A Survey on Hyperdimensional Computing aka Vector Symbolic Architectures, Part I: Models and Data Transformations

DENIS KLEYKO, University of California at Berkeley, USA and Research Institutes of Sweden, Sweden
DMITRI A. RACHKOVSKIJ, International Research and Training Center for Information Technologies and Systems, Ukraine and Luleå University of Technology, Sweden
EVGENY OSIPOV, Luleå University of Technology, Sweden
ABBAS RAHIMI, IBM Research – Zurich, Switzerland

This two-part comprehensive survey is devoted to a computing framework most commonly known under the names Hyperdimensional Computing and Vector Symbolic Architectures (HDC/VSA). Both names refer to a family of computational models that use high-dimensional distributed representations and rely on the algebraic properties of their key operations to incorporate the advantages of structured symbolic representations and distributed vector representations. Notable models in the HDC/VSA family are Tensor Product Representations, Holographic Reduced Representations, Multiply-Add-Permute, Binary Spatter Codes, and Sparse Binary Distributed Representations but there are other models too. HDC/VSA is a highly interdisciplinary field with connections to computer science, electrical engineering, artificial intelligence, mathematics, and cognitive science. This fact makes it challenging to create a thorough overview of the field. However, due to a surge of new researchers joining the field in recent years, the necessity for a comprehensive survey of the field has become extremely important. Therefore, amongst other aspects of the field, this Part I surveys important aspects such as: known computational models of HDC/VSA and transformations of various input data types to high-dimensional distributed representations. Part II of this survey [84] is devoted to applications, cognitive computing and architectures, as well as directions for future work. The survey is written to be useful for both newcomers and practitioners.

CCS Concepts: • **Theory of computation → Random projections and metric embeddings**; *Data structures design and analysis*; **Bloom filters and hashing**; • **Computing methodologies → Artificial intelligence**; *Knowledge representation and reasoning;*

It is worth noting that some of the literature referenced in this survey might be difficult to find, however, most of the literature can be traced using the publication list available online at: https://www.hd-computing.com/publications.

Denis Kleyko and Dmitri A. Rachkovskij contributed equally to this work.

The work of DK was supported by the European Union's Horizon 2020 Programme under the Marie Skłodowska-Curie Individual Fellowship Grant (839179). The work of DK was also supported in part by AFOSR FA9550-19-1-0241 and Intel's THWAI program. The work of DAR was supported in part by the National Academy of Sciences of Ukraine (grant no. 0120U000122, 0121U000016, and 0117U002286), the Ministry of Education and Science of Ukraine (grant no. 0121U000228 and 0122U000818), and the Swedish Foundation for Strategic Research (SSF, grant no. UKR22-0024).

Authors' addresses: D. Kleyko, University of California at Berkeley, Berkeley 94720, CA, Research Institutes of Sweden, Kista 16440, Sweden; email: denkle@berkeley.edu; D. A. Rachkovskij, International Research and Training Center for Information Technologies and Systems, Kiev 03680, Ukraine; Luleå University of Technology, Luleå 97187, Sweden; email: dar@infrm.kiev.ua; E. Osipov, Luleå University of Technology, Luleå 97187, Sweden; email: evgeny.osipov@ltu.se; A. Rahimi, IBM Research – Zurich, Zurich 8803, Switzerland; email: abr@zurich.ibm.com.

0360-0300/2022/12-ART130 $15.00
https://doi.org/10.1145/3538531

Additional Key Words and Phrases: Artificial intelligence, machine learning, distributed representations, data structures, hyperdimensional computing, vector symbolic architectures, holographic reduced representations, tensor product representations, matrix binding of additive terms, binary spatter codes, multiply-add-permute, sparse binary distributed representations, sparse block codes, modular composite representations, geometric analogue of holographic reduced representations

ACM Reference format:
Denis Kleyko, Dmitri A. Rachkovskij, Evgeny Osipov, and Abbas Rahimi. 2022. A Survey on Hyperdimensional Computing aka Vector Symbolic Architectures, Part I: Models and Data Transformations. *ACM Comput. Surv.* 55, 6, Article 130 (December 2022), 40 pages.
https://doi.org/10.1145/3538531

1 INTRODUCTION

The two main approaches to **Artificial Intelligence** (**AI**) are symbolic and connectionist. The symbolic approach represents information via symbols and their relations. Symbolic AI (the alternative term is Good Old-Fashioned AI, GOFAI) solves problems or infers new knowledge through the processing of these symbols. In the alternative connectionist approach, information is processed in a network of simple computational units often called neurons, so another name for the connectionist approach is artificial neural networks. This article presents a survey of a research field that originated at the intersection of GOFAI and connectionism, which is known under the names **Hyperdimensional Computing**, **HDC** (term introduced in [66]) and **Vector Symbolic Architectures**, **VSA** (term introduced in [34]).

In order to be consistent and to avoid possible confusion amongst the researchers outside the field, we will use the joint name HDC/VSA when referring to the field. It is also worth pointing out that probably the most influential and well-known (at least in the machine learning domain) HDC/VSA model is **Holographic Reduced Representations** (**HRR**) [139] and, therefore, this term is also used when referring to the field. For the reason of consistency, however, we use HDC/VSA as a general term while referring to HRR when discussing this particular model. HDC/VSA is the umbrella term for a family of computational models that rely on mathematical properties of high-dimensional vector spaces and use high-dimensional distributed representations called **hypervectors** (**HVs**[1]) for structured ("symbolic") representation of data while maintaining the advantages of connectionist distributed vector representations. This opens a promising way to build AI systems [104].

For a long time HDC/VSA did not gain much attention from the AI community. Recently, the situation has, however, begun to change and right now HDC/VSA is picking up momentum. We attribute this to a combination of several factors such as dissemination efforts from the members of the research community,[2,3] several successful engineering applications in the machine learning domain (e.g., [162, 167]) and cognitive architectures [21, 154]. The main driving force behind the current interest is the global trend of searching for computing paradigms alternative to the conventional (von Neumann) one, such as neuromorphic and nanoscalable computing, where HDC/VSA is expected to play an important role (see [78] and references therein for perspective).

The major problem for researchers new to the field is that the previous work on HDC/VSA is spread across many venues and disciplines and cannot be tracked easily. Thus, understanding the state-of-the-art of the field is not trivial. Therefore, in this article, we survey HDC/VSA with the aim of providing the broad coverage of the field, which is currently missing. While the survey

[1]Another term to refer to HVs, which is commonly used in the cognitive science literature, is "semantic pointers" [9].
[2]HDC/VSA Web portal. [Online.] Available: https://www.hd-computing.com.
[3]VSAONLINE. Webinar series. [Online.] Available: https://sites.google.com/ltu.se/vsaonline.

Table 1. A Qualitative Assessment of Existing HDC/VSA Literature with Elements of Survey

	Representation types	Connectionism challenges	The HDC/VSA models	Capacity of hypervectors	Hypervectors for symbols and sets	Hypervectors for numeric data	Hypervectors for sequences	Hypervectors for 2D images	Hypervectors for graphs
[137]	±	±	±	✗	✓	✗	±	✗	±
[66]	±	±	±	✗	✓	✗	±	✗	✗
[161]	✗	✗	±	±	✓	✗	±	✗	✗
[162]	✗	✗	✗	✗	✓	±	±	✗	✗
[122]	✗	✗	±	✗	✓	±	±	✗	✗
[36]	✗	✗	✗	✗	✓	✗	±	✗	✗
[169]	✗	✗	±	±	✓	✗	±	✗	✗
[78]	✗	✗	✗	✗	✓	✗	✓	✗	±
[43]	✗	✗	✗	✗	✓	✗	±	✗	✗
This survey, Part I Section #	2.1.1	2.1.2	2.3	2.4	3.1	3.2	3.3	3.4	3.5

is written to be accessible to a wider audience, it should not be considered as "the easiest entry point". For anyone who has not yet been exposed to HDC/VSA, before reading this survey, we highly recommend starting with three tutorial-like introductory articles [66, 68, 122]. The former two provide a solid introduction and motivation behind the field while the latter focuses on introducing HDC/VSA within the context of a particular application domain of robotics. Someone who is looking for a very concise high-level introduction to the field without too many technical details might consult [67]. Finally, there is a book [139] that provides a comprehensive treatment of fundamentals of two particular HDC/VSA models (see Sections 2.3.3 and 2.3.5). While [139] focused on the two specific models, many of the aspects presented there apply to HDC/VSA in general.

To our knowledge, there have been no previous attempts to make a comprehensive survey of HDC/VSA but there are articles that overview particular topics of HDC/VSA. Table 1 contrasts the coverage of this survey with those previous articles (listed chronologically). We use ± to indicate that an article partially addressed a particular topic, but either new results have been reported since then or not all related work was covered.

Part I of this survey has the following structure. In Section 2, we introduce the motivation behind HDC/VSA, their basic notions, and summarize currently known HDC/VSA models. Section 3 presents transformation of various data types to HVs. Discussion and conclusions follow in Sections 4 and 5, respectively.

Part II of this survey [84] will cover existing applications and the use of HDC/VSA in cognitive architectures.

Finally, due to the space limitations, there are topics that remain outside the scope of this survey. These topics include connections between HDC/VSA and other research fields as well as hardware implementations of HDC/VSA. We plan to cover these issues in a separate work.

2 HYPERDIMENSIONAL COMPUTING AKA VECTOR SYMBOLIC ARCHITECTURES

In this section, we describe the motivation that led to the development of early HDC/VSA models (Section 2.1), list their components (Section 2.2), overview existing HDC/VSA models (Section 2.3), and discuss the information capacity of HVs (Section 2.4).

2.1 Motivation and Basic Notions

The ideas relevant for HDC/VSA already appeared in the late 1980s and early 1990s [59, 99, 117, 132, 150, 174]. In this section, we first review the types of representations together with their advantages and disadvantages. Then we introduce some of the challenges posed to distributed representations, which turned out to be motivating factors for inspiring the development of HDC/VSA.

2.1.1 Types of Representation.

Symbolic representations. Symbolic representations [42, 124] are natural for humans and widely used in computers. In symbolic representations, each item or object is represented by a symbol. Here, by objects we refer to items of various nature and complexity, such as features, relations, physical objects, scenes, their classes, and so on. More complex symbolic representations can be composed from the simpler ones.

Symbolic representations naturally possess a combinatorial structure that allows producing indefinitely many propositions/symbolic expressions (see Section 2.1.2 below). This is achieved by composition using rules or programs as demonstrated by, e.g., Turing Machines. This process is, of course, limited by memory size that can, however, be expanded without changing the computational structure of a system. A vivid example of a symbolic representation/system is a natural language, where a plethora of words is composed from a small alphabet of letters. In turn, a finite number of words is used to compose an infinite number of sentences and so on.

Symbolic representations have all-or-none explicit similarity: the same symbols have maximal similarity, whereas different symbols have zero similarity and are, therefore, called dissimilar. To process symbolic structures, one needs to follow edges and/or match vertices of underlying graphs to, e.g., reveal the whole structure or calculate the similarity between composite symbolic structures. Therefore, symbolic models usually have problems with scaling because similarity search and reasoning in such models require complex and sequential operations, which quickly become intractable.

In the context of brain-like computations, the downside of the conventional implementation of symbolic computations is that they require reliable hardware [189], since any error in computation might result in a fatal fault. In general, it is also unclear how symbolic representations and computations with them could be implemented in a biological tissue, especially when taking into account its unreliable nature.

Connectionist representations: localist and distributed. In this article, connectionism is used as an umbrella term for approaches related to neural networks and brain-like computations. Two main types of connectionist representations are distinguished: localist and distributed [184, 186].

Localist representations are akin to symbolic representations in that for each object there exists a single corresponding element in the implementation of the representation. Examples of localist representations are a single neuron (node) or a single vector component.

There is some evidence that localist representations might be used in the brain (so-called "grandmother cells") [143]. In order to link localist representations, connections between elements can be created, corresponding to pointers in symbolic representations. However, constructing compositional structures that include combinations of already represented objects requires the allocation of (a potentially infinite number of) new additional elements and connections, which is neurobiologically questionable. For example, representing "abc", "abd", "acd", and so on requires introducing new elements for them, as well as connections to "a", "b", "c", "d" and so on. Also, localist representations share symbolic representations' drawbacks of lacking enough semantic basis, that may be considered as a lack of immediate explicit similarity between the representations. In other words, different neurons representing different objects are dissimilar and estimating object similarity requires additional processes.

Distributed representations[4] were inspired by the idea of a "holographic" representation as an alternative to the localist representation [48, 140, 184, 186]. They are attributed to a connectionist approach based on modeling the representation of information in the brain as "distributed" over many neurons. In distributed representations, the state of a set of neurons (of finite size) is modeled as a vector where each vector component represents a state of the particular neuron.

Distributed representations are defined as a form of vector representations, where each object is represented by a subset of vector components, and each vector component can belong to representations of many objects. This concerns (fully distributed) representations of objects of various complexity, from elementary features or atomic objects to complex scenes/objects that are represented by (hierarchical) compositional structures.

In distributed representations, the state of individual components of the representation cannot be interpreted without knowing the states of other components. In other words, in distributed representations the semantics of individual components of the representation are usually undefined, in distinction to localist representations.

In order to be useful in practice for engineering applications and cognitive modeling, distributed representations of similar objects should be similar (according to some similarity measure of the corresponding vector representations; see Section 2.2.2), thus addressing the semantic basis issue of symbolic and localist representations.

Distributed representations should have the following attractive properties:

— High representational capacity. For example, if one object is represented by M binary components of a D-dimensional vector, then the number of representable objects equals the number of combinations $\binom{D}{M}$, in contrast to D/M for localist representations;
— Explicit representation of similarity. Similar objects have similar representations that can be immediately compared by efficiently computable vector similarity measures (e.g., dot product, or Minkowski distance);
— A rich semantic basis due to the immediate use of representations based on features and the possibility of representing the similarity of the features themselves in their vector representations;
— The possibility of using well-developed methods for processing vectors;
— For many types of distributed representations—the ability to recover the original representations of objects;
— Ability to work in the presence of noise, malfunction, and uncertainty, in addition to neurobiological plausibility;
— Direct access to the representation of an object. Being a vector, a distributed representation of a compositional structure can be processed directly. This does not require tracing pointers as in symbolic representations or following connections between elements as in localist representations;
— Unified format. Every object, whether atomic or composite, is represented by a vector and, hence, the implementation operates at the level of vectors without being explicitly aware of the complexity of what is represented.

In our opinion, the last two properties listed above are closely connected to the challenges posed to distributed representations in the late 1980s and early 1990s that we briefly discuss in the next section.

[4]Note that here we discuss not only distributed representations in the form of HVs formed with HDC/VSA but also distributed representations in general, including the ones used in early connectionist approaches. We refer to the latter as conventional connectionist representations.

2.1.2 Challenges for Conventional Connectionist Representations. Let us consider several challenges faced by early distributed representations known as "superposition catastrophe", e.g., [153, 188]. And, at a higher level, by demand for "systematicity" [22], and much later, for fast compositionality [53]. These challenges led to the necessity to make distributed representations "structure-sensitive" by introducing the "binding" operation (Section 2.2.3).

"Superposition catastrophe". It was believed that connectionist representations cannot represent hierarchical compositional structures because of the superposition catastrophe, which manifestates itself in losing the information concerning object arrangements in structures, see [146, 153, 188] for discussion and references.

In the simplest case, let us activate binary localist representation elements corresponding to "a" and "b", then to "b" and "c". If we want to represent both "a" and "b", and "b" and "c" simultaneously, we activate all three representations: "a", "b", "c". Now, however, the information that "a" was with "b", and "b" was with "c" is lost. For example, the same "a", "b", "c" activation could be obtained by "a" and "c", and single "b". The same situation occurs if "a", "b", "c" are represented by distributed patterns.

Fodor and Pylyshyn criticisms of connectionism. In [22], criticism of connectionism was concerned with the parallel distributed processing approach covered in [48]. Fodor and Pylyshyn claimed that connectionism lacks Productivity, Systematicity, Compositionality, and Inferential Coherence that are inherent to systems operating with symbolic representations. Their definitions of these intuitively appealing issues are rather vague and interrelated, and their criticism is constrained to the early particular connectionist model that the authors chose for their critique. Therefore, we restate these challenges as formulated in [139]:

- Composition, decomposition, and manipulation: How are elements composed to form a structure, and how are elements extracted from a structure? Can the structures be manipulated using distributed representations?
- Productivity: A few simple rules for composing elements can give rise to a huge variety of possible structures. Should a system be able to represent structures unlike any it has previously encountered, if they are composed of the same elements and relations?
- Systematicity: Does the distributed representation allow processes to be sensitive to the structure of the objects? To what degree are the processes independent of the identity of elements in compositional structures?

Challenges to connectionism posed by Jackendoff. Four challenges to connectionism have been posed by Jackendoff [53], see also [34] for their HDC/VSA treatment. In principle, they are relevant to cognition but in particular they are related to language. The problem, in general, is how to neurally instantiate the rapid construction and transformation of the compositional structures.

- *Challenge 1. The binding problem*: the observation that linguistic representations must use compositional representations, taking into account order and occurring combinations. For example, the same words in different order and combination are going to produce different sentences.
- *Challenge 2. The problem of two*: how are multiple instances of the same object instantiated? For example, how are the "little star" and the "big star" instantiated so that they are both stars, yet distinguishable?
- *Challenge 3. The problem of variables*: concerns typed variables. One should be able to represent templates or relations with variables (e.g., names of relations) and values (e.g., arguments of relations).

— *Challenge 4. Binding in working and in long-term memories*: representations of the same binding should be identical in various types of memory. In other words, the challenge concerns the transparency of the boundary between a working memory and a long-term memory. It has been argued that linguistic tasks require the same structures to be instantiated in the working memory and the long-term memory and that the two instantiations should be functionally equivalent.

2.1.3 Binding to Address Challenges of Conventional Connectionist Representations. The challenges presented above made it clear that an adequate representation of compositional structures requires preserving information about their grouping and order. For example, in symbolic representations, brackets and symbolic order can be used to achieve this. For the same purpose, some mechanism of binding (à la "grouping brackets") was needed in distributed representations.

We view the binding operation as a means to form such a representation of an object that contains information about the context in which it was encountered. So, any implementation of the binding operation is expected to involve some modification of the representation of the original object. The context can essentially be anything, such as other homogeneous objects (e.g., data objects of the same type) or heterogeneous objects (e.g., data objects' positions, and roles). The binding operation, together with other operations, should provide a mechanism for constructing representations of compositional objects that reflect their similarity in a way that is useful for the problem being solved. For example, different combinations of the same objects should be represented differently. Also, it is often required to support a recovery (of the representation) of the original composite object.

