Decoding Superpositions of Bound Symbols Represented by Distributed Representations

Michael Hersche[1,2], Zuzanna Opala[1], Geethan Karunaratne[1,2], Abu Sebastian[1] and Abbas Rahimi[1,*]

[1]IBM Research-Zurich, Säumerstrasse, 4, 8803 Rüschlikon, Switzerland
[2]ETH Zürich, Rämistrasse 101, 8092 Zürich, Switzerland

Abstract

Vector-symbolic architectures (VSAs) express data structures with an arbitrary complexity and perform symbolic computations on them by exploiting high-dimensional distributed representations and associated key operations. VSAs typically use dense random vectors, aka hypervectors, to represent atomic symbols that can be combined into compound symbols by multiplicative binding and additive superposition operators. For instance, a VSA-based neural encoder can bind two atomic symbols, and further superpose a set of such bound symbols—all by distributed vectors that have the same dimension. Nevertheless, decoding such an additive-multiplicative vector, to the atomic symbols from which it is built, is not a trivial task. Recently, a solution based on resonator networks was proposed to iteratively factorize one of the bound symbols. After finding the factorization, it is explained away by subtracting it from the superposition. This explaining away, however, causes noise amplification that limits the number of symbols that can be reliably decoded in large problem sizes. Here, we present novel methods that efficiently decode VSA-based data structures consisting of multiplicative binding and additive superposition of symbols. We expand the pure sequential explaining away approach by performing multiple decodings in parallel using a dedicated query sampler. Compared to the baseline resonator network, this mix of sequential and parallel decoding retrieves up to 8× more additive components from larger problems in synthetic and real-world experiments.

Keywords

Vector-symbolic architectures, hyperdimensional computing, resonator networks, decoding neural structures, explaining away

1. Introduction

Vector-symbolic architectures (VSAs) [1, 2, 3, 4, 5] are a family of computational models that create an elegant formalism to encode, manipulate, and importantly bind symbols while keeping the size of the distributed representation fixed, regardless of whether they represent one single entity or a nested structure of multiple entities. VSAs achieve it by encoding base codewords as high-dimensional random vectors, aka hypervectors. As such, they represent data in a distributed manner and often serve as a mediator between the rule-based symbolic reasoning and connectionist models that include neural networks. Recent work [6] has shown how VSA,

Siena'22: NeSy 2023, 17th international workshop in Neural-Symbolyc Learning and Reasoning, July 03–05, 2022, Siena, Italy
*Corresponding author.
abr@zurich.ibm.com (A. Rahimi)

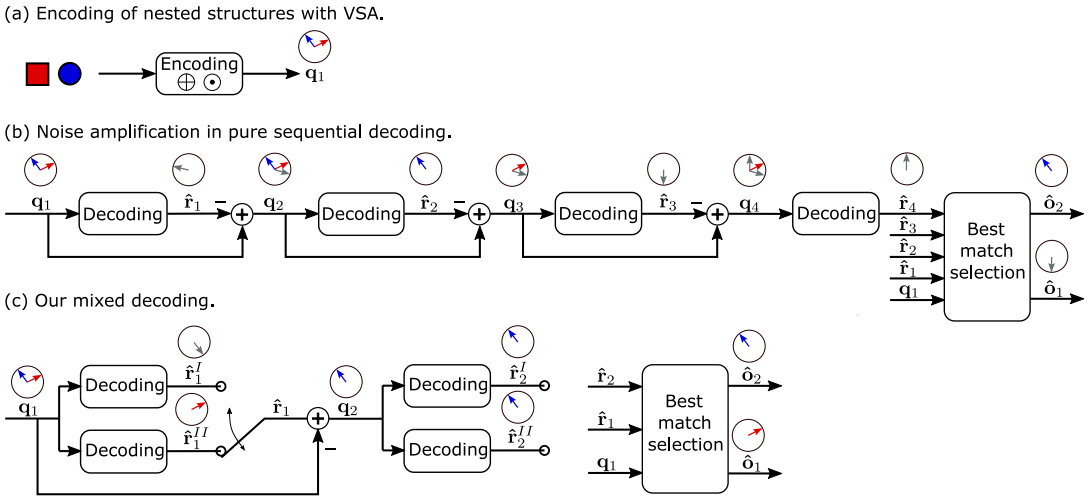

Figure 1: (a) We encode multiple objects as the superposition of the individual high-dimensional symbol binding. **(b)** Commonly used pure sequential explaining away decoding with resonator networks results in noise amplification (indicated with grey arrows). (**c**) Our mixed decoding performs multiple decodings in parallel which allows finding the correct solution.

as a common language between neural networks and symbolic AI, can overcome the neural network binding problem and exhaustive symbolic search.

