The Blessing of Dimensionality

Perspectives of Reasoning and Learning on Hyperdimensional Computing/Vector Symbolic Architectures

Nicola Fanizzi [a] and Claudia d'Amato [a,*]

[a] *CILA & Dipartimento di Informatica, Università degli Studi di Bari Aldo Moro, Italy*
E-mails: nicola.fanizzi@uniba.it, claudia.damato@uniba.it

Abstract. The paper surveys ongoing research on hyperdimensional computing and vector symbolic architectures which represent an alternative approach to neural computing with various advantages and interesting specific properties: transparency, error tolerance, sustainability. In particular, it can be demonstrated that hyperdimensional patterns are well-suited for the encoding of complex knowledge structures. Consequently, the related architectures offer perspectives for the development of innovative neurosymbolic models with a closer correspondence to cognitive processes in human brains. We revisit the fundamentals of hyperdimensional representations and examine some recent applications of related methods for analogical reasoning and learning tasks, with a particular focus on knowledge graphs. We then propose potential extensions and delineate prospective avenues for future investigations.

1. Introduction and Motivations

Artificial neural networks (ANNs) have been instrumental in the development of numerous successful deep-learning applications. Nevertheless, the increasing complexity of these systems has given rise to a number of challenges. It is extremely difficult to map their internal operations to the cognitive mechanisms that regulate reasoning and learning in the human brain, particularly in terms of the underlying representation of objects, their abstractions, and the relationships between them. Consequently, the transparency of the decisions made is hindered by the difficulty in their interpretation and justification [1]. Furthermore, the considerable and continuously increasing computational resources required to operate these systems have become a significant concern for sustainability [2, 3].

The limitations of sub-symbolic approaches have long been patent. For example, considering an ANN trained to recognize objects of different shapes may use neurons in its output layer, each indicating a different shape. To make the ANN also discern the color of an object many more output neurons would be needed: one per combination of shape and color. However, it seems unlikely that human brains are organized to have one neuron per combination. Instead, neuroscientists argue that information in the brain is represented by the simultaneous activity of numerous (thousands of) neurons and the same set of neurons could represent entirely different concepts [4, 5].

The 90s saw the emergence of *cognitive models* [4, 6] that depend on very high dimensionality and randomness. A framework called *Vector Symbolic Architectures* (VSAs) [7] is essentially a family of theories for cognitive representation that provide operations for building data structures based on high-dimensional vectors [8] and, in combination with neural networks, trace a path toward symbolic computation systems that are able to learn from data. In *hyperdimensional computing*[1] (HDC) [4, 9] each piece of information is represented as a *hyperdimensional vector*, or *hypervector*, i.e., a large array of numbers, representing a point in high-dimensional space. These mathematical

*Corresponding author. E-mail: claudia.damato@uniba.it.
[1] www.hd-computing.com

objects and the algebra to manipulate them are flexible and powerful enough to address the mentioned limitations fostering a new approach to Artificial Intelligence (AI): a new approach in which efficient and robust computing can make decisions that are more transparent. HDC/VSA originated from proposals of integrating the advantages of the symbolic approach to AI, such as *compositionality* and *systematicity*, and those of the ANNs to AI (*connectionism*), such as vector-based representations, learning, and grounding (see [10, 11] for a recent comprehensive survey).

Vectors are used to represent both features (variables) and also the values that can be assigned to the variables. The vectors must be distinct. This distinctness can be quantified by their *orthogonality*: in a hyperdimensional space, there is a huge number of such mutually orthogonal vectors. But if consider also vectors that are nearly orthogonal, the number of such distinct hypervectors explodes: millions of nearly orthogonal hypervectors in a 10^4-dimensional space. Because there are so many possible nearly orthogonal vectors in such a space, one can just represent the mentioned items with distinct random vectors which are almost guaranteed to be nearly orthogonal.

Given hypervectors for features and values, systems have been conceived to manipulate them using basic operations which correspond to ways of symbolically manipulating entities/concepts and allow to build a structured representation. Formal algebras of hypervectors have been devised based on such operations thus allowing for symbolic reasoning on the resulting representation models [9, 10].

An advantage of the representation is the *tolerance to errors*: even if a hypervector suffers significant numbers of random bit flips, it is still close to the original vector. Thus reasoning using these vectors is not meaningfully impacted in the face of errors. Another advantage of the approach is *transparency*: the algebra clearly tells why the system chose a given answer. This opens the way to *analogical reasoning*, which is expected of any AI system.

HDC seems also well suited for a new generation of low-power hardware and it is compatible with *in-memory computing*, that is performed on the same hardware that stores data unlike standard architectures that inefficiently transfer data back and forth between memory and CPU. Some of these new devices can be analog, operating at very low voltages, making them energy-efficient but also prone to random noise. As such this solution is gaining interest in the context of *Internet of Things* and *edge computing* [3].