One of the approaches to binding in distributed representations is based on the temporal synchronization of constituent activations [50, 111, 172, 188]. Although this mechanism may be useful on a single level of composition, its capabilities to represent compositional structures with multiple levels of hierarchy are questionable as it requires many time steps and complex "orchestration" to represent compositional structures. Another major problem is that such a temporal representation cannot be immediately stored in a long-term memory.

An alternative approach to binding, which eliminates these issues, is the so-called conjunctive coding approach used in HDC/VSA. Its predecessors were "extra units" considered by [46] to represent various combinations of active units of distributed patterns as well as outer products [117, 174], which, however, increased the dimensionality of representations (see details in Section 2.3.2). In HDC/VSA, the binding operation does not change the dimensionality of distributed representations. Moreover, it does not require any training.

To form "rich" compositional representations in HDC/VSA, both binding and superposition operations are used. Therefore, implementations of these operations need to maintain their properties when used together. In particular, the binding operation should not be associative with respect to the superposition operation in order to overcome the superposition catastrophe (Section 2.1.2).

Importantly, the schemes for forming distributed representations of compositional structures that exploit the binding and superposition operations produce distributed representations that are similar for similar objects (i.e., they take into account the similarity of object elements, their grouping, and order at various levels of hierarchy).

In summary, the main motivation for developing HDC/VSA was to combine the advantages of early distributed representations and those of symbolic representations, while avoiding their drawbacks, in pursuit for more efficient information processing and, ultimately, for better AI systems. One of the goals was to address the above challenges faced by conventional connectionist representations. The properties of HDC/VSA models introduced below allow addressing these challenges to a varying degree.

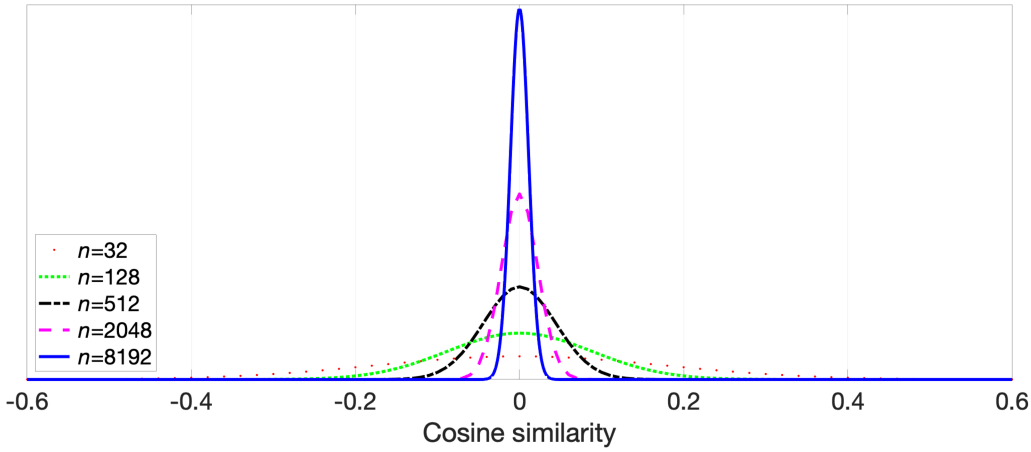

Fig. 1. Concentration of measure phenomenon between bipolar random HVs. Lines correspond to probability density functions of cosine similarities for different dimensionalities. Normal distributions were fitted to pairwise similarities for 2,000 random HVs.

2.2 Structure-sensitive Distributed Representations

2.2.1 Atomic Representations. When designing an HDC/VSA-based system it is common to define a set of the most basic objects/items/entities/concepts/symbols/scalars for the given problem and assign them HVs, which are referred to as atomic HVs. The process of assigning atomic HVs is often referred to as mapping, projection, embedding, formation, or transformation. To be consistent, we will use the term transformation.

For a given problem, we need to choose atomic representations of objects such that the similarity between the representations of objects corresponds to the properties we care about.[5] HVs of other (compositional) objects are formed by the atomic HVs (see Section 2.2.4). As follows from their name, atomic HVs are high-dimensional vectors. Values of HV components could be binary, real, or complex numbers and this list is not exhaustive, as we will see in Section 2.3.

In the early days of HDC/VSA, most of the works were focused on symbolic problems. In the case of working with symbols one could easily imagine many problems where a reasonable assumption would be that symbols are not related at all. So their atomic HVs are generated at random and are considered dissimilar, i.e., their expected similarity value is considered to be "zero". On the other hand, there are many problems where assigning atomic HVs fully randomly does not lead to any useful behavior of the designed system, see Section 3.2.

As mentioned above, in HDC/VSA random independent HVs generated from some distribution are used for representing objects that are considered independent and dissimilar. For example, randomly generated binary HVs with components from the set $\{0, 1\}$, with the probability of a 1-component being $p_1 = 1/2$ for dense binary representations [66] or for sparse binary representations [146] with $p_1 = 1/100$ or with a fixed number of $M << D$ randomly activated components. Such random HVs are analogous to "symbols" in symbolic representations. However, the introduction of new symbols in symbolic representations or nodes in localist representations requires changing the dimensionality of representations, whereas in HDC/VSA they are simply introduced as new HVs of fixed dimensionality. So HDC/VSA can accommodate N symbols with

[5]It is assumed that the problem to be solved has a static similarity structure. There may be problems that require a dynamic similarity structure, but these have not, so far, been the subjects of extensive study.

D-dimensional HVs such that $N >> D$. This happens because in high-dimensional spaces randomly chosen HVs are quasi-orthogonal to each other. The number of exactly orthogonal dimensions in a vector space is equal to its dimensionality D, but the number of quasi-orthogonal directions is exponential to the dimensionality. Exact orthogonality means that the angle between vectors is exactly 90 degrees, whereas quasi-orthogonality means that the angle between vectors is within some small interval around 90 degrees. As the dimensionality increases, the number of such quasi-orthogonal vectors increases exponentially even for a very tiny interval around 90 degrees. So, as the dimensionality increases, the number of quasi-orthogonal vectors increases rapidly while the mathematical properties of the quasi-orthogonal vectors become closer to those of exactly orthogonal vectors. This mathematical phenomenon is known as concentration of measure [103]. The peculiar property of this phenomenon is that quasi-orthogonality converges to exact orthogonality with increased dimensionality of HVs. This is sometimes referred to as the "blessing of dimensionality" [37]. Figure 1 provides a visual illustration of the case of bipolar HVs.

If the similarity between objects is important, then HVs are generated in such a way that their similarity characterizes the desired similarity between objects. The similarity between HVs is measured by standard vector similarity (or distance) measures.

2.2.2 *Similarity Measures.*

Similarity measure for dense representations. For dense HVs with real or integer components the most commonly used measures are the Euclidean distance:

$$\text{dist}_{\text{Euc}}(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_i (\mathbf{a}_i - \mathbf{b}_i)^2} = ||\mathbf{a} - \mathbf{b}||_2; \tag{1}$$

the dot (inner, scalar) product:

$$\text{sim}_{\text{dot}}(\mathbf{a}, \mathbf{b}) = \sum_i \mathbf{a}_i \mathbf{b}_i = \mathbf{a}^\top \mathbf{b}; \tag{2}$$

and the cosine similarity:

$$\text{sim}_{\text{cos}}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a}^\top \mathbf{b}}{||\mathbf{a}||_2 ||\mathbf{b}||_2}. \tag{3}$$

Note that these measures are closely related. In fact, when vectors $\mathbf{a}$ and $\mathbf{b}$ are normalized to have unit norm, there is exact correspondence: $\text{sim}_{\text{dot}}(\mathbf{a}, \mathbf{b}) = \text{sim}_{\text{cos}}(\mathbf{a}, \mathbf{b}) = 1 - \text{dist}_{\text{Euc}}(\mathbf{a}, \mathbf{b})^2/2$ (as follows from $||\mathbf{a} - \mathbf{b}||_2^2 = ||\mathbf{a}||_2^2 + ||\mathbf{b}||_2^2 - 2\mathbf{a}^\top \mathbf{b}$). For dense HVs with binary components, the normalized Hamming distance is the most common choice:

$$\text{dist}_{\text{Ham}}(\mathbf{a}, \mathbf{b}) = \frac{1}{D} \text{dist}_{\text{Euc}}(\mathbf{a}, \mathbf{b})^2 = \frac{1}{D} |\mathbf{a} \oplus \mathbf{b}|, \tag{4}$$

where $\oplus$ denotes a component-wise XOR operation. If the binary values of $\mathbf{a}$ and $\mathbf{b}$ are mapped to bipolar values by replacing 0-components to -1s, the Hamming distance is connected to the dot product as $\text{sim}_{\text{dot}}(\mathbf{2a - 1}, \mathbf{2b - 1}) = D(1 - 2\text{dist}_{\text{Ham}}(\mathbf{a}, \mathbf{b}))$.

Similarity measures for sparse representations. sim_{dot} and sim_{cos} are frequently used as similarity measures in the case of sparse HVs, i.e., when the number of non-zero components in HV is small.

Sometimes, Jaccard similarity is also used for this purpose:

$$\text{sim}_{\text{Jac}}(\mathbf{a}, \mathbf{b}) = \frac{|\mathbf{a} \wedge \mathbf{b}|}{|\mathbf{a} \vee \mathbf{b}|}, \tag{5}$$

where $\wedge$ and $\vee$ denote component-wise AND and OR operations, respectively.

2.2.3 Operations. Superposition and binding operations that do not change the HV dimensionality are of particular interest because they allow one to apply the same operations to their resultant HVs. But note that some applications might require changing the dimensionality of representations. Obviously, in biological systems one would be very surprised if dimensionalities were exactly equal. Assuming that the dimensionality is unchanged is a mathematical convenience for, e.g., using component-wise vector operations—rather than a requirement.

Superposition. The superposition is the most basic operation used to form an HV that represents several HVs. In analogy to simultaneous activation of neural patterns represented by HVs, this is usually modeled as a disjunction of binary HVs or addition of real-valued HVs. See more examples of particular implementations in Section 2.3. In HDC/VSA, this operation is known under several names: bundling, superposition, and addition. Due to its intuitive meaning, below, we will use the term superposition. The simplest unconstrained superposition (denoted as $\mathbf{s}$) is simply:

$$\mathbf{s} = \mathbf{a} + \mathbf{b} + \mathbf{c}. \tag{6}$$

In order to be used at later stages, the resultant HV is often required to preserve certain properties of atomic HVs. Therefore, some kind of normalization is often used to preserve, for instance, the norm, e.g., Euclidean:

$$\mathbf{s} = \frac{\mathbf{a} + \mathbf{b} + \mathbf{c}}{||\mathbf{a} + \mathbf{b} + \mathbf{c}||_2}; \tag{7}$$

or the type of components as in atomic HVs, e.g., integers in a limited range:

$$\mathbf{s} = f(\mathbf{a} + \mathbf{b} + \mathbf{c}), \tag{8}$$

where $f()$ is a function limiting the range of values of $\mathbf{s}$. For example, in the case of dense binary/bipolar HVs, the binarization of components via majority rule/sign function is used. In the case of sparse binary HVs, their superposition increases the density of the resultant HV. There are operations to "thin" the resultant HV and preserve the sparsity (see Section 2.3.8). Below, we will also use brackets $\langle\cdot\rangle$ when some sort of normalization is applied.

After superposition, the resultant HV is similar to its input HVs. Moreover, the superposition of HVs remains similar to each individual HV but the similarity decreases as more HVs are superimposed together. This could be seen as an unstructured and additive kind of similarity preservation. A fundamental property of the superposition is that it is associative, meaning that the sum is independent of the order of its components. However, [166] proposed a non-associative implementation of the superposition operation, which allows building a filter in the temporal (sequential) as well as in the elements' domains. Note that the properties above are desiderata for the superposition and, thus, any operation that has these properties shall be seen as an implementation of the superposition operation.

Binding. As mentioned above, the superposition operation alone leads to the superposition catastrophe (due to its associative property), i.e., during the recursive application of the superposition, the information about combinations of the initial objects (e.g., grouping) is lost since, e.g., $(\mathbf{a} + \mathbf{b}) + (\mathbf{c} + \mathbf{d}) = (\mathbf{a} + \mathbf{c}) + (\mathbf{b} + \mathbf{d}) = \cdots = \mathbf{a} + \mathbf{b} + \mathbf{c} + \mathbf{d}$.

This issue needs to be addressed in order to represent compositional structures (see Section 2.1.3). A binding operation (denoted as $\circ$ when no particular implementation is specified) is used in HDC/VSA as a solution. It is convenient to consider two types of binding: via multiplication-like operation and via permutation operation. These two types are selected because they are utilized by most applications. Let us first briefly discuss them before we consider concrete implementations of the binding operation by different HDC/VSA models in Section 2.3.

Multiplicative binding. Multiplication-like operations are used in different HDC/VSA models for implementing the binding operation when two or more HVs should be bound together. Concrete implementations of this type of binding are rather diverse and depend on a particular HDC/VSA model. Examples of multiplication-like binding operations include: conjunction for sparse binary HVs, component-wise multiplication for real-valued HVs, outer product for **tensor product representations** (**TPR**), and even matrix-vector multiplication when a role is represented by a matrix while a filler is represented by an HV. Please refer to Section 2.3 for concrete implementations for a particular HDC/VSA model. Figure 1 in [169] also presents the taxonomy of the most commonly used multiplicative bindings.

The HV obtained after binding together several input HVs depends on all of them. Similar input HVs produce similar bound HVs. However, the way the similarity is preserved after the binding is (generally) different from that of the superposition operation [138, 146]. We discuss this aspect in detail a few paragraphs further down.

In many situations, there is a need for an operation to reverse the result of the binding operation. This operation is called unbinding or release (denoted as $\oslash$). It allows the recovery of a bound HV [38, 191], e.g.,:

$$\mathbf{a} = \mathbf{b} \oslash (\mathbf{a} \circ \mathbf{b}). \tag{9}$$

Here, the binding of only two HVs is used for demonstration purposes. In general, many HVs can be bound to each other. As mentioned above, even matrix-vector multiplication can be used to implement binding. Note that when the unbinding operation is applied to a superposition of HVs, the result will not be exactly equal to the original bound HV. It will also contain crosstalk noise; therefore, the unbinding operation is commonly followed by a clean-up procedure (see Section 2.2.5) to obtain the original bound HV. In some models, the binding operation is self-inverse, which means that the unbinding operation is the same as the binding operation. We specify how the unbinding operations are realized in different models in Section 2.3.

Binding by permutation. Another type of HV binding is by permutation, which might correspond to, e.g., some position or some other role of an object. We denote the application of a permutation to an HV as $\rho(\mathbf{a})$[6]; if the same permutation is applied i times as $\rho^i(\mathbf{a})$ and the corresponding inverse is then $\rho^{-i}(\mathbf{a})$. Note that the permuted HV is dissimilar to the original one. However, the use of partial permutations [12, 92] allows preserving some similarity (see Section 3.4.1). For permutation binding, the unbinding is achieved by inverse permutation.

The permutation can be implemented via a "permutation vector" that associates each component's index with its index after permutation or by a matrix multiplication with the corresponding permutation matrix having a single 1-component in each row and column. Implementation of the permutation via the multiplication by the permutation matrix highlights some connection to the multiplicative binding implemented by matrix-vector multiplication as in [31]. Though the structure of permutation matrices is very different from the dense random matrices used in [31] (see also Section 2.3.4). However, a typical implementation of the multiplicative binding will take two HVs as an input, while the permutation operation takes as input an HV as well as permutation's identity. This difference in the type of input arguments explains why sometimes permutations are treated as the third type of basic HDC/VSA operations (in addition to superposition and binding).

[6]It is worth mentioning that $\rho(\cdot)$ denotes some arbitrary constant permutation. There may be multiple different permutations used in some applications, in which case they would need to be distinguished by, e.g., different symbols or sub-scripting. However, it is very common for only one permutation to be used, so the need to distinguish different permutations does not arise.

Often in practice a special case of permutation—cyclic shift—is used as it is very simple to implement. The use of cyclic (as well as non-cyclic) shift for positional binding in HDC/VSA was originally proposed in [91] for representing 2D structures (images) and 1D structures (sequences). Permutations were also used for representing the order in [146]. In [33], the permutation operation was introduced as essential for representing compositional structures when the binding operation is self-inverse. In a similar spirit, a permutation was used in [55, 135] to avoid the commutativity of a multiplicative binding by circular convolution. In general, a permutation can be used to do so for all commutative operations in all HDC/VSA models (see, e.g., Section 3.6.7 in [134]).

Later [66, 168] introduced a primitive for representing a sequence in a compositional HV using multiple applications of the same fixed permutation to represent a position in the sequence. This primitive was popularized via applications to texts [56, 168].

Similarity preservation by operations. For the superposition operation, the "contribution" of any input HV on the resultant HV is the same independently of other input HVs. In binding, the "contribution" of any input HV on the result depends on all other input HVs.

Also, the superposition and binding operations preserve similarity in a different manner. There is ***unstructured similarity***, which is the similarity of the resultant HV to the HVs used as input to the operation. The superposition operation preserves the similarity of the resultant HV to each of the superimposed HVs, that is, it preserves the unstructured similarity. For example, $\mathbf{d} = \mathbf{a} + \mathbf{b}$ is similar to $\mathbf{a}$ and $\mathbf{b}$. In fact, given some assumptions on the nature of the HVs being superimposed, it is possible to analytically analyze the information capacity of HVs (see Section 2.4 for details).