While VSAs efficiently encode single or multiple objects into complex structures, decoding is often challenging. Let us focus on two core operations used by VSAs. Additive superposition ($\oplus$), to which we will refer in short as a superposition, creates a sum vector similar to all of the input vectors. In contrast, multiplicative binding ($\odot$), or simply binding, produces a product vector dissimilar to its factors. To check whether a given query vector is available in the sum vector, we can calculate the similarity between the sum vector and the query. For binding, however, we would need to check all the possible combinations of factors. This makes the decoding of bound symbols a hard combinatorial problem [7].

An elegant solution is resonator networks [7, 8] which iteratively decode the binding by exploring only a fragment of the exponential search space. The resonator network also demonstrated to decode the superposition of more than one bound symbol by repeatedly invoking the resonator network decoding and applying an explaining away operation (subtraction) of the result from the query. There were, however, two limitations with the resonator network. First, its dynamical system suffers from a relatively low operational capacity (i.e., the low ratio between the exponential search space and the required vector dimensionality to represent bound symbols). This is mainly due to limit cycles, deterministic computation, and lack of nonlinearity during each iteration. A stochastic sparsely-activated resonator [9] recently addressed this limitation by exploiting the stochasticity of memristive devices and by proposing nonlinear sparse activations. These new dynamics enhance the operational capacity by five orders of magnitude and reduce the spatial and time complexity associated with the factorization task.

The second limitation of the resonator network is the explanation away decoding, which we address in this work (see Fig. 1). More precisely, when decoding the superposition by explaining

away the resonator's product estimates, a mistake is probable with more added bound symbols. It is because a query that contains several superposed vectors can be seen as noisy—each added term can be interpreted as noise from the standpoint of decoding the other. After decoding the wrong bound symbols, the resonator network runs into a noise amplification problem: subtracting bound symbols that were not present adds more noise to the query, making it even harder to decode the remaining bound symbols.

In this work, we present a configurable decoding method to efficiently extract a set of bound symbols—each represented by a Hadamard product distributed vector—from their fixed-width superposition. As such, we detail how naive sequential decoding can be enhanced by generating multiple queries in parallel by means of a sampling mechanism (see Fig. 1c). By combining sequential and parallel decoding methods, our mixed decoding approach mitigates the risk of noise amplification and increases the number of bound symbols that can be successfully decoded by up to $8\times$ at the same vector dimensionality.

2. Background

2.1. Vector Symbolic Architectures (VSAs)

Vector symbolic architectures (VSAs) represent base entities as random high-dimensional vectors. There are different VSA variants (see [10, 11] for a review). This work focuses on the Multiply-Add-Permute (MAP) coding scheme [2] where vectors are bipolar: $\mathbf{x} \in \{-1, 1\}^D$. The similarity of two vectors is defined as their cosine similarity. Usually, we assume that the code vectors are generated randomly and stored in a lookup memory, often called codebook. The algebra on such vectors is defined using three basic operations. Multiplicative binding ($\odot$) is the elementwise multiplication (i.e., the Hadamard product) of vectors. The resulting vector is dissimilar to the bound factors. The inverse operation is the unbinding, which is also the elementwise multiplication in the MAP architecture. Bundling ($\oplus$) is the additive superposition of a set of vectors. Optionally, the result can be biploarized by setting positive values to "+1" and negative ones to "-1"; zero values (ties) are randomly set to "+1" or "-1." The resulting vector is similar to all the superposed terms. Finally, permutation (Π) shuffles the elements in a vector according to the given permutation. It is suitable for encoding sequences.