Several applications of HDC/VSA have been proposed in the last decade (see [11] for a comprehensive survey). Specific algorithms for such representations have been developed to replicate typical tasks for (deep) neural models such as image classification. The strength of hyperdimensional computing lies in the ability to compose and decompose hypervectors for reasoning. This strength has been recently demonstrated [12] on the solution of a classic *abstract visual reasoning* problem, known as *Raven's progressive matrices*, that results particularly challenging both for ANNs and even for humans (it has been used in IQ tests).

Complex structures, such as those in images, scenes, episodes, can be represented as single hypervectors that contain information about all the objects involved, including their properties. Basically, an algorithm analyzes the features of each instance using some predetermined scheme and creates a hypervector for each image. Then, for each class it can aggregate via simple operators (e.g., addition) the hypervectors for all instances of the class to create a hypervector which can be stored in memory. Given a new unlabeled case, a hypervector is created for it and then compared against the stored class hypervectors to determine by analogy the most similar class. Our idea is to adopt this representation for AI methods dealing with more abstract objects such as individuals and their relations within *Knowledge Graphs* (KGs) [13], as well as related schema-level knowledge, such as classes and properties.

In the rest of the paper, the basics of neural representations and their properties are recalled in §2 while the distributed representation used for vector-space architectures is illustrated in §3. A simple HD model is presented in §4 to show how complex structures can be built from atomic symbols (hypervectors). Then, in §5 and §6, we examine the application of analogical reasoning and learning approaches, respectively, outlining possible extensions, with a particular focus on KGs. Finally, §7 concludes the paper and delineates further potential research.

2. Neural Representations and their Properties

Symbolic representations [14] are natural for humans and widely used in AI: each object is represented by a *symbol* (by objects we refer to items of various nature and complexity, such as physical objects, features, relations, classes, and so on). More complex representations can be composed from simpler ones and they naturally possess a combinatorial structure that allows producing indefinitely many symbolic expressions by means of rules / programs

(natural language can be considered as example of such representations). Symbols are characterized by an explicit 0/1 similarity: the same symbols have maximal similarity, whereas different symbols have null similarity and are, therefore, called dissimilar. Comparing composite symbolic structures requires to follow edges and/or match vertices of underlying graphs to reveal the whole structures and calculate the similarities. Hence, scalability problems arise in similarity search and reasoning with models when they require complex sequential operations.

Two main types of *connectionist representations* are distinguished: localized and distributed [15]. *Localized representations* are similar to symbolic representations as for each object there exists a single corresponding element in the representation (e.g., a single neuron / node or a single vector component). Connections between elements can be created to link localized representations, corresponding to *pointers* in symbolic representations. However, constructing structures that include combinations of object representatives may require the allocation of a potentially infinite number of new additional elements / connections, which is neuro-biologically questionable. Also, similarly to symbolic representations, localized representations lack a sufficient semantic basis, namely an immediate and explicit similarity between the representations. Different neurons representing different objects are dissimilar and estimating object similarity requires additional computation.

Distributed representations [8, 16] were inspired by the concept of *holographic* representation which is an alternative connectionist model of the brain in which information is distributed across many neurons. In distributed representations, the state of a finite set of neurons is modeled as a vector where each vector component represents a state of the particular neuron. They are defined as vector representations, where each object is represented by a subset of vector components, and each vector component can be part of the representations of many objects. This regards representations of objects of various complexity, from elementary features or atomic objects to complex scenes/objects that are represented by (hierarchical) compositional structures. The state of individual components of the representation cannot be interpreted without knowing the states of other components. Hence the semantics of individual components usually cannot be defined in isolation. However, for engineering applications and cognitive modeling, distributed representations of similar objects should be similar according to some similarity measure over the vector space. As pointed out in [10], distributed representations should endowed with some desirable properties:

- High *capacity*: i.e., they must allow the representation of a large number of entities;
- *Explicit* representation of *similarity*: similar objects must have similar representations that can be efficiently compared via vector similarity measures;
- A rich *semantic basis*, due to the direct use of features in the representations of objects and the possibility of defining their similarity based on their vectors;
- The ability to *recover* the original representations of objects;
- The ability to cope with noise, malfunction, and uncertainty, thus enforcing neuro-biological plausibility;
- A *direct access* to the representation of an object: in a vector, a representation of a compositional structure can be processed directly, because it does require tracing pointers, as in symbolic representations, or following connections between elements, as in localist representations;
- A *unified format*: every object, be it atomic or composite, is represented by a vector and, hence, the implementation operates on vectors without being explicitly aware of the complexity the related objects;
- The possibility of using well-consolidated mathematical methods for vector processing.