Most realizations of the binding operation do not preserve unstructured similarity. They do, however, preserve ***structured similarity***, that is, the similarity of the bound HVs to each other. Let us consider two i.i.d. random HVs: $\mathbf{a}$ and $\mathbf{b}$; $\mathbf{d} = \mathbf{a} \circ \mathbf{b}$ is similar to $\mathbf{d}' = \mathbf{a}' \circ \mathbf{b}'$ if $\mathbf{a}$ is similar to $\mathbf{a}'$ and $\mathbf{b}$ is similar to $\mathbf{b}'$. Thus, most realizations of the binding operation preserve structured similarity in a multiplicative fashion. When $\mathbf{a}$ and $\mathbf{b}$ are independent, as well as $\mathbf{a}'$ and $\mathbf{b}'$, then the similarity of $\mathbf{d}$ to $\mathbf{d}'$ is equal to the product of the similarities of $\mathbf{a}$ to $\mathbf{a}'$ and $\mathbf{b}$ to $\mathbf{b}'$. For instance, if $\mathbf{a} = \mathbf{a}'$, the similarity of $\mathbf{d}$ to $\mathbf{d}'$ will be equal to the similarity of $\mathbf{b}$ to $\mathbf{b}'$. If $\mathbf{a}$ is not similar to $\mathbf{a}'$, $\mathbf{d}$ will have no similarity with $\mathbf{d}'$ irrespective of the similarity between $\mathbf{b}$ and $\mathbf{b}'$ due to the multiplicative fashion of similarity preservation. The structured similarity is also preserved when using permutations since $\rho(\mathbf{a})$ is similar to $\rho(\mathbf{a}')$ if $\mathbf{a}$ is similar to $\mathbf{a}'$. This type of similarity is different from the unstructured similarity of the superimposed HVs: $\mathbf{d} = \mathbf{a} + \mathbf{b}$ will still be similar to $\mathbf{e} = \mathbf{a} + \mathbf{c}$ even if $\mathbf{b}$ is dissimilar to $\mathbf{c}$. Finally, it is worth noting that, in practice, it might be necessary to preserve both structured and unstructured similarities (e.g., for tasks involving a similarity search) when binding two objects. In that case, the HVs should be formed in such a way that both similarity types are taken into account. A simple example of such a representation is $\mathbf{a} + \mathbf{b} + \mathbf{a} \circ \mathbf{b}$ that still preserves some similarity to, e.g., $\mathbf{a} + \mathbf{c} + \mathbf{a} \circ \mathbf{c}$.

2.2.4 Representations of Compositional Structures. As introduced in Section 2.1.1, compositional structures are formed from objects where the objects can be either atomic or compositional. Atomic objects are the most basic (irreducible) elements of a compositional structure. More complex compositional objects are composed from atomic ones as well as from simpler compositional objects. Such a construction may be considered as a part-whole hierarchy, where lower-level parts are recursively composed to form higher-level wholes (see, e.g., an example of a graph in Section 3.5.2).

In HDC/VSA, compositional structures are transformed into their HVs using HVs of their elements and the superposition and binding operations introduced above. As mentioned in 2.1.3, it is a common requirement that similar compositional structures be represented by similar HVs. Another requirement that was studied intensively in the early days of HDC/VSA (see Section 3.1.1

of Part II of the survey [84] and, e.g., [123, 134, 174]) is the ability to transform representations of compositional structures without requiring the recovery of the structures (as would be the case for the symbolic approach). Recovering the original representation from its compositional HV, which we describe in the next section, might be an additional requirement. In Section 3 below, we will review a number of approaches to the formation of atomic and compositional HVs.

2.2.5 Recovery and Clean-up. Given a compositional HV, it is often desirable to find the particular input HVs from which it was constructed. We will refer to the procedure implementing this as recovery. This procedure is also known as decoding, reconstruction, restoration, decomposition, parsing, and retrieval.

For recovery, it is necessary to know the set of the input HVs from (some of) which the HV was formed. This set is usually called the dictionary, codebook, clean-up memory, or item memory.[7] In its simplest form, the item memory is just a matrix storing HVs explicitly, but it can be implemented as, e.g., a content-addressable associative memory [180]. Note that the stored HVs do not have to be atomic, they could also be compositional representations. For example, recent work [178] studied how to store compositional HVs in content-addressable associative memories à la Hopfield network. Moreover, recent works suggested [18, 79, 170] that it is not always necessary to store the item memory explicitly, as it can be easily rematerialized. Also, the recovery requires knowledge about the structure of the compositional HV, that is, information about the operations used to form a given HV (see an example below).

Most of the HDC/VSA models produce compositional HVs not similar to the input HVs due to the properties of their binding operations. So, to recover the HVs involved in the compositional HV, the use of the unbinding operation (Section 2.2.3 above) will typically be required. If the superposition operation was used when forming the compositional HV, after unbinding the obtained HVs will be noisy versions of the input HVs to be recovered (noiseless version can be obtained in the case when only binding operations were used to form a compositional HV). Therefore, a "clean-up" procedure is used as part of the recovery. In its most general form, the clean-up procedure can be viewed as projecting a query HV onto the subspace spanned by the item memory and returning, e.g., the weighted superposition (see examples in [24, 74]). In practice, however, the most common way of using the clean-up procedure is by finding the HV in the item memory that is most similar to the noisy query HV. The item memory performs the nearest neighbor search (i.e., returns an HV corresponding to the winner-take-all activation) that can be implemented by, e.g., comparing the query HV to each of the item memory's HVs or by some kind of search in an associative memory implementing the item memory. For HVs representing limited size compositional structures, there are guarantees for the exact recovery [183].

Let us consider a simple example. The compositional HV was obtained as follows: $\mathbf{s} = \mathbf{a} \circ \mathbf{b} + \mathbf{c} \circ \mathbf{d}$. For recovery from $\mathbf{s}$, we know that it was formed as the superposition of pair-wise bindings. In the simplest setup, we also know that the input HVs include $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, $\mathbf{d}$. The task is to find which HVs were in those pairs. Then, to find out which HV was bound with, e.g., $\mathbf{a}$ we unbind it with $\mathbf{s}$ as $\mathbf{a} \oslash \mathbf{s}$ resulting in $\mathbf{b} + \mathbf{a} \oslash (\mathbf{c} \circ \mathbf{d}) = \mathbf{b} + \text{noise} \approx \mathbf{b}$. Then, we use the clean-up procedure that returns $\mathbf{b}$ as the closest match (using the corresponding similarity measure). That way we know that $\mathbf{a}$ was bound with $\mathbf{b}$. In this setup, we immediately know that $\mathbf{c}$ was bound with $\mathbf{d}$. We can check this in the same manner, by first calculating, e.g., $\mathbf{d} \oslash \mathbf{s}$, and then cleaning it up. Note that noise is defined as whatever is quasi-orthogonal to the signals of interest, and this definition is operationalized by populating the item memory with the signals of interest. In other words, if the noise term in the

[7] In the context of HDC/VSA, the term clean-up was introduced in [132] while the term item memory was proposed in [64].

Table 2. Summary of HDC/VSA Models

HDC/VSA	Ref.	Space of atomic HVs	Binding	Unbinding	Superposition	Similarity
TPR	[174]	unit HVs	outer product	tensor-vector inner product	component-wise addition	sim_{dot}
HRR	[135]	unit HVs	circular convolution	circular correlation	component-wise addition	sim_{dot}
SMR	[73]	square matrices with unit norm	matrix multiplication	matrix multiplication with matrix transpose	component-wise addition	sim_{cos}
FHRR	[139]	complex unitary HVs	component-wise multiplication	component-wise multiplication with complex conjugate	component-wise addition	sim_{cos}
SBDR	[146]	sparse binary HVs	context-dependent thinning	repeated context-dependent thinning	component-wise disjunction	sim_{dot}
BSC	[63]	dense binary HVs	component-wise XOR	component-wise XOR	majority rule	dist_{Ham}
MAP	[33]	dense bipolar HVs	component-wise multiplication	component-wise multiplication	component-wise addition	sim_{cos}
MCR	[175]	dense integer HVs	component-wise modular addition	component-wise modular subtraction	component-wise discretized vector sum	modified Manhattan
CGR	[196]	dense integer HVs	component-wise modular addition	component-wise modular subtraction	component-wise discretized vector sum	sim_{cos}
MBAT	[31]	dense bipolar HVs	vector-matrix multiplication	multiplication with inverse matrix	component-wise addition	sim_{dot}
SBC	[102]	sparse binary HVs	block-wise circular convolution	block-wise circular convolution with approximate inverse	component-wise addition	sim_{dot}
GAHRR	[4]	unit HVs	geometric product	geometric product with inverse	component-wise addition	unitary product

Each model has its own atomic HVs and operations on them for binding and superposition, and a similarity measure.

above example ($\mathbf{a} \oslash (\mathbf{c} \circ \mathbf{d})$) was an HV in the item memory, then the system would not treat it as noise.

In a more complicated setup, it is not known that the input HVs were **a**, **b**, **c**, **d**, but the HVs in the item memory are known. So, we take those HVs, one by one, and repeat the operations above. A possible final step in the recovery procedure is to recalculate the compositional HV from the reconstructed HVs. The recalculated compositional HV should match the original compositional HV. Note that the recovery procedure becomes much more complex in the case where m HVs are bound instead of just pairs. It is easy to recover one of the HVs used in the binding, if the other $m - 1$ HVs are known. They can simply be unbound from the compositional HV.

When the other HVs used in the binding are not known, the simplest way to find the HVs used for binding is to compute the similarity with all possible bindings of HVs from the item memory. The complexity of this search grows exponentially with m. However, there is a recent work [24, 74] that proposed a mechanism called resonator network to address this problem.

2.3 The HDC/VSA Models

In this section, various HDC/VSA models are overviewed. For each model we provide a format of employed HVs and the implementation of the basic operations introduced in Section 2.2.3 above. Table 2 provides the summary of the models (see also Table 1 in [169]).[8]

2.3.1 A Guide on the Navigation Through the HDC/VSA Models. Before exposing a reader to the details of each HDC/VSA model, it is important to make a comment on the nature of their diversity and enable an intuition behind selecting the best model for the practical usage.

[8]Note that the binding via the permutation operation is not specified in the table. That is because it can be used in any of the models even if it was not originally proposed to be a part of it. Here, we limit ourselves to specifying only the details of different models but see, e.g., [169] for some comparisons of seven different models from Table 2 (the work did not cover the GAHRR, MCR, and TPR models) and some of their variations (2 for HRR, 3 for MAP, and 2 for SBDR). Note also that in the table we specify the similarity measure used in the source references. However, as indicated in Section 2.2.2, there is a number of alternatives. Thus, the mentioned similarity measure by no means should be considered as a prescription. Instead, in practice, any similarity measure that can be applied to the particular model can be used.

The current diversity of the HDC/VSA models is a result of the evolutionary development of the main vector symbolic paradigm by independent research groups and individual researchers. Initially, the diversity comes from different initial assumptions, variations in the neurobiological inspiration and the particular mathematical background of the originators. Therefore, from a historical perspective, the question of selecting a candidate for the best model is ill-posed. Recent work [169] started to perform a systematic experimental comparison between models. We emphasize the importance of further investigations in this direction in order to facilitate conscious choice of one or another model for a given problem and to raise HDC/VSA to the level of matured engineering discipline.

However, the usage of different models could already be prioritized at this moment using the target computing hardware perspective. The recent developments of unconventional computing hardware aim at improving the energy efficiency over the conventional von Neumann architecture in AI applications. Another driving factor is reliability. As conventional hardware for the von Neumann architecture gets smaller, it eventually becomes unreliable, so there is a need for computing frameworks that are robust to unreliable hardware and HDC/VSA is a promising candidate. The availability of such a robust computing framework also opens the door to other types of hardware implementation that are inherently unreliable and would not be suitable for a von Neumann architecture. Various unconventional hardware platforms (see Section 4.3) deliver great promises in moving the borders of energy efficiency and operational speed beyond the current standards.

Independently on the type of the computing hardware, any HDC/VSA model can be seen as an abstraction of the algorithmic layer and can thus be used for designing computational primitives, which can then be mapped to various hardware platforms using different models [78]. In the next subsections, we present the details of the currently known HDC/VSA models.

2.3.2 Tensor Product Representations. The TPR model is one of the earliest models within HDC/VSA. The TPR model was originally proposed by Smolensky in [174]. However, it is worth noting that similar ideas were also presented in [117, 118] around the same time but received much less attention from the research community. Atomic HVs are vectors selected uniformly at random from the Euclidean unit sphere $S^{(D-1)}$. The binding operation is an outer product of HVs. So, the dimensionality of bound HVs grows exponentially with their number (e.g., two bound HVs have the dimension D^2, three bound vectors have the dimension D^3, and so on). The vectors need not be of the same dimension. This points to another important note that the TPR model may or may not be categorized as an HDC/VSA model depending on whether the fixed dimensionality of representations is considered a compulsory attribute of an HDC/VSA model or not. Historically, HDC/VSA models have required the operator result HVs to be the same dimensionality as the argument HVs. This was for engineering feasibility and mathematical convenience. It is useful to think of HDC/VSA as TPR with the result projected into a lower dimensional space in a way that retains some useful properties of TPR. One way is to think of TPR as a predecessor of HDC/VSA.

The superposition is implemented by (tensor) addition. Since the dimensionality grows, a recursive application of the binding operation is challenging. The resultant tensor also depends on the order in which the HVs are presented. Binding similar HVs will result in similar tensors. The unbinding is realized by taking the tensor product representation of a compositional structure and extracting from it the HV of interest. For linearly independent HVs, the exact unbinding is done as the tensor multiplication by the unbinding HV(s). Unbinding HVs are obtained as the rows of inverse of the matrix with atomic HVs in columns. Approximate unbinding is done using an atomic HV instead of the unbinding HV. If the atomic HVs are orthonormal, this results in the exact unbinding.

Though the similarity measure was not specified, we assume it to be $\mathrm{sim}_{\mathrm{cos}}$ or $\mathrm{dist}_{\mathrm{Euc}}$.

2.3.3 Holographic Reduced Representations. The HRR model was developed by Plate in the early 1990s [132, 134, 135].

The HRR model was inspired by Smolensky's TPR model [174], Hinton's "reduced descriptions" [47], and many other works, e.g., [10, 19, 108, 120, 171] to mention a few. Note that due to the usage of HRR in Semantic Pointer Architecture Unified Network [20], sometimes the HRR model is also referred to as the **Semantic Pointer Architecture**. The most detailed source of information on HRR is Plate's book [139].

In HRR, the atomic HVs for representing dissimilar objects are real-valued and their components are independently generated from the normal distribution with mean 0 and variance $1/D$. For large D, the Euclidean norm is close to 1.[9] The binding operation is defined on two HVs (**a** and **b**) and implemented via the circular convolution, which projects the outer product back onto D-dimensional space:

$$z_j = \sum_{k=0}^{D-1} b_k a_{j-k \mod D}, \tag{10}$$

where z_j is the jth component of the resultant HV $\mathbf{z} = \mathbf{a} \circ \mathbf{b}$.

The circular convolution multiplies norms, and it approximately preserves the unit norms of input HVs. The bound HV is not similar to the input HVs. However, the bound HVs of similar input HVs are similar. The unbinding is done by the circular correlation of the bound HV with one of the input HVs. The result is noisy, so a clean-up procedure (see Section 2.2.5) is required. There is also a recent proposal in [38] for an alternative realization of the binding operation called **Vector-derived Transformation Binding**. It is claimed to better suit implementations in spiking neurons and was demonstrated to recover HVs of sequences (transformed with the approach from [165]) better than the circular convolution. Another proposal related to HRR is **Square Matrix Representations** (**SMR**) [73], which use random square matrices with unit norm and matrix multiplication as the binding operation (see Table 2 for summary). Finally, in [49] a general formulation of "quadratic binding" that subsumes, e.g., outer product-based binding and circular convolution-based binding was proposed. It can, for example, be used to adjust the dimensionality of the bound HV to be within D and D^2.

The superposition operation is component-wise addition. Often, the normalization is used after the addition to preserve the unit Euclidean norm of compositional HVs. In HRR, both superposition and binding operations are commutative. The similarity measure is sim_{dot} or sim_{cos}.

2.3.4 Matrix Binding of Additive Terms. In the **Matrix Binding of Additive Terms** (**MBAT**) model [31] by Gallant and Okaywe, it was proposed to implement the binding operation by matrix-vector multiplication. Such an option was also briefly mentioned in [139]. Generally, it can change the dimensionality of the resultant HV, if needed.

Atomic HVs are dense and their components are randomly selected independently from $\{-1, +1\}$. HVs with real-valued components are also possible. The matrices are random, e.g., with elements also from $\{-1, +1\}$. Matrix-vector multiplication results can be binarized by thresholding.

The properties of this binding are similar to those of most other models (HRR, Multiply-Add-Permute, and Binary Spatter Codes). In particular, similar input HVs will result in similar bound HVs. The bound HVs are not similar to the input HVs (which is especially evident here, since even the HV's dimensionality could change). The similarity measure is either dist_{Euc} or sim_{dot}. The superposition is a component-wise addition, which can be normalized. The unbinding could be

[9] It might be practical to use more constraints when forming atomic HVs, such as using HVs whose entries of fast Fourier transform have unit magnitude (see, e.g., [32, 88]).

done using the (pseudo) inverse of the role matrix (or the inverse for square matrices, guaranteed for orthogonal matrices, as in [185]).

In order to check if a particular HV is present in the sum of HVs that were bound by matrix multiplication, that HV should be multiplied by the matrix and the similarity of the resultant HV should be calculated and compared to a similarity threshold.

2.3.5 Fourier Holographic Reduced Representations. The **Fourier Holographic Reduced Representations** (**FHRR**) model [134, 139] was introduced by Plate as a model inspired by HRR where the HVs' components are complex numbers. The atomic HVs are complex-valued random vectors where each vector component can be considered as a phase angle (phasor) randomly selected independently from the uniform distribution over $(0, 2\pi]$. Usually, the unit magnitude of each component is used (unitary HVs). The similarity measure is the mean of sum of cosines of angle differences. The binding operation is a component-wise complex multiplication, which is often referred to as Hadamard product. The unbinding is implemented via the binding with an HV conjugate (component-wise angle subtraction modulo 2π). The clean-up procedure is used similarly to HRR.

The superposition is a component-wise complex addition, which can be followed by the magnitude normalization such that all components have the unit magnitude.

2.3.6 Binary Spatter Codes. The **Binary Spatter Codes** (**BSC**) model[10] was developed in a series of papers by Kanerva in the mid 1990s [60–63]. As noted in [137], the BSC model could be seen as a special case of the FHRR model where angles are restricted to 0 and π. In BSC, atomic HVs are dense binary HVs with components from $\{0, 1\}$. The superposition is a component-wise addition, thresholded (binarized) to obtain an approximately equal density of ones and zeros in the resultant HV. This operation is usually referred to as the majority rule or majority sum. It selects zero or one for each component depending on whether the number of zeros or ones in the summands is higher and ties are broken, e.g., at random. In order to have a deterministic implementation of the majority rule, it is common to assign a fixed random HV, which is included in the superposition when the number of summands is even. Another alternative is to use a deterministic variant of tie breaking as proposed in [41].

The binding is a component-wise exclusive OR (XOR), which is equivalent to taking the diagonal of the XOR outer product of the arguments. That is, it is another example of projecting the outer product onto D dimensions. The bound HV is not similar to the input HVs but bound HVs of two similar input HVs are similar. The unbinding operation is also XOR, meaning that BSC has a binding operation with a self-inverse binding property, which may be an important property for choosing an appropriate model for a specific application. The connection to the FHRR model also indicates that only phase angles with binary discrete phase values can support the self-inverse binding property. But will not support fractional exponentiation (see Section 3.2.1) because they do not have sufficient resolution of the phase angles. The clean-up procedure is used similar to the models above. The similarity measure is often $\mathrm{dist}_{\mathrm{Ham}}$.