2.2. Encoding Nested Structures with VSAs

Here, we formalize the representation of nested data structures using VSAs. Let us first focus on the VSA-based symbol binding with F symbols. The encoding starts with defining F separate codebooks ($\mathbf{X}^1, \mathbf{X}^2, ..., \mathbf{X}^F$). The codebooks are defined as $\mathbf{X}^f := \{\mathbf{x}_i^f\}_{i=1}^{M_f}$, where each $\mathbf{x}_i^f \in \{-1, 1\}^D$, $f \in \{1, 2, ..., F\}$ is a randomly drawn bipolar vector. Each codebook contains as many codewords (M_f) as the number of values the corresponding symbol can have[1]. For example, we could have one color codebook to store three random vectors representing red, blue, and green colors. In a second codebook, we could store the possible shapes with three

[1]In most cases, we assume that all codebooks contain the same number of codewords, i.e., $M_i = M_j \, \forall i, j \in \{1, 2, ..., F\}$.

code vectors for square, triangle, and circle. A product vector $\mathbf{p} = \bigodot_{f=1}^{F} \mathbf{x}_{c_f}^f$ is then formed by binding the selected symbols from the codebooks.

As a next step, the combination of binding and superposition can describe more complex structures:

$$\mathbf{q} = \sum_{i=1}^{N} \mathbf{p}_i = \sum_{i=1}^{N} \bigodot_{f=1}^{F} \mathbf{x}_{c_{i,f}}^f. \tag{1}$$

Overall, we superpose N different products (i.e., $\mathbf{p}_i \neq \mathbf{p}_j, \forall i \neq j$). In most applications, it is assumed that the number of superposed terms is known. The goal of the decoding is to find all indices $\hat{c}_{i,f}, \forall i \in \{1, ..., N\}, f \in \{1, ..., F\}$.

2.3. Resonator networks

To decompose a single symbol bound representation ($\mathbf{p}$), we aim to find the estimated factors $\hat{\mathbf{x}}^f \in \mathbf{X}^f$, $\forall f \in \{1, 2, ..., F\}$ that satisfy

$$\mathbf{p} = \bigodot_{f=1}^{F} \hat{\mathbf{x}}^f. \tag{2}$$

A naive brute-force approach would compare the product vector ($\mathbf{p}$) to all possible combinations spanned by the product space

$$\mathbf{P} = \bigodot_{f=1}^{F} \mathbf{X}^f := \left\{ \mathbf{x}_1^1 \odot \mathbf{x}_1^2 \odot ... \odot \mathbf{x}_1^F, \mathbf{x}_2^1 \odot \mathbf{x}_1^2 \odot ... \odot \mathbf{x}_1^F, ..., \mathbf{x}_{M_1}^1 \odot \mathbf{x}_{M_2}^2 \odot ... \odot \mathbf{x}_{M_F}^F \right\}. \tag{3}$$

This results in a combinatorial search problem requiring $\left(\prod_{f=1}^{F} M_f\right)$ similarity computations.

Alternatively, resonator networks [7, 8] reduce the decoding complexity by iteratively searching in superposition through all possible solutions. Recently, a nondeterministic and nonlinear variant [9] introduced nonlinear activations and stochastic behavior, notably improving the factorization convergence and the solvable problem size.

The decoding begins with initializing the estimate factors $(\hat{\mathbf{x}}^1(0), ..., \hat{\mathbf{x}}^F(0))$ by bundling all vectors from the corresponding codebooks. Then, each estimate factor $(\hat{\mathbf{x}}^f(t))$ at iteration t is updated through the following steps.

