Networks*. Berlin, Germany: Springer, 1994, pp. 282–302.

[61] G. Palm, "Neural associative memories and sparse coding," *Neural Netw.*, vol. 37, no. 1, pp. 165–171, Jan. 2013.

[62] D. Dua and C. Graff. (2019). *UCI Machine Learning Repository*. [Online]. Available: https://archive.ics.uci.edu/ml

[63] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?" *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3133–3181, 2014.

[64] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan. 2018.

[65] I. Kanter, "Potts-glass models of neural networks," *Phys. Rev. A, Gen. Phys.*, vol. 37, no. 7, pp. 2739–2742, Apr. 1988.
[66] D. A. Rachkovskij and E. M. Kussul, "Binding and normalization of binary sparse distributed representations by context-dependent thinning," *Neural Comput.*, vol. 13, no. 2, pp. 411–452, 2001.
[67] M. Soll, T. Hinz, S. Magg, and S. Wermter, "Evaluating defensive distillation for defending text processing neural networks against adversarial examples," in *Proc. Int. Conf. Artif. Neural Netw. (ICANN)*. Cham, Switzerland: Springer, 2019, pp. 685–696.
[68] G. Marcus, "The next decade in AI: Four steps towards robust artificial intelligence," 2020, *arXiv:2002.06177*. [Online]. Available: https://arxiv.org/abs/2002.06177
[69] L. Ge and K. K. Parhi, "Classification using hyperdimensional computing: A review," *IEEE Circuits Syst. Mag.*, vol. 20, no. 2, pp. 30–47, Jun. 2020.
[70] A. Rahimi, P. Kanerva, L. Benini, and J. M. Rabaey, "Efficient biosignal processing using hyperdimensional computing: Network templates for combined learning and classification of ExG signals," *Proc. IEEE*, vol. 107, no. 1, pp. 123–143, Jan. 2019.
[71] P. Kanerva, *Sparse Distributed Memory*. Cambridge, MA, USA: MIT Press, 1988.
[72] F. T. Sommer and P. Dayan, "Bayesian retrieval in associative memories with storage errors," *IEEE Trans. Neural Netw.*, vol. 9, no. 4, pp. 705–713, Jul. 1998.
[73] V. Hyvonen *et al.*, "Fast nearest neighbor search through sparse random projections and voting," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2016, pp. 881–888.
[74] S. Dasgupa and C. F. Stevens, "A neural algorithm for a fundamental computing problem," *Science*, vol. 358, no. 6364, pp. 793–796, Nov. 2017.
[75] J. Gou, Y. Xu, D. Zhang, Q. Mao, L. Du, and Y. Zhan, "Two-phase linear reconstruction measure-based classification for face recognition," *Inf. Sci.*, vols. 433–434, pp. 17–36, Apr. 2018.
[76] E. P. Frady *et al.*, "Neuromorphic nearest neighbor search using Intel's Pohoiki springs," in *Proc. Neuro-Inspired Comput. Elements Workshop*, Mar. 2020, pp. 1–10.
[77] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Statist. Soc., B, Methodol.*, vol. 58, no. 1, pp. 267–288, 1996.
[78] J. Gou, L. Wang, Z. Yi, Y. Yuan, W. Ou, and Q. Mao, "Weighted discriminative collaborative competitive representation for robust image classification," *Neural Netw.*, vol. 125, pp. 104–120, May 2020.
[79] J. Gordon, D. Rawlinson, and S. Ahmad, "Long distance relationships without time travel: Boosting the performance of a sparse predictive autoencoder in sequence modeling," in *Proc. IAPR Workshop Artif. Neural Netw. Pattern Recognit. (ANNPR)*, 2020, pp. 52–64.
[80] M. E. Larkum and T. Nevian, "Synaptic clustering by dendritic signalling mechanisms," *Current Opinion Neurobiol.*, vol. 18, no. 3, pp. 321–331, Jun. 2008.
[81] B. W. Mel and C. Koch, "Sigma-pi learning: On radial basis functions and cortical associative learning," in *Proc. Adv. neural Inf. Process. Syst.*, 1990, pp. 474–481.
[82] V. B. Mountcastle, "Modality and topographic properties of single neurons of cat's somatic sensory cortex," *J. Neurophysiol.*, vol. 20, no. 4, pp. 408–434, Jul. 1957.
[83] D. H. Hubel and T. N. Wiesel, "Shape and arrangement of columns in cat's striate cortex," *J. Physiol.*, vol. 165, no. 3, pp. 559–568, Mar. 1963.
[84] D. H. Hubel and T. N. Wiesel, "Sequence regularity and geometry of orientation columns in the monkey striate cortex," *J. Comparative Neurol.*, vol. 158, no. 3, pp. 267–293, Dec. 1974.
[85] N. V. Swindale, "Is the cerebral cortex modular?" *Trends Neurosciences*, vol. 13, no. 12, pp. 487–492, Dec. 1990.
[86] R. B. Yishai, R. L. Bar-Or, and H. Sompolinsky, "Theory of orientation tuning in visual cortex," *Proc. Nat. Acad. Sci. USA*, vol. 92, no. 9, pp. 3844–3848, 1995.
[87] E. P. Frady and F. T. Sommer, "A normative hippocampus model: Robustly encoding variables on smooth manifolds using spiking neurons," in *Proc. CoSyNe*, 2020, p. 221.

Edward Paxon Frady received the B.S. degree in computation and neural systems from California Institute of Technology, Pasadena, CA, USA, in 2008 and the Ph.D. degree in neuroscience from the University of California San Diego, La Jolla, CA, USA, in 2014.

He is currently a Researcher in Residence with the Neuromorphic Computing Lab, Intel Labs, Santa Clara, CA, and a Visiting Scholar with the Redwood Center for Theoretical Neuroscience at the University of California, Berkeley, CA. His research interests include neuromorphic engineering, vector symbolic architectures/hyperdimensional computing, and machine learning.

Denis Kleyko (Member, IEEE) received the B.S. degree (Hons.) in telecommunication systems and the M.S. degree (Hons.) in information systems from Siberian State University of Telecommunications and Information Sciences, Novosibirsk, Russia, in 2011 and 2013, respectively, and the Ph.D. degree in computer science from Luleå University of Technology, Luleå, Sweden, in 2018.

He is currently a Post-Doctoral Researcher on a joint appointment between the Redwood Center for Theoretical Neuroscience at the University of California, Berkeley, CA, USA, and the Intelligent Systems Lab, Research Institutes of Sweden, Kista, Sweden. His current research interests include machine learning, reservoir computing, and vector symbolic architectures/hyperdimensional computing.

Friedrich T. Sommer received the Diploma degree in physics from the University of Tuebingen, Tuebingen, Germany, in 1987 and the Ph.D. degree from the University of Duesseldorf, Duesseldorf, Germany, in 1993. He received his habilitation in computer science from the University of Ulm, Ulm, Germany, in 2002.

He is currently a Researcher in Residence with the Neuromorphic Computing Lab, Intel Labs, Santa Clara, CA, USA, and an Adjunct Professor with the Redwood Center for Theoretical Neuroscience at the University of California, Berkeley, CA. His research interests include neuromorphic engineering, vector symbolic architectures/hyperdimensional computing, and machine learning.