2.3.7 Multiply-Add-Permute. The **Multiply-Add-Permute** (**MAP**) model was proposed by Gayler [33]. It has several variants (see [169]); the bipolar one is isomorphic to the BSC model. In the MAP model, atomic HVs are dense bipolar HVs with components from $\{-1, 1\}$. This is used more often than the originally proposed version with the real-valued components from $[-1, 1]$.

The binding is component-wise multiplication, which is equivalent to taking the diagonal of the outer product of the arguments. The bound HV is not similar to the input HVs. However, bound

[10]This acronym was not used in the original publications but we use it here to be consistent with the later literature.

HVs of two similar input HVs are similar. The unbinding is also component-wise multiplication and requires knowledge of the bound HV and one of the input HVs for the case of two bound input HVs.

The superposition is a component-wise addition. The result of the superposition can either be left as is, normalized using the Euclidean norm, or binarized to $\{-1, 1\}$ with the sign function where ties for 0-components should be broken randomly (but deterministically for a particular input HV set). The choice of normalization is use-case dependent. The similarity measure is either sim_{dot} or sim_{cos}.

2.3.8 Sparse Binary Distributed Representations. The **Sparse Binary Distributed Representations** (**SBDR**) model [154], also known as Binary Sparse Distributed Codes [146, 153], proposed by Kussul, Rachkovskij, et al. emerged as a part of Associative-Projective Neural Networks [99] (see Section 3.2.2 in Part II of this survey [84]). There are two variants of the binding operation in the SBDR model: Conjunction and **Context-Dependent Thinning** (**CDT**). The idea of both variants is that they produce a resultant HV where the 1-components are a subset of the 1-components of the input HVs. This subset depends on all of the input HVs, so that (with the proper control of the sparsity) a particular combination of the input HVs results in a unique resultant HV, thus binding the input HVs and preserving the information on the input HVs used to obtain the resultant HV.

Conjunction-Disjunction. The Conjunction-Disjunction is one of the earliest HDC/VSA models. It uses binary vectors for atomic HVs [150] (the model was also proposed independently in [65]). Component-wise conjunction of HVs is used as the binding operation. Component-wise disjunction of HVs is used as the superposition operation. Both basic operations are defined for two or more HVs. Both operations produce HVs similar to the HVs from which they were obtained (unstructured similarity). That is, the bound HV is similar to the input HVs, unlike most of the other known binding operations, except for CDT (see below).

The HVs resulting from both basic operations are similar if their input HVs are similar (structured similarity). The operations are commutative so the resultant HV does not depend on the order in which the input HVs are presented. However, as noted in Section 2.2.3, this could easily be changed with the use of some random permutations to represent the order of arguments.

The probability of 1-components in the resultant HV can be controlled by the probability of 1-components in the input HVs. Sparse HVs allow using efficient algorithms such as inverted indexing or auto-associative memories for similarity search. Since conjunction reduces the density of the resultant HV compared to the density of the input HVs, it could be considered as another form of "reduced descriptions" from [47]. At the same time, the recursive application of such binding leads to HVs with all components set to zero. However, the superposition by disjunction increases the number of 1-components. These operations can therefore be used to compensate each other (up to a certain degree; see the next section). A modified version of the Conjunction-Disjunction model preserves the density of 1-components in the resultant HV as shown in the following subsubsection.

Context-Dependent Thinning. The CDT procedure proposed by Rachkovskij and Kussul [153] can be considered either as a binding operation or as a combination of superposition and binding. It was already used in the 1990s [98–100, 150] under the name "normalization" since it approximately maintains the desired density of 1-components in the resultant HV. This density, however, also determines the "degree" or "depth" of the binding as discussed in [153].

The CDT procedure is implemented as follows. First, m input binary HVs are superimposed by component-wise disjunction as:

$$\mathbf{z} = \vee_{i=1}^{m} \mathbf{a}_i. \tag{11}$$

This leads to an increased number of 1-components in the resultant HV and the input HVs are not bound yet. Second, in order to bind the input HVs, the resultant HV $\mathbf{z}$ from the first stage is permuted and the permuted HV is conjuncted with the HV from the first stage. This binds the input HVs and reduces the density of 1-components. Third, T such conjunctive bindings can be obtained by using different random permutations or by recursive usage of a single permutation. They are superimposed by disjunction to produce the resultant HV. These steps are described by the following equation:

$$\langle \mathbf{z} \rangle = \vee_{k=1}^{T} (\mathbf{z} \wedge \rho_k(\mathbf{z})) = \mathbf{z} \wedge \left(\vee_{k=1}^{T} \rho_k(\mathbf{z}) \right), \tag{12}$$

where ρ_k denotes the kth random permutation. The density of the resultant HV depends on the number of permutation-disjunctions T. If many of them are used, we eventually get the HV from the first stage, i.e., with no binding of the input HVs. The described version of the CDT procedure is called "additive". Several alternative versions of the CDT procedure were proposed in [153]. The CDT procedure preserves the information about HVs that were bound together by their remaining 1-components. The CDT procedure is commutative. It is defined for several input HVs. As well as Conjunction-Disjunction, the CDT procedure preserves both unstructured and structured similarity.

Due to the fact that both variants of binding in SBDR preserve unstructured similarity, the resultant HV is similar to the input HVs. This leads to $\langle \mathbf{a}, \mathbf{b} \rangle$ being similar to $\langle \mathbf{a}, \mathbf{c} \rangle$, which is not the case for, e.g., $\mathbf{a} \circ \mathbf{b}$ and $\mathbf{a} \circ \mathbf{c}$ in most other implementations of multiplicative binding. Here, the preservation of the unstructured similarity allows recovering the input HVs from the resultant HV by using the similarity search in the item memory containing all possible input HVs. This does not require any unbinding operation. However, it should be noted that an analog of unbinding could be constructed for SBDR using repeated binding. This was not investigated in details (except for some preliminary results in [98, 144]).

Finally, it is important to take note of the character of the structured similarity preservation by the CDT procedure. It depends on the thinning depth T. For a large T, the result is close to the superposition of the input HVs, and so the character of the structured similarity is close to additive. The smaller T is, the closer the character of the structured similarity is to the multiplicative one.

2.3.9 Sparse Block Codes. The main motivation behind the **Sparse Block Codes** (**SBC**) model, proposed by Laiho et al. [28, 102], is to use sparse binary HVs as in SBDR but with a binding operation where the bound HV is not similar to the input HVs (in contrast to SBDR).

Similar to SBDR, atomic HVs in the SBC model are sparse and binary with components from $\{0, 1\}$. HVs, however, are imposed with an additional structure. An HV is partitioned into blocks of equal size (so that the HV's dimensionality is a multiple of the block size). In each block, there is only one non-zero component, i.e., the activity in each block is maximally sparse as in Potts associative memory, see, e.g., [39].

In SBC, the binding operation is defined for two HVs and implemented via circular convolution, which is applied block-wise. The unbinding is done by the block-wise circular correlation of the bound HV with one of the input HVs. Note that with maximally sparse blocks, the binding operation is equivalent to the modulo sum of the corresponding indices [102], which demonstrates that each block is effectively a phase angle that is discretized to the block size. When the block is not maximally sparse, it functions more like a superposition of phase angles.

The superposition operation is a component-wise addition. If the compositional HV is going to be used for binding, it might be binarized by leaving only one of the components with the largest magnitude within each block active. If there are several components with the same largest

magnitude then a deterministic choice (e.g., the component with the largest position number could be chosen) can be made. The similarity measure is sim_{dot} or sim_{cos}.

2.3.10 Modular Composite Representations. The **Modular Composite Representations** (**MCR**) model [175] proposed by Snaider and Franklin shares some similarities with FHRR, BSC, and SBS. Components of atomic HVs are integers drawn uniformly from some limited range (denoted as r), which is a parameter of the MCR model. Note that this model is also connected to the FHRR model since the components of HVs can be seen as phase angles that are discretized to r values.

The binding operation is defined as component-wise modular addition (the module value depends on the range limit), which generalizes XOR used for binding in BSC. Binding properties are similar to most of other models. The unbinding operation is the component-wise modular subtraction.

The similarity measure is a variation of the Manhattan distance [175]:

$$\sum_{i=1}^{D} \min(\text{mod}_r(a_i - b_i), \quad \text{mod}_r(b_i - a_i)). \tag{13}$$

The superposition operation resembles that of FHRR. Integers are interpreted as discretized phase angles on a unit circle. First, phase angles are superimposed for each component (i.e., vector addition is performed). Second, the result of the superposition is normalized by setting the magnitude to one and the phase to the nearest phase corresponding to an integer from the defined range.

Finally, there is also a recent proposal of the **Cyclic Group Representations** (**CGR**) model [196] that highly resembles the idea of the MCR model. As it also defines a model that operates with integer-valued HVs but uses slightly different definitions for the similarity measure and the superposition operation.

2.3.11 Geometric Analogue of Holographic Reduced Representations. As its name suggests, the **Geometric Analogue of Holographic Reduced Representations** (**GAHRR**) model [4] (proposed by Aerts, Czachor, and De Moor) was developed as a model alternative to HRR. An earlier proposal was a geometric analogue to BSC [3]. The main idea is to reformulate HRR in terms of geometric algebra. The binding operation is implemented via the geometric product (a generalization of the outer product). Because of not projecting the geometric product onto a lower dimensional space, GAHRR is conceptually closest to the TPR model. The superposition operation is component-wise addition.

So far, GAHRR is mainly an interesting theoretical effort, as its advantages over conceptually simpler models are not particularly clear, but it might become more relevant in the future. Readers interested in GAHRR are referred to the original publications. An introduction to the model is given in [2, 4, 127, 131], examples of representing data structures with GAHRR are in [129] and some experiments comparing GAHRR to other HDC/VSA models are presented in [128, 130].

2.4 Information Capacity of HVs

An interesting question, which is often brought up by newcomers to the field, is how much information one could store in the superposition of HVs.[11] This value is called the information capacity of HVs. Usually, it is assumed that atomic HVs in superposition are random and dissimilar to each

[11]This question is almost always posed with respect to the superposition. It is interesting, however, to consider whether the question makes sense when applied to multiplicative binding. It has recently been mentioned in the context of studying resonator networks (see Section 2.2.5) [74].

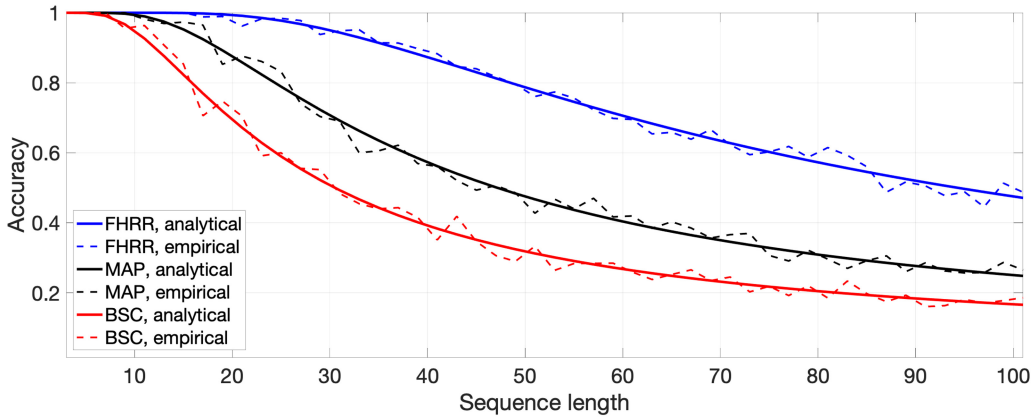

Fig. 2. The analytical and empirical accuracies of recovering sequences from HVs against the sequence length for three HDC/VSA models, $D = 256$, $N = 64$. The reported empirical accuracies were averaged over 5 simulations with randomly initialized item memories.

other. In other words, one could think about such superposition as an HV representing, e.g., a set of symbols. In general, the capacity depends on parameters such as the number of symbols in the item memory (denoted as N), the dimensionality of HVs (D), the type of the superposition operation, and the number of HVs being superimposed (m).

Early results on the capacity were given in [134, 139]. Some ideas for the case of binary/bipolar HVs in BSC, MAP, and MBAT were also presented in [31, 83]. The capacity of SBDR was analyzed in [86]. The most general and comprehensive analysis of the capacity of different HDC/VSA models (and also some classes of recurrent neural networks) was recently presented in [27]. The key idea of the capacity theory [27] can be illustrated by the following scenario when an HV to be recovered contains a valid HV from the item memory and crosstalk noise from some other elements of the compositional HV (e.g., from role-filler bindings, see Section 3.1.3). Statistically, we can think of the problem of recovering the correct atomic HV from the item memory as as a detection problem with two normal distributions: hit and reject; where hit corresponds to the distribution of similarity values (e.g., sim_{dot}) of the correct atomic HV while reject is the distribution of all other atomic HVs (assuming all HVs are random). Each distribution is characterized by its corresponding mean and standard deviation: μ_h & σ_h and μ_r & σ_r, respectively. Given the values of μ_h, σ_h, μ_r, and σ_r, we can compute the expected accuracy (p_{corr}) of retrieving the correct atomic HV according to:

$$p_{corr} = \int_{-\infty}^{\infty} \frac{dx}{\sqrt{2\pi}\sigma_h} e^{-\frac{(x-(\mu_h-\mu_r))^2}{2\sigma_h^2}} (\Phi(x, 0, \sigma_r))^{N-1}, \tag{14}$$

where $\Phi(x)$ is the cumulative Gaussian and N denotes the size of the item memory.

Figure 2 depicts the accuracy of retrieving sequence elements from its compositional HV (see Section 3.3) for three HDC/VSA models: BSC, MAP, and FHRR. The accuracies are obtained either empirically or with (14). As we can see, the capacity theory (14) predicts the expected accuracy very accurately.

The capacity theory [27] has recently been extended to also predict the accuracy of HDC/VSA models in classification tasks [87]. Reference [183] presented bounds for the perfect retrieval of sets and sequences from their HVs. Recent works [112, 169] have reported empirical studies of the capacity of HVs. Some of these results can be obtained analytically using the capacity theory. Additionally, [27, 45, 75, 182] elaborated on methods for recovering information from compositional

HVs beyond the standard nearest neighbor search in the item memory, reaching to the capacity of up to 1.2 bits/component [45]. Additional improvements to increase the capacity up to 1.4 bits/component were reported in [77]. This work also provided a taxonomy of existing methods for recovering information from compositional HVs and reported an empirical comparison of these methods.

The works above were focused on the case where a single HV was used to store information but as it was demonstrated in [14] the decoding from HVs can be improved if the redundant storage is used. As an example, the "multiset intersection" circuit in [35] effectively does this by summing over multiple copies of the same data (derived from multiple data permutations) to average away noise.

3 DATA TRANSFORMATION TO HVS

As mentioned above, the similarity of atomic symbolic and localist representations is all-or-none: identical symbols are considered maximally similar, whereas different symbols are considered dissimilar. On the other hand, the similarity of structures consisting of symbols (such as sequences, graphs, etc.) is often calculated using computationally expensive procedures such as an edit distance. HVs are vector representations, and, therefore, rely on simple vector similarity measures that can be calculated component-wise and provide the resultant similarity value immediately. This also concerns similarity of compositional structures. Thus, for example, the similarity of relational structures, taking into account the similarity of their elements, their grouping, and order, can be estimated by measuring the similarity of their HVs, without the need for explicit decomposition such as following edges or matching vertices of underlying graphs. Moreover, HDC/VSA can overcome problems with symbolic and localist representations concerning the lack of semantic basis (i.e., the lack of immediate similarity of objects; see Section 2.1.1). HDC/VSA overcomes these problems by explicit similarity in their representations, because HVs do not have to represent similarity in all-or-none fashion. These promises, however, bring the problem of designing concrete transformations that form HVs of various compositional structures such that the similarity between their HVs will reflect the similarity of the underlying objects.

In this section, we consider how data of various types are transformed to HVs to create such representations. The data types include symbols (Section 3.1.1), sets (Section 3.1.2), role-filler bindings (Section 3.1.3), numeric scalars and vectors (Section 3.2), sequences (Section 3.3), 2D images (Section 3.4), and graphs (Section 3.5).

3.1 Symbols and Sets

3.1.1 Symbols. As mentioned in Section 2.2.1, the simplest data type is a symbol. Usually, different symbols are considered dissimilar, therefore, using i.i.d. random HVs for different symbols is a standard way of transforming symbols into HVs. Such HVs correspond to symbolic representations (Section 2.1.1) since they behave like symbols in the sense that the similarity between the HV and its copy is maximal (i.e., the same symbol) while the similarity between two i.i.d. random HVs is minimal (i.e., different symbols).

3.1.2 Sets. In localist representations, sets are often represented as binary "characteristic vectors", where each vector component corresponds to a particular symbol from the universe of all possible symbols. Symbols present in a particular set are represented by one in the corresponding vector component. In the case of multisets, the values of the components of the characteristic vector are the counters of the corresponding symbols.

In HDC/VSA, a set of symbols is usually represented by an HV formed by the superposition of the symbols' HVs. The compositional HV preserves the similarity to the symbols' HVs. Notice that

Bloom filter [8]—a well-known data structure for approximate membership—is an HV where the superposition operation is obtained by the component-wise disjunction of binary HVs of symbols (as in SBDR) and, thus, can be seen as a special case of HDC/VSA [85].

Also note that, in principle, multiplicative bindings can be used to represent sets [80, 115], however, the similarity properties will be completely different from the representation by the superposition.

3.1.3 Role-filler Bindings. Role-filler bindings, which are also called slot-filler or key-value pairs, are a very general way of representing structured data records. For example, in localist representations the role (key) is the component's ID, whereas the filler (value) is the component's value. In HDC/VSA, this is represented by the result of binding. Both multiplicative binding and binding by permutation can be used. With multiplicative binding, both the role and the filler are transformed to HVs, which are bound together. When using permutation, it is common that the filler is represented by its HV, while the role is associated with either a unique permutation or a number of times that some fixed permutation should be applied to the filler's HV.

In the case of multiplicative binding, the associations do not have to be limited to two HVs and, therefore, data structures involving more associations can be represented. For example, the representation of "schemas" [121] might involve binding three HVs corresponding to {*context*, *action*, *and result*}.[12] It is worth mentioning that in symbolic role-filler bindings roles are usually atomic symbols. In HDC/VSA, role HVs are not required to be representations of atomic objects. They may represent compositional structures and may be constructed to have a desired similarity between roles.

A set of role-filler bindings is represented by the superposition of their HVs. Note that the number of (role-filler) HVs superimposed is limited to preserve the information contained in them, e.g., if one wants to recover the HVs being superimposed (see Sections 2.2.5 and 2.4).