The estimate factors from the previous iteration are unbound from the product vector using the Hadamard product:

$$\tilde{\mathbf{x}}^f(t) = \mathbf{p} \odot \left(\bigodot_{i=1, i\neq f}^{F} \hat{\mathbf{x}}^i(t) \right) \tag{4}$$

Next, we query the associative memory, which contains the codebook $\mathbf{X}^f$ for the factor f, with the unbound factor estimates. At iteration t, this yields an M_f-dimensional vector of

similarity scores ($\mathbf{a}^f(t)$) for each factor f. The i-th element in $\mathbf{a}^f(t)$ is computed using the cosine similarity ($\cos$):

$$\mathbf{a}^f(t)[i] = \cos(\tilde{\mathbf{x}}^f(t), \mathbf{x}_i^f). \tag{5}$$

The nondeterministic and nonlinear resonator network variant [9] processes the similarity vector by adding additive random noise and passing it through a threshold function:

$$\mathbf{a}'^f(t)[i] = \text{thresh}\left(\mathbf{a}^f(t)[i] + n; T\right) \tag{6}$$

where n is an i.i.d. Gaussian random variable with zero mean an standart deviation σ, and $\text{thresh}\,(\cdot; T)$ propagates similarity values that are larger than the threshold T, whereas lower ones get zeroed out.

Finally, we generate the next factor estimate $\hat{\mathbf{x}}^f(t+1)$ as the weighted bundling of the factor's codevectors together with an elementwise sign activation (sign):

$$\hat{\mathbf{x}}^f(t+1) = \text{sign}\left((\mathbf{X}^f)^T \mathbf{a}'^f(t)\right) \tag{7}$$

The iterative decoding is repeated until the resonator converges or a predefined maximum number of iterations (N) is reached. The convergence detection mechanism is based on an additional, fixed threshold. Decoding is stopped as soon as all similarity vectors contain an element that exceeds a predefined detection threshold value [9].

3. Related Works

The decoding of the nested structures in Eq. (1) can be posed as a pure superposition decoding task when unrolling the entire product space. A simple approach is to compute the similarity between the superposition ($\mathbf{q}$) and each vector in the product space ($\mathbf{P}$). Instead of the bipolar product matrix, an alternative readout matrix can be optimized to minimize the mean-squared error (MMSE) between the estimated and ground-truth indices, which enables the retrieval of more vectors from the superposition [12]. Another approach is to do iterative decoding using explaining away steps, where the current estimate is subtracted from the superposition, effectively reducing the interference. The iterative decoder is beneficial in robust short-package communication [13], and has been improved with a soft-feedback alignment and an MMSE readout for better retrieval, reaching to the state-of-the-art capacity of 1.2 bits/dimension [14].

The approaches above are able to retrieve many elements from a superposition; however, they cannot leverage the implicit factor structure that the product space provides. As a result, they need to search through all possible combinations in each decoding step, which requires maintaining explicitly all product vectors in the memory and computing the similarity between each product vector and the query. To this end, resonator networks have not only demonstrated to factorize single bound symbols, but also the superposition of multiple bound symbols. In an experiment [7], the colored MNIST digits were positioned at nine different positions (three vertical and three horizontal). Each colored digit was then described as the binding of four symbols: vertical position with three values, horizontal position with three values, color with

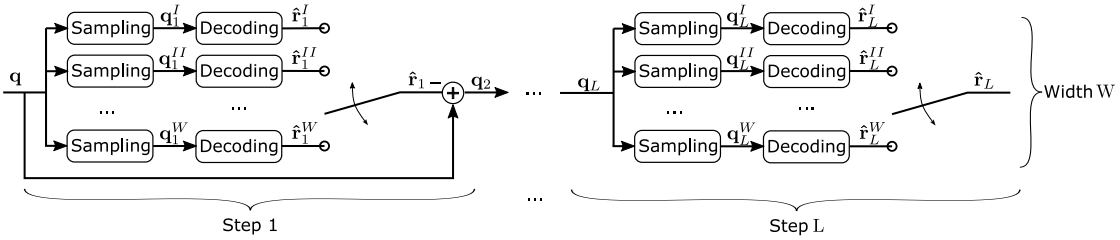

Figure 2: Configurable mixed decoding of superpositions of bound symbols. The variable width (W) describes the number of parallel decodings per step using query sampling and resonator-based decoding. The explaining away procedure is repeated for L steps.

seven values, and digits with ten values. This yielded a product space with 630 combinations. A vector dimension of $D = 500$ was sufficient to distinguish between all combinations. The resonator network was queried in multiple sequential steps to retrieve multiple bound symbols. In each step, the resonator network's estimation was subtracted from the input query. This explaining away approach reduced the interference from all the elements in the initial query. Despite limited problem size (630), the resonator network faced challenges to reliably retrieve more than one bound symbol from the superposition. The main problem is the noise amplification in the pure sequential decoding: an additional vector is added to the superposition if the prediction is wrong (see Fig. 1b). This additional vector can be interpreted as noise.