3.2 Numeric Scalars and Vectors

In practice, numeric scalars are the data type present in many tasks, especially as components of vectors. Representing close numeric values with i.i.d. random HVs does not preserve the similarity of the original data. Therefore, when transforming numeric data to HVs, the usual requirement is that the HVs of nearby data points are similar and those of distant ones are dissimilar.

We distinguish three approaches to the transformation of numeric vectors to HVs [154]: compositional (Section 3.2.1), explicit receptive fields (Section 3.2.2), and **random projections** (**RP**) (Section 3.2.3).

3.2.1 Compositional Approach.

Representation of scalars in the compositional approach. In order to represent close values of a scalar by similar HVs, correlated HVs should be generated, so that the similarity decreases with the increasing difference of scalar values. Usually, a scalar is first normalized into some pre-specified range (e.g., [0, 1]). Next, it is quantized into finite grades (levels). Since HVs have a finite dimension and are random, they can reliably (with non-negligible differences in similarity) represent only a finite number of scalar grades. So only a limited number of grades are represented by correlated HVs, usually up to several dozens.

Early schemes for representing scalars by HVs were proposed independently in [150, 173]. These and other schemes [142, 158] can be considered as implementing some kind of "flow" from one component set of an HV to another, with or without substitution. For example, "encoding by

[12]Note that the original proposal in [121] was based of the superposition of three role-filler bindings.

concatenation" [158] uses two random HVs: one is assigned to the minimum grade in the range, while the second one is used to represent the maximum grade. The HVs for intermediate grades are formed using some form of interpolation, such as the concatenation of the parts of the HVs where the lengths of these two parts are proportional to the distances from the current grade to the minimum and maximum one. Note that in this scheme the representations of all values in the range have some non-zero degree of similarity to each other (apart from the two extreme values). It is possible to apply the same scheme to multiple concatenated ranges, so that the values separated by more than the range size have zero similarity.

Similar types of interpolations were proposed in [13, 86, 191]. The scheme in [173] and the "subtractive-additive" one [158] start from a random HV and recursively flip the values of some components to obtain the HVs of the following grades. This scheme was popularized, e.g., in [86, 160], for representing values of features when applying HDC/VSA in classification tasks. Another early scheme proposed for complex and real-valued HVs is called fractional power encoding (see Section 5.6 in [134]). It is based on the fact that complex-valued HVs (random angles on a unit circle) can be (component-wise) exponentiated to any value:

$$\mathbf{z}(x) = \mathbf{z}^{\beta x}, \tag{15}$$

where $\mathbf{z}(x)$ is an HV representing scalar x, $\mathbf{z}$ is a random HV called the base HV [25, 26], and β is the bandwidth parameter controlling the width of the resultant similarity kernel. This approach requires neither normalizing scalars in a pre-specified range nor quantizing them. The fractional power encoding can be immediately applied to FHRR and to any other representation where the phase angle resolution is sufficient. It is also used for HRR by making complex-valued HV using fast Fourier transform, exponentiating, and then making inverse fast Fourier transform to return to real-valued HVs (see [25, 26] for the details).

Representation of numeric vectors in the compositional approach. In the compositional approach to numeric vector representation by HVs ([150, 159, 177], we first form HVs for scalars (i.e., for components of a numeric vector). Then, HVs corresponding to values of different components are combined (often via superposition but the multiplicative binding is used too) to form the compositional HV of the numeric vector.

When constructing the compositional HV, it is important to make sure that scalar values of different components are represented by dissimilar HVs (even when two components have the same value). Otherwise, the association between a component and its value is lost. If scalars in different components are represented by dissimilar HVs, they can be superimposed or bound as is. Sometimes it is more practical (e.g., to reduce memory requirements) to keep a single set of correlated HVs representing all considered values of scalars.[13] The set is shared across all components. In this case, prior to applying the superposition operation, one has to associate the scalar's HV with its component identity in the numeric vector. This can be done by representing role-filler bindings (see Section 3.1.3) with some form of binding. This means that when using the role-filler binding scheme, the HV of a scalar corresponds to the filler while its component identity in the numeric vector corresponds to the role. As presented in Section 3.1.3, the association between role's and filler's HV can be realized either via the multiplicative binding with a role's HV or by permuting a filler's HV using the permutation assigned to the role.

Note that different schemes and models provide different types of similarity, for example, Manhattan distance (dist_{Man}) in [159, 183]. Note also that, unlike usual vectors, where the components

[13]This set of HVs is sometimes called "continuous item memory" since HVs are correlated.

are orthogonal, role HVs of nearby components could be made to correlate (e.g., nearby filtration banks are more correlated than banks located far away) [150].

3.2.2 Explicit Receptive Fields. The receptive field approach is often called coarse coding since a numeric vector is coarsely represented by the receptive fields activated by the vector. There are several well-known schemes utilizing explicit receptive fields, such as **Cerebellar Model Articulation Controller** [6], Prager Codes [141], and **Random Subspace Codes** (**RSC**) [94, 101], where receptive fields are various kinds of randomly placed and sized hyperrectangles, often in only some of the input space dimensions. It is worth noting that these schemes form (sparse) binary HVs. Similar approaches, however, can produce real-valued HVs, e.g., by using Radial Basis Functions as receptive fields [5, 119]. Details and comparisons can be found in [156, 157]. In particular, a similarity function between the input numeric vectors represented by RSCs was obtained.

3.2.3 Random Projections-based Approach. RP is another approach which allows forming an HV of a numeric vector $\mathbf{x}$ by multiplying it by an RP matrix $\mathbf{R}$:

$$\mathbf{z} = \mathbf{R}\mathbf{x}, \tag{16}$$

and possibly performing some normalization, binarization, and/or sparsification (such as preserving k largest vector components).[14] It is a well-known approach, which has been extensively studied in mathematics and computer science [52, 54, 126, 151, 187]. Originally, the RP approach was used in the regime where the resultant vectors $\mathbf{Rx}$ of smaller dimensionality were produced (i.e., used for dimensionality reduction). This is useful for, e.g., fast estimation of dist_{Euc} of original high-dimensional numeric vectors (as well as sim_{dot} and $\text{dist}_{\text{angle}}$). Well-known applications are in similarity search [52, 72] and compressed sensing [16]. First, a random $\mathbf{R}$ with components from the normal distribution was investigated [52, 54, 187], but then the same properties were proved for RP matrices with components from $\{-1, +1\}$ (bipolar matrices) and $\{-1, 0, +1\}$ (ternary and sparse ternary matrices) [1, 58, 106].

In the context of HDC/VSA, the RP approach using a sparse ternary matrix was first applied in [69] (see also Section 2.2.1 of Part II of the survey [84] for details). In [113, 114] (see Section 2.2.1 of Part II of the survey [84]), it was proposed to binarize or ternarize (by thresholding) the result of $\mathbf{Rx}$, which produced sparse HVs. The analysis of a sim_{cos} estimation by sparse HVs was first reported in 2006/2007 and then published in [147, 148, 155], whereby the RP with sparse ternary and binary matrices was used. In [15], the RP with sparse binary matrices was used for expanding the dimensionality of the original vector. Note that increasing the dimensionality of the original vector has been widely used previously in machine learning classification tasks as it allows linearizing non-linear class boundaries. In [7], thresholding of the HV produced by $\mathbf{Rx}$ was investigated and applied in the context of fast (i.e., sublinear versus the number of objects in the base) similarity search. It is common to use a single RP matrix to form an HV, but there are some approaches that rely on the use of several RP matrices [163]: $\mathbf{z} = \sum_{i=1}^{n} \lambda_i \mathbf{R}_i \mathbf{x}$, where λ_i determines the contribution of the ith RP $\mathbf{R}_i\mathbf{x}$ to the resultant HV and, thus, can be interpreted as a weighted superposition of n separate representations.

An appealing feature of the RP approach is that it can be applied to input vectors of an arbitrary dimensionality and number of component gradations, unlike the compositional or the receptive fields approaches. Some theoretical analysis of using RP for forming HVs was performed in [183].

[14] Note that (16) represents the values of $\mathbf{x}$ via the magnitudes of the corresponding columns in $\mathbf{R}$, so that $\mathbf{z}$ is the weighted superposition of the columns in $\mathbf{R}$. This approach to representing numeric vectors is different from the ideas in Section 3.2.1 above, where the difference in the direction of HVs is used to represent relative similarity between the corresponding scalars. However, the representation of numeric vectors via weighted superposition is rarely used in HDC/VSA since the result of (16) needs to be normalized, and, thus, the information about the absolute magnitudes of $\mathbf{z}$ is lost (see Section 2.2.3).

3.3 Sequences

Below we present several standard approaches for representing sequences with HVs. These approaches include binding of elements' HVs with their position HVs, bindings using permutations, binding with the context HVs, and using HVs for n-grams. For an overview of these approaches refer to, e.g., [176].

3.3.1 Binding of Elements' HVs with their Position or Context HVs. One of the standard ways of representing a sequence is to use a superposition of role-filler binding HVs where the filler is a symbol's HV and the role is its position's HV [41, 98, 136]. In [98], the order was represented by the weight of the subset of components selected from the initial HVs of sequence symbols (e.g., each preceding symbol was represented by more 1-components of the binary HV than the succeeding symbol). In [144], it was proposed to use thinning from the HVs of the previous sequence symbols (context). It is, however, most common to use i.i.d. random HVs to represent positions. An approach to avoid storing HVs for different positions is to use HDC/VSA models where the binding operation is not self-inverse. In this case, the absolute position k in a sequence is represented by binding the position HV to itself k times [133, 136]. Such a mechanism is called "trajectory association". The trajectory association can be convenient when the sequence representation is formed sequentially, allowing the sequence HV to be formed in an incremental manner. However, the sequence HV does not preserve the similarity to symbols' HVs in nearby positions, since the position HVs are not similar and so the role-filler binding HVs are also dissimilar. The MBAT model (Section 2.3.4) has similar properties, where the position binding is performed by multiplying by a random position matrix.

In order to make symbols in nearby positions similar to each other, one can use correlated position HVs [134, 176] such that, e.g., shifted sequences will still result in a high similarity between their HVs. This approach was also proposed independently in [13, 191] (see Section 3.2). Similar ideas for 2D images are discussed in Section 3.4.2.

Also, the HV of the next symbol can be bound to the HV of the previous one, or to several (weighted) HVs of previous ones, i.e., implementing the "binding with the context" approach [144].

3.3.2 Representing Positions via Permutations. An alternative way of representing position in a sequence is to use permutation [66, 91, 168]. Before combining the HVs of sequence symbols, the order i of each symbol is represented by applying some specific permutation to its HV i times (e.g., $\rho^2(\mathbf{c})$). However, such a permutation-based representation does not preserve the similarity of the same symbols in nearby positions, since the permutation does not preserve the similarity between the permuted HV and the original one. To preserve the similarity of HVs when using permutations, in [12, 92], it was proposed to use partial (correlated) permutations.

3.3.3 Compositional HVs of Sequences. Once symbols of a sequence are associated with their positions, the last step is to combine the sequence symbols' HVs into a single compositional HV representing the whole sequence. There are two common ways to combine these HVs.

The first way is to use the superposition operation, similar to the case of sets in Section 3.1.2. For example, for the sequence (a,b,c,d,e) the resultant HV (denoted as $\mathbf{s}$) is:

$$\mathbf{s} = \rho^0(\mathbf{a}) + \rho^1(\mathbf{b}) + \rho^2(\mathbf{c}) + \rho^3(\mathbf{d}) + \rho^4(\mathbf{e}). \tag{17}$$

Here, we exemplified the representation of positions via permutations but the composition will be the same for other approaches. The advantage of the approach with the superposition operation is that it is possible to estimate the similarity of two sequences by measuring the similarity of their HVs.

The second way of forming the compositional HV of a sequence involves binding of the permuted HVs, e.g., the sequence above is represented as (denoted as $\mathbf{p}$):

$$\mathbf{p} = \rho^0(\mathbf{a}) \circ \rho^1(\mathbf{b}) \circ \rho^2(\mathbf{c}) \circ \rho^3(\mathbf{d}) \circ \rho^4(\mathbf{e}). \tag{18}$$

The advantage of this sequence representation is that it allows forming quasi-orthogonal HVs even for sequences that differ in only one position. Similar to the trajectory association, extending a sequence can be done by permuting the current sequence's HV and adding or multiplying it with the next HV in the sequence, hence, incurring a fixed computational cost per symbol.

Note that compositional HVs of sequences can also be of a hybrid nature. For example, when representing positions via permutations the similarity of the same symbols in nearby positions is not preserved. One way to preserve similarity when some of the symbols in a sequence are permuted was presented in [81], where compositional HVs of sequences included two parts: a representation of a multiset of symbols constituting a sequence (a bag-of-symbols) and an ordered representation of symbols. The first part is transformed into an HV as a multiset of symbols (Section 3.1.2) by superimposing atomic HVs of all symbols present in a sequence. The second part is transformed into an HV as a sequence using permutations of symbols' HVs to encode their order (Section 3.3.2). Both representations are superimposed together to obtain the compositional HV.

In [149, 152], it was proposed to form such representations of sequences that preserve similarity of sequence elements in nearby positions as well as being equivariant with respect to sequence shifts. The property of equivariance means that the HV of the shifted sequence can be obtained not only by transforming this sequence into an HV, but just by some transformation of the HV of the original sequence. To achieve these properties of representations, [149] formed symbols' HVs using the superposition of several appropriately permuted versions of the corresponding atomic HV. The same idea was also used in [152] but with recursive applications of multiplicative binding instead of permutations.

3.3.4 n-grams. n-grams are n consecutive symbols of a sequence. In the n-gram representation of a sequence, all n-grams of the sequence are extracted, sometimes for different n. Usually, a vector containing n-gram statistics is formed such that its components correspond to different n-grams. The value of the component is the frequency (counter) of the occurrence of the corresponding n-gram.

There are various transformations of n-grams into HVs. The possibility of representing multisets with HVs (Section 3.1.2) allows forming HVs representing n-gram statistics as the superposition of HVs of individual n-grams. So the representations of n-grams in HVs are usually different in the following aspects:

— Distinguishing different n-grams, where "different" means even a single unique symbol (e.g., (abc) vs. (abd)) or the same symbols in different positions (e.g., (abc) vs. (acb));
— Forming similar HVs for similar n-grams.

To form dissimilar HVs for different n-grams, each n-gram can be assigned an i.i.d. random HV. However, an n-gram HV is usually generated from the HVs of its symbols. So in order to save space for the HV representations of n-grams, it is possible to represent them using compositional HVs [56, 145] instead of generating an atomic HV for each n-gram. Thus, the approaches above for representing sequences can be used. In [56, 76, 145], the permuted HVs of the n-gram symbols were bound by multiplicative binding. Alternatively, in [55, 57], the multiplicative binding of pairs of HVs was performed recursively, whereby the left HV corresponds to the already represented part of the n-gram, and the right HV corresponds to the next element of the n-gram. The left and right HVs are permuted by different random permutations.

In contrast to the approaches above, in [51, 164] the permuted HVs of the n-gram symbols were not bound multiplicatively, but were superimposed forming similar HVs for similar n-grams. In [41], the superposition of HVs corresponding to symbols in their positions was also used. However, the position of a symbol in a bi-gram (possibly from non-adjacent symbols) was specified by the multiplicative binding with the HVs of the left and right positions instead of permutations.

3.3.5 Stacks. A stack can be seen as a special case of a sequence where elements are either inserted or removed in a last-in-first-out manner. At any given moment, only the top-most element of the stack can be accessed and elements written to the stack before are inaccessible until all later elements are removed. HDC/VSA-based representations of stacks were proposed in [134, 179, 193]. So, the HV of a stack is a special kind of sequence representation and the permutation of stack's HV plays the role of moving the elements in the stack.

3.4 2D Images

There are a number of proposals for representing objects with a two-dimensional structure, such as 2D images, in HDC/VSA models. In this section, we cluster the existing proposals for representations into three groups[15]: permutation-based, role-filler binding-based, and neural network-based.

3.4.1 Permutation-based Representations. As discussed in Section 2.2.3, permutations are used commonly as a way of implementing binding with the position information. In the context of 2D images, permutations can be used by assigning one permutation for the x-axis and another one for the y-axis.

The simplest way of implementing a permutation is to use a cyclic shift as in [91, 96]. In order to implement two orthogonal shifts corresponding to the x- and y-axes, the HVs of features were restructured as a 3D tensor with two dimensions corresponding to the retina size along the x- and y-axes. Then the HV of a feature at some (x, y) position was obtained by shifting the corresponding 3D tensor x times along the x-axis and y times along the y-axis.

In a similar way, when using permutations as shift generalizations, to bind an HV of a pixel's value (or of an image feature) with its position, x-axis permutation is applied x times and y-axis permutation is applied y times [80, 115]. The image is represented as a compositional HV containing the superposition of all permuted HVs of pixels' values. HVs representing pixels' (features') values are formed using some approach for representing scalars. As permuted HVs are not similar in any of the works above, the same pixel's value even in nearby positions will be represented by dissimilar HVs.

The issue of the absence of similarity in permutation-based representations was addressed in [92, 97] by using partial permutations. The number of permuted components of some pixel's value HV (or, more generally, of some feature's HV) increases with the coordinate change (within some radius), up to the full permutation. So, inside the radius the similarity decreases. Outside the radius, the same process is repeated again. As a result, a pixel's value HV at (x, y) is similar to the HVs in nearby positions within the similarity radius. The features at positions were **Random Local Descriptors** (**RLD**) detected at some points of interest. For instance, for binary images, the RLD feature is detected if some (varied for different features) pixels inside a local receptive field take zero and one values. Note that RLD may be considered as a version of receptive fields approach

[15]There are proposals that cannot be placed unambiguously in one of the clusters. For example, the proposal from [73] directly used pixels values that were first flattened and normalized and then shuffled by some random permutation. The resultant vector was bound to itself with circular convolution to obtain a "trace" HV. Thus, this proposal is neither purely "permutation-based" nor "role-filler binding-based" as it relies on both. Nevertheless, the identified groups are instructive as they are used as design primitives in many transformations of 2D images into HVs.

(Section 3.2.2) and can form HVs representing images, as in the case of the **LInear Receptive Area** (**LIRA**) features [93].

3.4.2 Role-filler Binding-based Representations. In a 2D image, it is natural to represent a pixel's position and its value as a role-filler binding, where the pixel's position HV is the role and an HV corresponding to the pixel's value is the filler. Similar to the case of using permutations, the image HV is formed as a compositional HV containing the superposition of all role-filler binding HVs.