In this work, we enhance the sequential decoding by performing multiple decodings in parallel in each step. We generate multiple queries for the resonator network by performing a dedicated sampling. Our mixed decoding approach can retrieve up to 8× more bound symbols from a larger problem space (12,100) and a smaller dimensionality (D=256).

4. Mixed Approach to Superposition Decoding

This section presents the main contribution of the paper: we propose a mixed decoding approach to retrieve superpositions of bound symbols. Fig. 2 depicts the architecture of our mixed decoding. It relies on sequential decoding by explaining away steps, and extends it with parallel decodings in each step. The number of parallel decodings (denoted as the width W) and the number of sequential steps (depth L) can be configured depending on the problem at hand, as we will show in the experimental results (see Table 1).

4.1. Parallel Decoding

The parallel decoding makes W predictions per step by decoding W different sub-queries. The sub-queries can be generated by random sampling. For example, if the input query is a superposition of an even number of bound bipolar symbols, the zeros (ties) in the query vector can be randomly replaced with either a "+1" or a "-1". Suppose no ties are present in the superposition (e.g., due to an odd number of superpositions). In that case, we use an alternative

sampling strategy that samples each element based on its magnitude:

$$\mathbf{q}^w[i] = \begin{cases} +1, & \text{w. p. } (N + \mathbf{q}[i])/(2N) \\ -1, & \text{otherwise,} \end{cases} \tag{8}$$

where N is the number of superpositions and w is the index of the sub-query. At step l, the parallel decoding yields W possible solutions. Finally, we select the most promising solution based on the highest similarity between the predicted bound symbol ($\hat{\mathbf{r}}_l^w$) and the original query. The selected predicted bound symbol ($\hat{\mathbf{r}}_l$) is then passed to the next step for the explaining away.

4.2. Sequential Explaining Away

As the next step, the selected predicted bound symbol is subtracted from the input query, yielding

$$\mathbf{q}_{l+1} = \mathbf{q}_l - \mathrm{sim}(\mathbf{q}, \hat{\mathbf{r}}_l) \cdot \hat{\mathbf{r}}_l, \tag{9}$$

where $\mathrm{sim}(\cdot, \cdot)$ is the cosine similarity. We scale the subtraction with the similarity between the selected estimate and the original query, which serves as a confidence indicator [14]. The explaining away step attenuates predictions with low confidence, whereas highly confident predictions are fully subtracted.

4.3. Final Result Selection

After performing L decoding steps, we select the N distinct predictions with the highest confidence based on the cosine similarity between their bound symbol representation and the input query.

5. Experimental Results

We evaluate the proposed mixed decoding on the synthetic superposition of bound symbols and query examples generated by a neural network. We use the state-of-the-art stochastic sparsely-activated resonator [9] for optimal factorization performance. Following the practice by [8], the number of iterations is limited to 1/1000 of the product space and at least 2000.

Retrieval Capacity on Synthetic Queries. We start with the evaluation of our approach to synthetically generated data. We choose a problem with F=2 factors, an equal codebook size of M_1=M_2=110, and a dimension D=256. This yields a total problem size of $M_1 \cdot M_2$=12,100 which, e.g., could be encountered in the extreme-label text classification task of Amazon-12K [15]. Compared to [7], our proposed synthetic problem is notably larger (12,100 vs. 630) while the dimension is reduced (256 vs. 500). To evaluate the performance, we conduct 1000 experiments. We randomly select N symbol pairs in each experiment, bind them, and superpose all the bound symbol pairs. The average decoding accuracy is determined as the average ratio between correctly factorized symbol pairs and the total number of superposed pairs (N). Finally, we

Table 1
Retrieval capacity from synthetic examples. The stochastic sparsely-activated resonator is configured with F=2 factors, codebook size of M_1=M_2=110, dimension D=256, and maximum of 2000 iterations. N is the number of superposed bound symbols.