When it comes to pixels position HVs, the simplest approach is to treat each position as if it would be a unique symbol so that it can be assigned with a unique random HV. This approach allows forming image HVs through the role-filler bindings. It was investigated in [83, 110] for real-valued and dense binary HVs and in [82] for sparse binary HVs. The approach with unique HVs for roles, however, neither imposes any relations between pixels' position HVs nor does it allow preserving the similarity of pixels' position HVs in a local neighborhood, which, e.g., is useful for robustness against small changes in the 2D image. To address the latter issue, in [95] a method was described where the HVs of nearby x and y coordinates were represented as correlated HVs, using approaches for representing scalars (Section 3.2). Then the pixel value in a position (x, y) was represented by binding three HVs: one representing a pixel's value at (x, y) and two representing the x and y coordinates themselves (component-wise conjunction was used for the binding in [95]). Thus, similar values in close proximity were represented by similar HVs.

MOving MBAT (**MOMBAT**) [30] was proposed to explicitly preserve the similarity of HVs for nearby pixels. As in the MBAT model (Section 2.3.4), matrices were used as roles. In particular, the matrix for x and y coordinates was formed as a weighted superposition of two random matrices. Thus, for both x and y, nearby coordinates were represented by similar (correlated) matrices. Then, each (x, y) pair was represented by a matrix obtained by binding (multiplying) those corresponding matrices. A value of a pixel at (x, y) was represented by binding its HV with the matrix of (x, y). Finally, the obtained HVs were superimposed. All the HVs participating in MOMBAT had real-valued components. A similar approach with local similarity preservation for BSC and dense binary HVs was considered in [11].

The Fractional Power Encoding approach (Section 3.2.1) to the representation of 2D images was employed in [23, 90, 190]. It imposes a structure between pixels' positions and can preserve similarity in a local neighborhood. In particular, the image HV is formed as follows. Two random base HVs are assigned for the x-axis and y-axis, respectively. In order to represent x and y coordinates, the corresponding base HVs are exponentiated to coordinate values; and $\mathbf{z}(x, y)$ —the HV for the (x, y) coordinate—is formed as their binding (assuming $\beta = 1$ cf. (15)):

$$\mathbf{z}(x, y) = \mathbf{x}^x \circ \mathbf{y}^y. \tag{19}$$

Finally, the binding with the HV of the pixel value is done. The variant of the fractional power encoding-based representation in hexagonal coordinates was presented in [89].

3.4.3 Neural Network-based Representations. Since neural networks are currently one of the main tools for processing 2D images, it is becoming common to use them for producing HVs of 2D images. This is especially relevant since directly transforming pixel values into HVs does not usually provide a good representation for solving machine learning tasks with competitive performance. For example, there were some studies [43, 110] directly transforming 2D images in MNIST to HVs. Their accuracy was further improved by either enhancing the HV encoder (e.g., with a receptive field and max pooling layer [107], or learning binary class HVs from a cross-entropy loss [17]. However, the reported accuracy was, in fact, lower than the one obtained with, e.g., a kNN classifier applied directly to pixels' values, or a simple neural network with two layers. Recall, however, that the HV representations must be designed to capture the information that is

important for solving the problem and presenting it in a form that can be exploited by the HDC/VSA mechanisms. Therefore, it is important to either apply some feature extraction techniques (as was, e.g., the case in [93, 97]) or use neural networks as a front-end. One of the earliest attempts to demonstrate this was presented in [194, 195]. The proposed approach takes activations of one of the hidden layers of a convolutional neural network caused by an input image. These activations are then binarized and the result of the binarization is used as an initial grid state of an elementary cellular automaton. The automaton is evolved for several steps and the results of previous steps are concatenated together, the result of which is treated as an HV corresponding to the image. The automaton evolution performs non-linear feature extraction, in the spirit of LIRA and RLD.

Later works used activations of convolutional neural networks without extra cellular automata computations. For example, in [116, 122] pre-trained off-the-shelf convolutional neural networks were used, while in [44, 71] a network's attention mechanism and loss function were specifically designed to produce HVs with desirable properties. For example, it directed the output of the neural network to assign quasi-orthogonal HVs to images from different classes [71]. This quasi-orthogonality of the class HVs also enables continual learning of many novel classes, from very few training samples, with small interference, which leads to the state-of-the art classification accuracy on natural and handwritten images [44]. It is important to note that obtaining HVs from neural networks is a rather new direction and there are no studies that would scrupulously compare HVs obtained from different neural network architectures.

3.5 Graphs

3.5.1 Undirected and Directed Graphs. A graph, denoted as G, consists of nodes and edges. Edges can either be undirected or directed. Figure 3 presents examples of both directed and undirected graphs. First, we consider the following simple transformation of graphs into HVs [35]. A random HV is assigned to each node of the graph, following Figure 3 node HVs are denoted by letters (i.e., $\mathbf{a}$ for node "a" and so on). An edge is represented as the binding of HVs of the connected nodes, e.g., the edge between nodes "a" and "b" is $\mathbf{a} \circ \mathbf{b}$. The whole graph is represented simply as the superposition of HVs (denoted as $\mathbf{g}$) representing all edges in the graph, e.g., the undirected graph in Figure 3 is:

$$\mathbf{g} = \mathbf{a} \circ \mathbf{b} + \mathbf{a} \circ \mathbf{e} + \mathbf{b} \circ \mathbf{c} + \mathbf{c} \circ \mathbf{d} + \mathbf{d} \circ \mathbf{e}. \tag{20}$$

To represent directed graphs, the directions of the edges should be included into their HVs. This could be done, e.g., by applying a permutation, so that the directed edge from node "a" to "b" in Figure 3 is represented as $\mathbf{a} \circ \rho(\mathbf{b})$. Thus, the directed graph in Figure 3 is represented by the following HV:

$$\mathbf{g} = \mathbf{a} \circ \rho(\mathbf{b}) + \mathbf{a} \circ \rho(\mathbf{e}) + \mathbf{c} \circ \rho(\mathbf{b}) + \mathbf{d} \circ \rho(\mathbf{c}) + \mathbf{e} \circ \rho(\mathbf{d}). \tag{21}$$

Note that such representation implicitly labels graph's nodes. The elements of the graph $\mathbf{g}$ can be recovered using an item memory (Section 2.2.5). For graphs that have the same node HVs, the dot product is a measure of the number of overlapping edges. The described graph representations do not represent isolated vertices, but this could be fixed, by, e.g., separate representations of the vertex set and edge set [35].

3.5.2 Directed Ordered Acyclic Graphs. When modeling, e.g., analogical reasoning (Section 3.1.2 of Part II of the survey [84]) or knowledge graphs, it is common to use acyclic graphs where both edges and nodes have associated labels, and edges are directed. Let us consider the representation of a graph shown in Figure 4, which can also be written in the bracketed notation as: `cause(bite(Spot,Jane),flee(Jane,Spot))`.

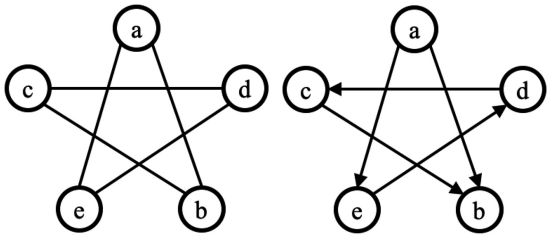

Fig. 3. An example of an undirected and a directed graph with 5 nodes. In the case of the undirected graph, each node has two edges.

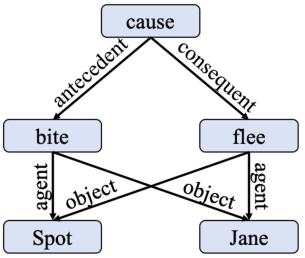

Fig. 4. An example of a labeled graph representing the episode – `cause(bite(Spot, Jane), flee(Jane,Spot))`.

Using the role-filler binding approach (see Section 3.1.3), a relation, e.g., `bite(Spot,Jane)`, can be represented by HVs (in HRR or BSC) as follows [139]:

$$\mathbf{BITE} = \langle \mathbf{bite} + \langle \mathbf{spot} + \mathbf{jane} \rangle + \mathbf{bite}_{\text{agent}} \circ \mathbf{spot} + \mathbf{bite}_{\text{object}} \circ \mathbf{jane} \rangle. \tag{22}$$

Here, **bite**, $\mathbf{bite}_{\text{agent}}$, and $\mathbf{bite}_{\text{object}}$ are atomic HVs, while $\mathbf{spot} = \langle \mathbf{spotID} + \mathbf{dog} \rangle$ and $\mathbf{jane} = \langle \mathbf{janeID} + \mathbf{human} \rangle$ are the compositional ones. In the same way, **FLEE** is formed, and finally the HV of the whole graph:

$$\langle \mathbf{cause} + \langle \mathbf{BITE} + \mathbf{FLEE} \rangle + \mathbf{cause}_{\text{antecedent}} \circ \mathbf{BITE} + \mathbf{cause}_{\text{consequent}} \circ \mathbf{FLEE} \rangle. \tag{23}$$

This particular representation was chosen to influence the similarity of the graph HVs because of the multiplicative binding operation properties in HRR and BSC. In SBDR, the binding preserves structured similarity in a different manner so the formation of graph's resultant HV could be made more compact [146]:

$$\mathbf{BITE} = \langle \langle \mathbf{bite}_{\text{agent}} \vee \mathbf{spot} \rangle \vee \langle \mathbf{bite}_{\text{object}} \vee \mathbf{jane} \rangle \rangle; \tag{24}$$

$$\mathbf{FLEE} = \langle \langle \mathbf{flee}_{\text{agent}} \vee \mathbf{jane} \rangle \vee \langle \mathbf{flee}_{\text{object}} \vee \mathbf{spot} \rangle \rangle; \tag{25}$$

$$\mathbf{CAUSE} = \langle \langle \mathbf{cause}_{\text{antecedent}} \vee \mathbf{BITE} \rangle \vee \langle \mathbf{cause}_{\text{consequent}} \vee \mathbf{FLEE} \rangle \rangle. \tag{26}$$

Also, Predicate-Arguments relation representation using random permutations to represent the order of relation arguments was proposed in [68, 146, 153]:

$$\mathbf{BITE} = \langle \mathbf{bite} + \mathbf{spot} + \mathbf{jane} + \rho_{\text{agent}}(\mathbf{spot}) + \rho_{\text{object}}(\mathbf{jane}) \rangle; \tag{27}$$

$$\mathbf{FLEE} = \langle \mathbf{flee} + \mathbf{spot} + \mathbf{jane} + \rho_{\text{agent}}(\mathbf{jane}) + \rho_{\text{object}}(\mathbf{spot}) \rangle; \tag{28}$$

$$\mathbf{CAUSE} = \langle \mathbf{cause} + \mathbf{BITE} + \mathbf{FLEE} + \rho_{\text{antecedent}}(\mathbf{BITE}) + \rho_{\text{consequent}}(\mathbf{FLEE}) \rangle. \tag{29}$$

Note that in the last six equations $\langle \cdot \rangle$ refers to the CDT procedure. These types of graph representations by HVs will be further discussed in Section 3.1.2 of Part II of the survey [84] when describing analogical reasoning with HDC/VSA.

Representations of knowledge graphs were further studied in [109]. The work proposed to use a Cauchy distribution to generate atomic HVs and demonstrated the state-of-the-art results on the task of inferring missing links using HVs of knowledge graphs as an input to a neural network. HVs of knowledge graphs can also be constructed using HVs of nodes and relations that are learned from data as proposed in [125], where HRR was used due to its differentiability.

3.5.3 Trees. Trees are an instance of graphs where a node at the lower level (child) belongs to a single higher-level node (parent). Therefore, trees can be represented in the same manner as the graphs above. Please refer to the examples of the transformations to HVs given, e.g., in [153] for unordered binary trees and in [24, 78] for ordered binary trees. A representation of trees for the TPR model, using cryptographic hashing algorithms, was investigated in [40].

4 DISCUSSION

In this section, we only discuss of the HDC/VSA aspects covered in this first part of the survey. Please refer to Part II [84] for an extensive discussion of the aspects related to application areas, interplay with neural networks, and open issues.

4.1 Connections Between HDC/VSA Models

As presented in Section 2.3, there are several HDC/VSA models. Currently, there are many unanswered questions about the connections between models. For example, some models use interactions between multiple components of HVs when performing the multiplicative binding operation (e.g., circular convolution or the CDT procedure). Other models (e.g., FHRR and BSC) use multiplicative binding operations that are component-wise. It is not always clear in which situations one is better than the other. Therefore, it is important to develop a more principled understanding of the relations between the multiplicative binding operation and of the scope of their applicability. For example, a recent theoretical result in this direction [28] is that under certain conditions the multiplicative binding in MAP is mathematically equivalent to the multiplicative binding in TPR.

Moreover, in general it is not clear whether there is one best model, which will dominate all others, or whether each model has its own applicability scope. There are works [73, 169] reporting systematic experimental comparisons between the models for aspects such as, e.g., mathematical properties of the operations, information capacity of representations, or robustness to noise. We believe that this line of work should be continued since an abstract mathematical characterization of the necessary properties for HDC/VSA might make it easier to generate new HDC/VSA models as alternative instantiations of such properties. Finally, there is the question of whether there is a need for new HDC/VSA models.

4.2 Theoretical Foundations of HDC/VSA

It should be noted that HDC/VSA have largely started as an empirical field. At the same time, HDC/VSA implicitly used well-known mathematical phenomena such as concentration of measure [28] and RP [183].

Currently, there is a demand for laying out solid theoretical principles for HDC/VSA. We are starting to see promising results in this direction. For example, there is a recent "capacity theory" [27] (Section 2.4), which provides a way of estimating the amount of information that could be reconstructed from HVs using the nearest neighbor search in the item memory. Another example is the upper bound guarantees for the reconstruction of data structures from HVs [183]. In [28] manipulations of HVs in HDC/VSA were shown to be connected to the compressed sensing.

Recently, in [196], the expressivity of HDC/VSA models was characterized using similarity matrices. It proved that BSC could not approximate a given similarity matrix arbitrarily well, and could therefore not learn a Bayes optimal classifier. This work has also formalized HDC/VSA using the notion of finite groups and as a result introduced a new model CGR that generalized BSC, and is equivalent to FHRR in the limit.

Another important aspect that should be studied theoretically is the definition of the binding operation. Currently, there is no agreement on what shall be considered a canonical definition of the binding. Various researchers define it differently. For example, is it compulsory that the

result of binding is dissimilar to its arguments as is, e.g., the case in FHRR or BSC, or shall it preserve some similarity as in SBDR? An attempt to taxonomize the existing proposals can be found in [169], but it does not provide an answer to the question of whether there exists some abstract mathematical definition of a "pure" binding operation and whether it is useful in practice to deviate from this definition and create concrete realizations of the binding operation that would not comply completely with that definition. In that vein, a recent theoretical consideration of multiplicative binding was presented in [49]. We foresee that, as HDC/VSA will be exposed more to theoretical computer scientists and applied mathematicians, we will see more works building the theoretical foundations of the field.

4.3 Implementation of the HDC/VSA Models in Hardware

A large part of the promise of HDC/VSA is based on the fact that they are suitable for implementations on a variety of unconventional hardware such as neuromorphic hardware [29], in-memory computing [70], monolithic 3D integration hardware [105, 192], and so on. It is interesting that the motivation to use HDC/VSA on specialized hardware was present from the very beginning. Early examples are specialized neurocomputers [100] built to operate with SBDR. Moreover, the topic of hardware implementations has received a lot of attention recently. This is mostly due to the fact that modern machine learning algorithms such as deep neural networks require massive resources to train and run them [181]. In our opinion, an additional motivation comes from the fact that HDC/VSA can be seen as an abstraction algorithmic layer and can, thus, be used for designing computational primitives, which can then be mapped to various hardware platforms [78]. Nevertheless, hardware for HDC/VSA is an active research field and a topic of its own, therefore, we have decided to leave it outside the scope of this survey. However, we expect that in the coming years this topic is going to play an important role in the development of the community. As another interesting direction, HDC/VSA can embrace noise and stochasticity of in-memory computing hardware [70, 71, 197] to enhance security as shown in [198] by directly generating time-variable cipher-texts that are yet decryptable.

5 CONCLUSION

In this Part I of the survey, we provided a comprehensive coverage of the computing framework known under the names HDC and VSA. We paid particular attention to existing HDC/VSA models and the transformations of input data of various types into hypervector representations.

Part II of the survey [84] reviews known applications, touches upon cognitive modeling and cognitive architectures, and discusses the open problems along with the most promising directions for the future work.

ACKNOWLEDGMENTS

We would like to thank three reviewers, the editors, and Pentti Kanerva for their insightful feedback as well as Linda Rudin for the careful proofreading that contributed to the final shape of the survey.

REFERENCES

[1] D. Achlioptas. 2003. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of Computer and System Sciences* 66, 4 (2003), 671–687.

[2] D. Aerts and M. Czachor. 2008. Tensor-product versus geometric-product coding. *Physical Review A* 77, 012316 (2008), 1–7.

[3] D. Aerts, M. Czachor, and B. De Moor. 2006. On geometric algebra representation of binary spatter codes. arXiv:0610075. Retrieved from https://arxiv.org/abs/0610075.