Method	Width	Depth	Total decodings	Retrieval capacity
Explaining away [7]	1	N	N	1
Sequential decoding	1	$2N$	$2N$	**5**
	1	$4N$	$4N$	5
Parallel decoding	$2N$	1	$2N$	1
Mixed decoding	2	N	$2N$	**5**
	4	N	$4N$	**8**

define the retrieval capacity as the highest number of superposed symbol pairs that can be successfully decoded with >99% accuracy.

Table 1 compares the retrieval capacity of the stochastic sparsely-activated resonator with different decoding approaches. Each approach can be described by its width and depth, which yields the resonator's total queries to be decoded. The standard explaining away [7] can only retrieve one bound symbol pair, i.e., it cannot decode any superposition. Increasing the depth to twice the number of superpositions ($2N$) improves the retrieval capacity to 5 superpositions. However, the benefit of the depth-increasing approach rapidly saturates: the retrieval capacity stays the same for a depth of $2N$ and $4N$. Even though the parallel decoding is not effective in isolation, it notably improves the retrieval capacity when applied with sequential decoding (hence we name it, mixed decoding). Increasing the width to 2 improves the retrieval capacity to 5. Moreover, further increasing the width to 4 yields the highest retrieval capacity (8). Compared to pure sequential decoding with a depth of $4N$, our mixed decoding can retrieve more bound symbol pairs (8 vs. 5) while requiring the same number of queries ($4N$).

RAVEN. The RAVEN dataset [16] contains Raven's progressive matrices tests with gray-scale images with a resolution of 160×160. A test provides 8 context and 8 answer panels, each containing objects with the following attributes: position, type, size, and color. The objects can be aligned in 7 different constellations containing a variable maximum number of objects (see N_{max} in Table 2). Following [6], we combine the positions from the different constellations, yielding 22 unique positions. Overall, the dataset contains C=6600 attribute combinations (22 positions $\times$ 5 types $\times$ 6 sizes $\times$10 colors). For every constellation, the dataset provides 6000 examples for training, 2000 for validation, and 2000 for testing.

This work focuses on recognizing all the objects in a given panel. We define the four codebooks for position, type, size, and color. We then train a ResNet-18 to map an input image to a D-dimensional query ($\mathbf{q}$), which resembles the superposition of all the bound symbols present in the image. This is achieved by maximizing the similarity between the query and each bound symbol using an additive cross-entropy loss [6]. Note that only the trainable weights of the ResNet-18 are updated during training while the codebooks are frozen. We train the architecture for 100 epochs using stochastic gradient descent (SGD) with a batch size of 256.

Table 2
Average object recognition accuracy on the RAVEN dataset. We set the decoding depth to N_{max}+1 and the width to 1. The number of objects in each panel is either known (oracle) or not (threshold). C=Center; L-R=Left-right; U-D=Up-down; OI-C=Out-in center; 2x2=2x2 grid, OI-G=out-in grid; 3x3=3x3 grid; Avg.=Average accuracy.

		C	**L-R**	**U-D**	**OI-C**	**2x2**	**OI-G**	**3x3**	**Avg.**
Max. number of objects (N_{max})		1	2	2	2	4	5	9	
Brute force [6]	threshold	100%	99.9%	100%	99.9%	99.5%	99.0%	83.6%	97.4%
	oracle	100%	100%	100%	100%	100%	99.6%	95.3%	99.3%
Resonator (ours)	threshold	100%	100%	100%	100%	100%	99.0%	83.6%	97.5%
	oracle	100%	100%	100%	100%	100%	99.9%	99.3%	99.9%

As a decoding baseline, the brute force approach computes the similarity between the query and each possible bound symbol in the product space. It selects the top-k bound symbols with the highest similarity if an oracle provides the number of objects. Otherwise, it detects the bound symbols whose similarity exceeds a threshold. Our resonator-based approach can distinguish the same between oracle and threshold.

Table 2 compares the classification accuracy of the brute force and our resonator-based decoding. Both approaches can recognize up to five objects with high accuracy ($\geq 99\%$) using only the threshold. The main challenge, however, appears in the 3x3 grid constellation with up to nine objects. If the exact number of objects is unknown, the brute force and resonator are on par (83.6%). However, the resonator outperforms the brute force in the oracle case by a margin of 4% (99.3% vs. 95.3%). At the same time, it reduces the computational cost by performing searching in superposition without explicitly searching through the entire product space.

6. Conclusions

We present a new decoding approach for efficient retrieval of hyperdimensional bound symbols from additived superpositions using the stochastic sparsely-activated resonator networks. Our novel mixed decoding includes both sequential and parallel decoding with configurable width and depth. When combined with a query sampling mechanism, our mixed approach retrieves up to 8$\times$ more additive components, compared to pure sequential decoding. In future work, we plan to investigate real-world applications with larger problems such as extreme-label classification containing up to 670,000 combinations [15].

References

[1] R. W. Gayler, Multiplicative binding, representation operators & analogy, in: Advances in Analogy Research: Integration of Theory and Data from the Cognitive, Computational, and Neural Sciences, 1998.

[2] R. W. Gayler, Vector symbolic architectures answer Jackendoff's challenges for cognitive

neuroscience, in: Proceedings of the Joint International Conference on Cognitive Science. ICCS/ASCS, 2003, pp. 133–138.
[3] T. A. Plate, Holographic reduced representations, IEEE Transactions on Neural Networks 6 (1995) 623–641.
[4] T. A. Plate, Holographic Reduced Representations: Distributed Representation for Cognitive Structures, Center for the Study of Language and Information, Stanford, 2003.
[5] P. Kanerva, Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors, Cognitive Computation 1 (2009) 139–159.
[6] M. Hersche, M. Zeqiri, L. Benini, A. Sebastian, A. Rahimi, A neuro-vector-symbolic architecture for solving Raven's progressive matrices, Nat. Mach. Intell. (2023).
[7] E. P. Frady, S. J. Kent, B. A. Olshausen, F. T. Sommer, Resonator networks, 1: An efficient solution for factoring high-dimensional, distributed representations of data structures, Neural Computation 32 (2020) 2311–2331.
[8] S. J. Kent, E. P. Frady, F. T. Sommer, B. A. Olshausen, Resonator networks, 2: Factorization performance and capacity compared to optimization-based methods, Neural Computation 32 (2020) 2332–2388.
[9] J. Langenegger, G. Karunaratne, M. Hersche, L. Benini, A. Sebastian, A. Rahimi, In-memory factorization of holographic perceptual representations, Nature Nanotechnology (2023).
[10] D. Kleyko, D. Rachkovskij, E. Osipov, A. Rahimi, A survey on hyperdimensional computing aka vector symbolic architectures, part i: Models and data transformations, ACM Computing Surveys 55 (2022).
[11] D. Kleyko, D. Rachkovskij, E. Osipov, A. Rahimi, A survey on hyperdimensional computing aka vector symbolic architectures, part ii: Applications, cognitive models, and challenges, ACM Computing Surveys 55 (2023).
[12] E. P. Frady, D. Kleyko, F. T. Sommer, A theory of sequence indexing and working memory in recurrent neural networks, Neural Computation 30 (2018) 1449–1513.
[13] H.-S. Kim, HDM: Hyper-dimensional modulation for robust low-power communications, in: 2018 IEEE International Conference on Communications (ICC), 2018.
[14] M. Hersche, S. Lippuner, M. Korb, L. Benini, A. Rahimi, Near-channel classifier: symbiotic communication and classification in high-dimensional space, Brain Informatics 8 (2021) 16.
[15] A. Ganesan, H. Gao, S. Gandhi, E. Raff, T. Oates, J. Holt, M. McLean, Learning with holographic reduced representations, in: Advances in Neural Information Processing Systems, volume 34, 2021, pp. 25606–25620.
[16] C. Zhang, F. Gao, B. Jia, Y. Zhu, S.-C. Zhu, Raven: A dataset for relational and analogical visual reasoning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.