[4] D. Aerts, M. Czachor, and B. De Moor. 2009. Geometric analogue of holographic reduced representation. *Journal of Mathematical Psychology* 53, 5 (2009), 389–398.
[5] M. A. Aiserman, E. M. Braverman, and L. I. Rozonoer. 1964. Theoretical foundations of the potential function method in pattern recognition. *Avtomatika i Telemekhanika* 25, 6 (1964), 917–936.
[6] J. S. Albus. 1975. Data storage in the cerebellar model articulation controller. *Journal of Dynamic Systems, Measurement and Control* 97, 3 (1975), 228–233.
[7] A. Becker, L. Ducas, N. Gama, and T. Laarhoven. 2016. New directions in nearest neighbor searching with applications to lattice sieving. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*. 10–24.
[8] B. H. Bloom. 1970. Space/time trade-offs in hash coding with allowable errors. *Communication of the ACM* 13, 7 (1970), 422–426.
[9] P. Blouw, E. Solodkin, P. Thagard, and C. Eliasmith. 2016. Concepts as semantic pointers: A framework and computational model. *Cognitive Science* 40, 5 (2016), 1128–1162.
[10] A. Borsellino and T. Poggio. 1973. Convolution and correlation algebras. *Kybernetik* 13, 2 (1973), 113–122.
[11] A. Burco. 2018. *Exploring Neural-symbolic Integration Architectures for Computer Vision*. Master's thesis. ETH Zurich.
[12] T. Cohen and D. Widdows. 2018. Bringing order to neural word embeddings with embeddings augmented by random permutations. In *Proceedings of the Conference on Computational Natural Language Learning*. 465–475.
[13] T. Cohen, D. Widdows, M. Wahle, and R. W. Schvaneveldt. 2013. Orthogonality and orthography: Introducing measured distance into semantic space. In *Proceedings of the International Symposium on Quantum Interaction,* Lecture Notes in Computer Science, Vol. 8369. 34–46.
[14] I. Danihelka, G. Wayne, B. Uria, N. Kalchbrenner, and A. Graves. 2016. Associative long short-term memory. In *Proceedings of the International Conference on Machine Learning*. 1986–1994.
[15] S. Dasgupta, C. F. Stevens, and S. Navlakha. 2017. A neural algorithm for a fundamental computing problem. *Science* 358, 6364 (2017), 793–796.
[16] D. L. Donoho. 2006. Compressed sensing. *IEEE Transactions on Information Theory* 52, 4 (2006), 1289–1306.
[17] S. Duan, Y. Liu, S. Ren, and X. Xu. 2022. LeHDC: Learning-based hyperdimensional computing classifier. In *Proceedings of the ACM/ESDA/IEEE Design Automation Conference*. 1–6.
[18] M. Eggimann, A. Rahimi A., and L. Benini. 2021. A 5 μW standard cell memory-based configurable hyperdimensional computing accelerator for always-on smart sensing. *IEEE Transactions on Circuits and Systems I: Regular Papers* 68, 10 (2021), 4116–4128.
[19] J. M. Eich. 1982. A composite holographic associative recall model. *Psychological Review* 89, 6 (1982), 627–661.
[20] C. Eliasmith. 2013. *How to build a brain: A neural architecture for biological cognition*. Oxford University Press. 456.
[21] C. Eliasmith, T. C. Stewart, X. Choo, T. Bekolay, T. DeWolf, Y. Tang, and D. Rasmussen. 2012. A large-scale model of the functioning brain. *Science* 338, 6111 (2012), 1202–1205.
[22] J. A. Fodor and Z. W. Pylyshyn. 1988. Connectionism and cognitive architecture: A critical analysis. *Cognition* 28, 1–2 (1988), 3–71.
[23] E. P. Frady, S. J. Kent, P. Kanerva, B. A. Olshausen, and F. T. Sommer. 2018. Cognitive neural systems for disentangling compositions. In *Proceedings of the Cognitive Computing*. 1–3.
[24] E. P. Frady, S. J. Kent, B. A. Olshausen, and F. T. Sommer. 2020. Resonator networks, 1: An efficient solution for factoring high-dimensional, distributed representations of data structures. *Neural Computation* 32, 12 (2020), 2311–2331.
[25] E. P. Frady, D. Kleyko, C. J. Kymn, B. A. Olshausen, and F. T. Sommer. 2021. Computing on functions using randomized vector representations. arXiv:2109.03429 (2021), 1–33.
[26] E. P. Frady, D. Kleyko, C. J. Kymn, B. A. Olshausen, and F. T. Sommer. 2022. Computing on functions using randomized vector representations (in brief). In *Proceedings of the Neuro-Inspired Computational Elements Conference*. 115–122.
[27] E. P. Frady, D. Kleyko, and F. T. Sommer. 2018. A theory of sequence indexing and working memory in recurrent neural networks. *Neural Computation* 30, 6 (2018), 1449–1513.
[28] E. P. Frady, D. Kleyko, and F. T. Sommer. 2021. Variable binding for sparse distributed representations: Theory and applications. *IEEE Transactions on Neural Networks and Learning Systems* 99, (2021), 1–14.
[29] E. P. Frady and F. T. Sommer. 2019. Robust computation with rhythmic spike patterns. *Proceedings of the National Academy of Sciences* 116, 36 (2019), 18050–18059.
[30] S. I. Gallant and P. Culliton. 2016. Positional binding with distributed representations. In *Proceedings of the International Conference on Image, Vision and Computing*. 108–113.
[31] S. I. Gallant and T. W. Okaywe. 2013. Representing objects, relations, and sequences. *Neural Computation* 25, 8 (2013), 2038–2078.
[32] A. Ganesan, H. Gao, S. Gandhi, E. Raff, T. Oates, J. Holt, and M. McLean. 2021. Learning with holographic reduced representations. In *Proceedings of the Advances in Neural Information Processing Systems*. 1–15.
[33] R. W. Gayler. 1998. Multiplicative binding, representation operators & analogy. *Advances in Analogy Research: Integration of Theory and Data from the Cognitive, Computational, and Neural Sciences*. 1–4.

[34] R. W. Gayler. 2003. Vector symbolic architectures answer jackendoff's challenges for cognitive neuroscience. In *Proceedings of the Joint International Conference on Cognitive Science*. 133–138.
[35] R. W. Gayler and S. D. Levy. 2009. A distributed basis for analogical mapping: New frontiers in analogy research. In *New frontiers in Analogy Research, Second International Conference on the Analogy*. 165–174.
[36] L. Ge and K. K. Parhi. 2020. Classification using hyperdimensional computing: A review. *IEEE Circuits and Systems Magazine* 20, 2 (2020), 30–47.
[37] A. N. Gorban and I. Y. Tyukin. 2018. Blessing of dimensionality: Mathematical foundations of the statistical physics of data. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 376, 2118 (2018), 1–18.
[38] J. Gosmann and C. Eliasmith. 2019. Vector-derived transformation binding: An improved binding operation for deep symbol-like processing in neural networks. *Neural Computation* 31, 5 (2019), 849–869.
[39] V. I. Gritsenko, D. A. Rachkovskij, A. A. Frolov, R. W. Gayler, D. Kleyko, and E. Osipov. 2017. Neural distributed autoassociative memories: A survey. *Cybernetics and Computer Engineering* 2, 188 (2017), 5–35.
[40] C. Haley and P. Smolensky. 2020. Invertible tree embeddings using a cryptographic role embedding scheme. In *Proceedings of the International Conference on Computational Linguistics*. 3671–3683.
[41] T. Hannagan, E. Dupoux, and A. Christophe. 2011. Holographic string encoding. *Cognitive Science* 35, 1 (2011), 79–118.
[42] S. Harnad. 1990. The symbol grounding problem. *Physica D: Nonlinear Phenomena* 42, 1–3 (1990), 335–346.
[43] E. Hassan, Y. Halawani, B. Mohammad, and H. Saleh. 2021. Hyper-dimensional computing challenges and opportunities for AI applications. *IEEE Access* (2021), 1–15.
[44] M. Hersche, G. Karunaratne, G. Cherubini, L. Benini, A. Sebastian, and A. Rahimi. 2022. Constrained few-shot class-incremental learning. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*. 1–19.
[45] M. Hersche, S. Lippuner, M. Korb, L. Benini, and A. Rahimi. 2021. Near-channel classifier: Symbiotic communication and classification in high-dimensional space. *Brain Informatics* 8, 1 (2021), 1–15.
[46] G. E. Hinton. 1981. Implementing semantic networks in parallel hardware. *Parallel Models of Association Memory*. Erlbaum Associates, 191–217.
[47] G. E. Hinton. 1990. Mapping part-whole hierarchies into connectionist networks. *Artificial Intelligence* 46, 1–2 (1990), 47–75.
[48] G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. 1986. Distributed representations. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*. 77–109.
[49] N. Hiratani and H. Sompolinsky. 2022. Optimal quadratic binding for relational reasoning in vector symbolic neural architectures. arXiv:2204.07186. Retrieved from https://arxiv.org/abs/2204.07186.
[50] J. E. Hummel and K. J. Holyoak. 1997. Distributed representations of structure: A theory of analogical access and mapping. *Psychological Review* 104, 3 (1997), 427–466.
[51] M. Imani, T. Nassar, A. Rahimi, and T. Rosing. 2018. HDNA: Energy-efficient DNA sequencing using hyperdimensional computing. In *Proceedings of the IEEE International Conference on Biomedical and Health Informatics*. 271–274.
[52] P. Indyk and R. Motwani. 1998. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Annual ACM Symposium on Theory of Computing*. 604–613.
[53] R. Jackendoff. 2002. *Foundations of Language: Brain, Meaning, Grammar, Evolution*. Oxford University Press.
[54] W. B. Johnson and J. Lindenstrauss. 1984. Extensions of lipschitz mapping into hilbert space. *Contemporary Mathematics* 26 (1984), 189–206.
[55] M. N. Jones and D. J. K. Mewhort. 2007. Representing word meaning and order information in a composite holographic lexicon. *Psychological Review* 114, 1 (2007), 1–37.
[56] A. Joshi, J. T. Halseth, and P. Kanerva. 2016. Language geometry using random indexing. In *Proceedings of the International Symposium on Quantum Interaction*. 265–274.
[57] G. Kachergis, G. E. Cox, and M. N. Jones. 2011. OrBEAGLE: Integrating orthography into a holographic model of the lexicon. In *Proceedings of the International Conference on Artificial Neural Networks*. 307–314.
[58] D. M. Kane and J. Nelson. 2014. Sparser Johnson-Lindenstrauss transforms. *Journal of the ACM* 61, 1 (2014), 1–23.
[59] P. Kanerva. 1988. *Sparse Distributed Memory*. The MIT Press. 155.
[60] P. Kanerva. 1994. The spatter code for encoding concepts at many levels. In *Proceedings of the International Conference on Artificial Neural Networks*. 226–229.
[61] P. Kanerva. 1995. A family of binary spatter codes. In *Proceedings of the International Conference on Artificial Neural Networks*. 517–522.
[62] P. Kanerva. 1996. Binary spatter-coding of ordered K-tuples. In *Proceedings of the International Conference on Artificial Neural Networks,* Lecture Notes in Computer Science, Vol. 1112. 869–873.
[63] P. Kanerva. 1997. Fully distributed representation. In *Proceedings of the Real World Computing Symposium*. 358–365.
[64] P. Kanerva. 1998. Dual role of analogy in the design of a cognitive computer. *Advances in Analogy Research: Integration of Theory and Data from the Cognitive, Computational, and Neural Sciences*. 164–170.

[65] P. Kanerva. 1998. Encoding structure in boolean space. In *Proceedings of the International Conference on Artificial Neural Networks*. 387–392.
[66] P. Kanerva. 2009. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation* 1, 2 (2009), 139–159.
[67] P. Kanerva. 2014. Computing with 10,000-bit words. In *Proceedings of the Annual Allerton Conference on Communication, Control, and Computing*. 1–7.
[68] P. Kanerva. 2019. Computing with high-dimensional vectors. *IEEE Design & Test* 36, 3 (2019), 7–14.
[69] P. Kanerva, J. Kristoferson, and A. Holst. 2000. Random indexing of text samples for latent semantic analysis. In *Proceedings of the Annual Meeting of the Cognitive Science Society*. 1036.
[70] G. Karunaratne, M. Le Gallo, G. Cherubini, L. Benini, A. Rahimi, and A. Sebastian. 2020. In-memory hyperdimensional computing. *Nature Electronics* 3, 6 (2020), 327–337.
[71] G. Karunaratne, M. Schmuck, M. Le Gallo, G. Cherubini, L. Benini, A. Sebastian, and A. Rahimi. 2021. Robust high-dimensional memory-augmented neural networks. *Nature Communications* 12, 1 (2021), 1–12.
[72] Samuel Kaski. 1998. Dimensionality reduction by random mapping: Fast similarity computation for clustering. In *Proceedings of the International Joint Conference on Neural Networks*, Vol. 1. 413–418.
[73] M. A. Kelly, D. Blostein, and D. J. K. Mewhort. 2013. Encoding structure in holographic reduced representations. *Canadian Journal of Experimental Psychology* 67, 2 (2013), 79–93.
[74] S. J. Kent, E. P. Frady, F. T. Sommer, and B. A. Olshausen. 2020. Resonator networks, 2: Factorization performance and capacity compared to optimization-based methods. *Neural Computation* 32, 12 (2020), 2332–2388.
[75] H.-S. Kim. 2018. HDM: Hyper-dimensional modulation for robust low-power communications. In *Proceedings of the IEEE International Conference on Communications*. 1–6.
[76] Y. Kim, M. Imani, N. Moshiri, and T. Rosing. 2020. GenieHD: Efficient DNA pattern matching accelerator using hyperdimensional computing. In *Proceedings of the Design, Automation Test in Europe Conference Exhibition*. 115–120.
[77] D. Kleyko, C. Bybee, P.-C. Huang, C. J. Kymn, B. A. Olshausen, F. T. Sommer, and E. P. Frady. 2022. Efficient decoding of compositional structure in holistic representations. *arXiv* (2022).
[78] D. Kleyko, M. Davies, E. P. Frady, P. Kanerva, S. J. Kent, B. A. Olshausen, E. Osipov, J. M. Rabaey, D. A. Rachkovskij, A. Rahimi, and F. T. Sommer. 2021. Vector symbolic architectures as a computing framework for nanoscale hardware. arXiv:2106.05268. Retrieved from https://arxiv.org/abs/2106.05268.
[79] D. Kleyko, E. P. Frady, and F. T. Sommer. 2021. Cellular automata can reduce memory requirements of collective-state computing. *IEEE Transactions on Neural Networks and Learning Systems* 33, 6 (2022), 2701–2713.
[80] D. Kleyko, R. W. Gayler, and E. Osipov. 2020. Commentaries on "Learning Sensorimotor Control with Neuromorphic Sensors: Toward Hyperdimensional Active Perception." *Science Robotics* 4, 30 (2019), arXiv:2003.11458. Retrieved from https://arxiv.org/abs/2003.11458.
[81] D. Kleyko, E. Osipov, and R. W. Gayler. 2016. Recognizing permuted words with vector symbolic architectures: A cambridge test for machines. *Procedia Computer Science* 88 (2016), 169–175.
[82] D. Kleyko, E. Osipov, and D. A. Rachkovskij. 2016. Modification of holographic graph neuron using sparse distributed representations. *Procedia Computer Science* 88 (2016), 39–45.
[83] D. Kleyko, E. Osipov, A. Senior, A. I. Khan, and Y. A. Sekercioglu. 2017. Holographic graph neuron: A bioinspired architecture for pattern processing. *IEEE Transactions on Neural Networks and Learning Systems* 28, 6 (2017), 1250–1262.
[84] D. Kleyko, D. A. Rachkovskij, E. Osipov, and A. Rahimi. 2021. A survey on Hyperdimensional Computing aka Vector Symbolic Architectures, Part II: Applications, cognitive models, and challenges. arXiv:2112.15424. Retrieved from https://arxiv.org/abs/2112.15424.
[85] D. Kleyko, A. Rahimi, R. W. Gayler, and E. Osipov. 2020. Autoscaling Bloom filter: Controlling trade-off between true and false positives. *Neural Computing and Applications* 32, 8 (2020), 3675–3684.
[86] D. Kleyko, A. Rahimi, D. A. Rachkovskij, E. Osipov, and J. M. Rabaey. 2018. Classification and recall with binary hyperdimensional computing: Tradeoffs in choice of density and mapping characteristic. *IEEE Transactions on Neural Networks and Learning Systems* 29, 12 (2018), 5880–5898.
[87] D. Kleyko, A. Rosato, E. P. Frady, M. Panella, and F. T. Sommer. 2020. Perceptron theory for predicting the accuracy of neural networks. arXiv:2012.07881. Retrieved from https://arxiv.org/abs/2012.07881.
[88] B. Komer. 2020. Biologically inspired spatial representation. University of Waterloo, PhD Thesis.
[89] B. Komer and C. Eliasmith. 2020. Efficient navigation using a scalable, biologically inspired spatial representation. In *Proceedings of the Annual Meeting of the Cognitive Science Society*. 1532–1538.
[90] B. Komer, T. C. Stewart, A. R. Voelker, and C. Eliasmith. 2019. A neural representation of continuous space using fractional binding. In *Proceedings of the Annual Meeting of the Cognitive Science Society*. 2038–2043.
[91] E. M. Kussul and T. N. Baidyk. 1993. On information encoding in associative-projective neural networks. Technical Report. Report 93-3, V. M. Glushkov Institute of Cybernetics (in Russian). 1–18.

[92] E. M. Kussul and T. N. Baidyk. 2003. Permutative coding technique for handwritten digit recognition system. In *Proceedings of the International Joint Conference on Neural Networks*. 2163–2168.
[93] E. M. Kussul and T. N. Baidyk. 2004. Improved method of handwritten digit recognition tested on MNIST database. *Image and Vision Computing* 22, 12 (2004), 971–981.
[94] E. M. Kussul, T. N. Baidyk, V. V. Lukovich, and D. A. Rachkovskij. 1994. Adaptive high performance classifier based on random threshold neurons. In *Proceedings of the European Meeting on Cybernetics and Systems*. 1687–1694.
[95] E. M. Kussul, T. N. Baidyk, and D. A. Rachkovskij. 1992. Neural network for recognition of small images. In *Proceedings of the 1st All-Ukrainian Conference UkrOBRAZ*. 151–153.
[96] E. M. Kussul, T. N. Baidyk, and D. C. Wunsch. 2010. *Neural Networks and Micromechanics*. Springer.
[97] E. M. Kussul, T. N. Baidyk, D. C. Wunsch, O. Makeyev, and A. Martin. 2006. Permutation coding technique for image recognition system. *IEEE Transactions on Neural Networks* 17, 6 (2006), 1566–1579.
[98] E. M. Kussul and D. A. Rachkovskij. 1991. Multilevel assembly neural architecture and processing of sequences. *Neurocomputers and Attention: Connectionism and Neurocomputers*, Vol. 2. 577–590.
[99] E. M. Kussul, D. A. Rachkovskij, and T. N. Baidyk. 1991. Associative-projective neural networks: Architecture, implementation, applications. In *Proceedings of the International Conference on Neural Networks and Their Applications*. 463–476.
[100] E. M. Kussul, D. A. Rachkovskij, and T. N. Baidyk. 1991. On image texture recognition by associative-projective neurocomputer. *Intelligent Engineering Systems through Artificial Neural Networks*. 453–458.
[101] E. M. Kussul, D. A. Rachkovskij, and D. C. Wunsch. 1999. The random subspace coarse coding scheme for real-valued vectors. In *Proceedings of the International Joint Conference on Neural Networks*, Vol. 1. 450–455.
[102] M. Laiho, J. H. Poikonen, P. Kanerva, and E. Lehtonen. 2015. High-dimensional computing with sparse vectors. In *Proceedings of the IEEE Biomedical Circuits and Systems Conference*. 1–4.
[103] M. Ledoux. 2001. The concentration of measure phenomenon. American Mathematical Society.
[104] S. D. Levy and R. W. Gayler. 2008. Vector symbolic architectures: A new building material for artificial general intelligence. In *Proceedings of the Artificial General Intelligence*. 414–418.
[105] H. Li, T. F. Wu, A. Rahimi, K.-S. Li, M. Rusch, C.-H. Lin, J.-L. Hsu, M. M. Sabry, S. B. Eryilmaz, J. Sohn, W.-C. Chiu, M.-C. Chen, T.-T. Wu, J.-M. Shieh, W.-K. Yeh, J. M. Rabaey, S. Mitra, and H.-S. P. Wong. 2016. Hyperdimensional computing with 3D VRRAM in-memory kernels: Device-architecture co-design for energy-efficient, error-resilient language recognition. In *Proceedings of the IEEE International Electron Devices Meeting*. 1–4.
[106] P. Li, T. J. Hastie, and K. W. Church. 2006. Very sparse random projections. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*. 287–296.
[107] D. Liang, J. Shiomi, N. Miura, and H. Awano. 2022. DistriHD: A memory efficient distributed binary hyperdimensional computing architecture for image classification. In *Proceedings of the Asia and South Pacific Design Automation Conference*. 43–49.
[108] H. C. Longuet-Higgins. 1968. Holographic model of temporal recall. *Nature* 217, 5123 (1968), 104.
[109] Y. Ma, M. Hildebrandt, V. Tresp, and S. Baier. 2018. Holistic representations for memorization and inference. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. 1–11.
[110] A. X. Manabat, C. R. Marcelo, A. L. Quinquito, and A. Alvarez. 2019. Performance analysis of hyperdimensional computing for character recognition. In *Proceedings of the International Symposium on Multimedia and Communication Technology*. 1–5.
[111] P. M. Milner. 1974. A model for visual shape recognition. *Psychological Review* 81, 6 (1974), 521–535.
[112] F. Mirus, T. C. Stewart, and J. Conradt. 2020. Analyzing the capacity of distributed vector representations to encode spatial information. In *Proceedings of the International Joint Conference on Neural Networks*. 1–7.
[113] I. S. Misuno, D. A. Rachkovskij, and S. V. Slipchenko. 2005. Vector and distributed representations reflecting semantic relatedness of words. *Mathematical Machines and Systems*. 3 (2005), 50–66.
[114] I. S. Misuno, D. A. Rachkovskij, S. V. Slipchenko, and A. M. Sokolov. 2005. Searching for text information with the help of vector representations. *Problems of Programming*. 4 (2005), 50–59.
[115] A. Mitrokhin, P. Sutor, C. Fermuller, and Y. Aloimonos. 2019. Learning sensorimotor control with neuromorphic sensors: Toward hyperdimensional active perception. *Science Robotics* 4, 30 (2019), 1–10.
[116] A. Mitrokhin, P. Sutor, D. Summers-Stay, C. Fermuller, and Y. Aloimonos. 2020. Symbolic representation and learning with hyperdimensional computing. *Frontiers in Robotics and AI* 7 (2020), 1–11.
[117] E. Mizraji. 1989. Context-dependent associations in linear distributed memories. *Bulletin of Mathematical Biology* 51, 2 (1989), 195–205.
[118] E. Mizraji. 1992. Vector logics: The matrix-vector representation of logical calculus fuzzy sets and systems. *Bulletin of Mathematical Biology* 50, 2 (1992), 179–185.
[119] J. Moody and C. J. Darken. 1989. Fast learning in networks of locally-tuned processing units. *Neural Computation* 1, 2 (1989), 281–294.

[120] B. B. Murdock. 1982. A theory for the storage and retrieval of item and associative information. *Psychological Review* 89, 6 (1982), 609–626.
[121] P. Neubert and P. Protzel. 2018. Towards hypervector representations for learning and planning with schemas. In *Proceedings of the Joint German/Austrian Conference on Artificial Intelligence,* Lecture Notes in Computer Science, Vol. 11117. 182–189.
[122] P. Neubert, S. Schubert, and P. Protzel. 2019. An introduction to hyperdimensional computing for robotics. *KI - Künstliche Intelligenz* 33, 4 (2019), 319–330.
[123] J. Neumann. 2002. Learning the systematic transformation of holographic reduced representations. *Cognitive Systems Research* 3, 2 (2002), 227–235.
[124] A. Newell and H. A. Simon. 1976. Computer science as empirical inquiry: Symbols and search. *Communications of the Association for Computing Machinery* 19, 3 (1976), 113–126.
[125] M. Nickel, L. Rosasco, and T. Poggio. 2016. Holographic embeddings of knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence.* 1955–1961.
[126] C. H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala. 2000. Latent semantic indexing: A probabilistic analysis. *Journal of Computer and System Sciences* 61, 2 (2000), 217–235.
[127] A. Patyk-Lonska. 2010. Geometric algebra model of distributed representation. In *Proceedings of the Geometric Algebra Computing*. 401–430.
[128] A. Patyk-Lonska. 2011. Experiments on preserving pieces of information in a given order in holographic reduced representations and the continuous geometric algebra model. *Informatica* 35, 4 (2011), 419–427.
[129] A. Patyk-Lonska. 2011. Preserivng pieces of information in a given order in HRR and GAc. In *Proceedings of the Federated Conference on Computer Science and Information Systems.* 213–220.
[130] A. Patyk-Lonska, M. Czachor, and D. Aerts. 2011. A comparison of geometric analogues of holographic reduced representations, original holographic reduced representations and binary spatter codes. In *Proceedings of the Federated Conference on Computer Science and Information Systems.* 221–228.
[131] A. Patyk-Lonska, M. Czachor, and D. Aerts. 2011. Distributed representations based on geometric algebra: The continuous model. *Informatica* 35, 4 (2011), 407–417.
[132] T. A. Plate. 1991. Holographic reduced representations: Convolution algebra for compositional distributed representations. In *Proceedings of the International Joint Conference on Artificial Intelligence.* 30–35.
[133] T. A. Plate. 1992. Holographic recurrent networks. In *Proceedings of the Advances in Neural Information Processing Systems.* 34–41.
[134] T. A. Plate. 1994. *Distributed representations and nested compositional structure.* University of Toronto, PhD Thesis.
[135] T. A. Plate. 1995. Holographic reduced representations. *IEEE Transactions on Neural Networks* 6, 3 (1995), 623–641.
[136] T. A. Plate. 1995. *Networks which learn to store variable-length sequences in a fixed set of unit activations.* Citeseer.
[137] T. A. Plate. 1997. A common framework for distributed representation schemes for compositional structure. *Connectionist Systems for Knowledge Representation and Deduction.* 15–34.
[138] T. A. Plate. 2000. Analogy retrieval and processing with distributed vector representations. *Expert Systems: The International Journal of Knowledge Engineering and Neural Networks* 17, 1 (2000), 29–40.
[139] T. A. Plate. 2003. *Holographic reduced representations: distributed representation for cognitive structures.* Stanford: Center for the Study of Language and Information.
[140] T. A. Plate. 2006. Distributed representations. *Encyclopedia of Cognitive Science.* 1–9.
[141] R. W. Prager. 1993. Networks based on Kanerva's sparse distributed memory: Results showing their strengths and limitations and a new algorithm to design the location matching layer. In *Proceedings of the IEEE International Conference on Neural Networks.* 1040–1045.
[142] S. Purdy. 2016. Encoding data for HTM systems. arXiv:1602.05925. Retrieved from https://arxiv.org/abs/1602.05925.
[143] R. Q. Quiroga, L. Reddy, G. Kreiman, C. Koch, and I. Fried. 2005. Invariant visual representation by single neurons in the human brain. *Nature* 435, 7045 (2005), 1102–1107.
[144] D. A. Rachkovskij. 1990. *Development and investigation of multilevel assembly neural networks.* Glushkov Institute of Cybernetics, PhD Thesis.
[145] D. A. Rachkovskij. 1996. Application of stochastic assembly neural networks in the problem of interesting text selection. *Neural Network Systems for Information Processing* (1996), 52–64.
[146] D. A. Rachkovskij. 2001. Representation and processing of structures with binary sparse distributed codes. *IEEE Transactions on Knowledge and Data Engineering* 3, 2 (2001), 261–276.
[147] D. A. Rachkovskij. 2014. Vector data transformation using random binary matrices. *Cybernetics and Systems Analysis* 50, 6 (2014), 960–968.
[148] D. A. Rachkovskij. 2015. Formation of similarity-reflecting binary vectors with random binary projections. *Cybernetics and Systems Analysis* 51, 2 (2015), 313–323.

[149] D. A. Rachkovskij. 2021. Shift-equivariant similarity-preserving hypervector representations of sequences. arXiv: 2112.15475. Retrieved from https://arxiv.org/abs/2112.15475.

[150] D. A. Rachkovskij and T. V. Fedoseyeva. 1990. On audio signals recognition by multilevel neural network. In *Proceedings of the International Symposium on Neural Networks and Neural Computing*. 281–283.

[151] D. A. Rachkovskij and V. I. Gritsenko. 2018. *Distributed representation of vector data based on random projections*. Interservice.

[152] D. A. Rachkovskij and D. Kleyko. 2022. Recursive binding for similarity-preserving hypervector representations of sequences. In *Proceedings of the International Joint Conference on Neural Networks*. 1–8.

[153] D. A. Rachkovskij and E. M. Kussul. 2001. Binding and normalization of binary sparse distributed representations by context-dependent thinning. *Neural Computation* 13, 2 (2001), 411–452.

[154] D. A. Rachkovskij, E. M. Kussul, and T. N. Baidyk. 2013. Building a world model with structure-sensitive sparse binary distributed representations. *Biologically Inspired Cognitive Architectures* 3 (2013), 64–86.

[155] D. A. Rachkovskij, I. S. Misuno, and S. V. Slipchenko. 2012. Randomized projective methods for the construction of binary sparse vector representations. *Cybernetics and Systems Analysis* 48, 1 (2012), 146–156.

[156] D. A. Rachkovskij, S. V. Slipchenko, A. A. Frolov, and D. Husek. 2005. Resolution of binary coding of real-valued vectors by hyperrectangular receptive fields. *Cybernetics and Systems Analysis* 41, 5 (2005), 635–646.

[157] D. A. Rachkovskij, S. V. Slipchenko, E. M. Kussul, and T. N. Baidyk. 2005. Properties of numeric codes for the scheme of random subspaces RSC. *Cybernetics and Systems Analysis* 41, 4 (2005), 509–520.

[158] D. A. Rachkovskij, S. V. Slipchenko, E. M. Kussul, and T. N. Baidyk. 2005. Sparse binary distributed encoding of scalars. *Journal of Automation and Information Sciences* 37, 6 (2005), 12–23.

[159] D. A. Rachkovskij, S. V. Slipchenko, I. S. Misuno, E. M. Kussul, and T. N. Baidyk. 2005. Sparse binary distributed encoding of numeric vectors. *Journal of Automation and Information Sciences* 37, 11 (2005), 47–61.

[160] A. Rahimi, S. Benatti, P. Kanerva, L. Benini, and J. M. Rabaey. 2016. Hyperdimensional biosignal processing: A case study for EMG-based hand gesture recognition. In *Proceedings of the IEEE International Conference on Rebooting Computing*. 1–8.

[161] A. Rahimi, S. Datta, D. Kleyko, E. P. Frady, B. Olshausen, P. Kanerva, and J. M. Rabaey. 2017. High-dimensional computing as a nanoscalable paradigm. *IEEE Transactions on Circuits and Systems I: Regular Papers* 64, 9 (2017), 2508–2521.

[162] A. Rahimi, P. Kanerva, L. Benini, and J. M. Rabaey. 2019. Efficient biosignal processing using hyperdimensional computing: Network templates for combined learning and classification of ExG signals. *Proceedings of the IEEE* 107, 1 (2019), 123–143.

[163] O. Räsänen. 2015. Generating hyperdimensional distributed representations from continuous valued multivariate sensory input. In *Proceedings of the Annual Meeting of the Cognitive Science Society*. 1943–1948.

[164] G. Recchia, M. N. Jones, M. Sahlgren, and P. Kanerva. 2010. Encoding sequential information in vector space models of semantics: Comparing holographic reduced representation and random permutation. In *Proceedings of the Annual Meeting of the Cognitive Science Society*. 865–870.

[165] G. Recchia, M. Sahlgren, P. Kanerva, and M. N. Jones. 2015. Encoding sequential information in semantic space models: Comparing holographic reduced representation and random permutation. *Computational Intelligence and Neuroscience* (2015), 1–18.

[166] S. Reimann. 2022. A novel HD computing algebra: Non-associative superposition of states creating sparse bundles representing order information. arXiv:2202.08633. Retrieved from https://arxiv.org/abs/2202.08633.

[167] M. Sahlgren. 2005. An introduction to random indexing. In *Proceedings of the International Conference on Terminology and Knowledge Engineering*. 1–9.

[168] M. Sahlgren, A. Holst, and P. Kanerva. 2008. Permutations as a means to encode order in word space. In *Proceedings of the Annual Meeting of the Cognitive Science Society*. 1300–1305.

[169] K. Schlegel, P. Neubert, and P. Protzel. 2021. A comparison of vector symbolic architectures. *Artificial Intelligence Review* (2021), 1–33.

[170] M. Schmuck, L. Benini, and A. Rahimi. 2019. Hardware optimizations of dense binary hyperdimensional computing: Rematerialization of hypervectors, binarized bundling, and combinational associative memory. *ACM Journal on Emerging Technologies in Computing Systems* 15, 4 (2019), 1–25.

[171] P. H. Schönemann. 1987. Some algebraic relations between involutions, convolutions, and correlations, with applications to holographic memories. *Biological Cybernetics* 56, 5–6 (1987), 367–374.

[172] L. Shastri and V. Ajjanagadde. 1993. From simple associations to systematic reasoning: A connectionist representation of rules, variables and dynamic bindings using temporal synchrony. *Behavioral and Brain Sciences* 16, 3 (1993), 417–494.

[173] D. Smith and P. Stanford. 1990. A random walk in hamming space. In *Proceedings of the International Joint Conference on Neural Networks*, Vol. 2. 465–470.

[174] P. Smolensky. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence* 46, 1–2 (1990), 159–216.
[175] J. Snaider and S. Franklin. 2014. Modular composite representation. *Cognitive Computation* 6, 3 (2014), 510–527.
[176] A. M. Sokolov and D. A. Rachkovskij. 2006. Approaches to sequence similarity representation. *Information Theories and Applications* 13, 3 (2006), 272–278.
[177] P. Stanford and D. Smith. 1994. Multidimensional scatter code: A data fusion technique with exponential capacity. In *Proceedings of the International Conference on Artificial Neural Networks*, Vol. 2. 1432–1435.
[178] J. Steinberg and H. Sompolinsky. 2022. Associative memory of structured knowledge. *bioRxiv*. Cold Spring Harbor Laboratory
[179] T. C. Stewart, X. Choo, and C. Eliasmith. 2014. Sentence processing in spiking neurons: A biologically plausible left-corner parser. In *Proceedings of the Annual Meeting of the Cognitive Science Society*. 1533–1538.
[180] T. C. Stewart, Y. Tang, and C. Eliasmith. 2011. A biologically realistic cleanup memory: autoassociation in spiking neurons. *Cognitive Systems Research* 12, 2 (2011), 84–92.
[181] E. Strubell, A. Ganesh, and A. McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. 3645–3650.
[182] D. Summers-Stay, P. Sutor, and D. Li. 2018. Representing sets as summed semantic vectors. *Biologically Inspired Cognitive Architectures* 25 (2018), 113–118.
[183] A. Thomas, S. Dasgupta, and T. Rosing. 2021. A theoretical perspective on hyperdimensional computing. *Journal of Artificial Intelligence Research* 72 (2021), 215–249.
[184] S. J. Thorpe. 2003. Localized versus distributed representations. *Handbook of Brain Theory and Neural Networks*. MIT Press, 643–646.
[185] M. D. Tissera and M. D. McDonnell. 2014. Enabling "Question Answering" in the MBAT vector symbolic architecture by exploiting orthogonal random matrices. In *Proceedings of the IEEE International Conference on Semantic Computing*. 171–174.
[186] T. van Gelder. 1999. Distributed vs. local representation. *MIT Encyclopedia of the Cognitive Sciences*. 235–237.
[187] S. S. Vempala. 2005. *The Random Projection Method*. Vol. 65. American Mathematical Society.
[188] C. von der Malsburg. 1986. Am I thinking assemblies? *Brain Theory*. 161–176.
[189] N. J. Wang, J. Quek, T. M. Rafacz, and S. J. Patel. 2004. Characterizing the effects of transient faults on a high-performance processor pipeline. In *Proceedings of the International Conference on Dependable Systems and Networks*. 61–70.
[190] E. Weiss, B. Cheung, and B. A. Olshausen. 2016. A neural architecture for representing and reasoning about spatial relationships. OpenReview Preprint, 4.
[191] D. Widdows and T. Cohen. 2015. Reasoning with vectors: A continuous model for fast robust inference. *Logic Journal of the IGPL* 23, 2 (2015), 141–173.
[192] T. F. Wu, H. Li, P.-C. Huang, A. Rahimi, G. Hills, B. Hodson, W. Hwang, J. M. Rabaey, H.-S. P. Wong, M. M. Shulaker, and S. Mitra. 2018. Hyperdimensional computing exploiting carbon nanotube FETs, resistive RAM, and their monolithic 3D integration. *IEEE Journal of Solid-State Circuits* 53, 11 (2018), 3183–3196.
[193] T. Yerxa, A. G. Anderson, and E. Weiss. 2018. The hyperdimensional stack machine. In *Proceedings of the Cognitive Computing*. 1–2.
[194] O. Yilmaz. 2015. Machine learning using cellular automata based feature expansion and reservoir computing. *Journal of Cellular Automata* 10, 5–6 (2015), 435–472.
[195] O. Yilmaz. 2015. Symbolic computation using cellular automata-based hyperdimensional computing. *Neural Computation* 27, 12 (2015), 2661–2692.
[196] T. Yu, Y. Zhang, Z. Zhang, and C. De Sa. 2022. Understanding Hyperdimensional Computing for parallel single-pass learning. arXiv:2202.04805. Retrieved from https://arxiv.org/abs/2202.04805.
[197] D. Kleyko, G. Karunaratne, J. M. Rabaey, A. Sebastian, and A. Rahimi. 2022. Generalized key-value memory to flexibly adjust redundancy in memory-augmented networks. *IEEE Transactions on Neural Networks and Learning Systems* 99, PP (2022), 1–6.
[198] J. Cai, A. Amirsoleimani, and R. Genov. 2022. HYPERLOCK: In-Memory Hyperdimensional Encryption in Memristor Crossbar Array. 1–5. arXiv:2201.11362.

Received 8 November 2021; revised 30 April 2022; accepted 6 May 2022