Conventional implementations of *brain-like* symbolic computations also require reliable hardware [17], since any error might result in a fatal fault [10]. Neural representations are subject to various *context effects*, as evident from the kinds of errors we make, as they are not optimized for fast and reliable arithmetic operations. The context effects likely reflect a compromise in favor of vital functions of the brains [4, 9].

Computers adopt a binary representation: an individual circuit component has two possible states, as electronic components are most reliable when they are *bistable*. The representation must discriminate. The binary patterns for different things must be different. However the choice of the patterns is often a compromise aimed at performing efficient operations on them (e.g., the positional representation for numbers). Neural representations are carried on components that are nonbinary. Yet many context effects can be demonstrated with binary patterns and operations. The important properties of the representation descend from *high dimensionality* rather than from the nature of the single dimensions. In contrast to *dimensionality reduction*, a standard practice in the processing of high-dimensional

data, a very high dimensionality can actually facilitate processing: "*instead of being a curse, high dimensionality can be a blessing*" [4]:

Hyperdimensionality Brains contain massive numbers of neurons and synapses. Computing with very long words leads to high-dimensional spaces and vectors; the term *hyperdimensional* will be used to indicate a dimensionality in the thousands (hence *hyperspace/hypervector* are shorthands for hyperdimensional space/vector). Hyperspaces have specific properties that have led to the successes of deep learning. Yet, much more can be accomplished by further exploiting the properties of hyperspaces.

Randomness No two brains are identical: they are highly structured but many details are determined by learning or are left to chance. They are *incompatible* at the hardware level: internal patterns cannot be transferred from one brain to another. Each system builds its model of the world from *random patterns*, i.e., vectors drawn randomly from the hyperspace. Compatibility is rather to be sought in the reciprocal *relations* among patterns within each system (e.g., in the case of languages, the same meaning can be expressed adopting a different syntax and lexicon). Thus, for example, at the internal code level, the patterns for `bike` and `car` should be more similar than those for `boat` and `car` in the same neural system, while the patterns for `bike` in different systems need not exhibit any similarity. Randomness can be thought as the *path of least assumptions* that make systems easy to design.

Holism Even simple mental events involve the simultaneous activity of widely dispersed neurons. Discovering how the activity is exactly organized is extremely hard. In a *holistic* representation, for maximum robustness the information encoded into a representation should be distributed equally over the entire hypervector. The vectors are not subdivided into nonoverlapping fields. Anything that is encoded into a vector is distributed equally over all its components. Any part of a vector represents the same thing as the entire vector, only less reliably. When bits fail, the information degrades in relation to the number of failing bits irrespective of their position (differently from binary numbers where the position of the bits determines its value). Position-independence is applicable to representations at the abstract levels of cognition where information from different senses has been integrated and where some more general mechanisms come into play.

Robustness The neural architecture is tolerant of component failure given a redundant representation in which many patterns are considered equivalent as they mean the same thing. A simple way to achieve redundancy is replication, but it is not energy-efficient, and it deviates from a potential correspondence to models of animal brains. Error-correcting codes used in telecommunications can tolerate a number of errors. The proportion of *allowable errors* increases with dimensionality. With a hyperdimensional representation the number of places where equivalent patterns can differ becomes very large.

3. Hyperdimensional Representation Models

Hyperdimensional representation spaces are primarily characterized by the domain of the vector components (e.g., binary, real, or complex numbers) and by their dimensionality, which determines the set of *atomic hypervectors*. They are also characterized by their sparseness, given a probability distribution which defines a space of hypervectors with independent components drawn from this distribution. Actually, different representation spaces can coexist in a cognitive system. For example, considering a typical representation space of 10^4-bit patterns contains a total of 2^{10^4} hypervectors (points). The space essentially comprises all the corners of a 10^4-dimensional unit (hyper)cube, yet an infinitesimal fraction of them would be required to represent meaningful entities.

Basic Operations. It was shown [4, 9] that it is possible to encode and decode all data structures typical of ordinary computing, such as sets, sequences, lists, and further extensions, in terms of basic operations on holographic vectors, namely, **superposition** and **binding**. The former is used to form an hypervector that represents several others (in analogy to simultaneous activation of neural patterns). The latter is used to associate two (or more) hypervectors that should be bound together (e.g., a variable and its value).

Considering the operations[2] on vectors of real numbers, which are commonly used in the ANNs research, these basic operations[3] can be defined in terms of:

addition component-wise operation, resulting in a vector of the same dimensionality; the *sum-vector* is usually normalized, yielding a *mean vector*, via some form of weighting or other transformations[4] of the vector components [4]. The sum (and the mean) of random vectors is similar to each of the input vectors, making it a possible representation for their set.

multiplication component-wise operation that produces a new vector, with the desirable properties of *invertibility*, to avoid any loss of information, *distributiveness* over addition, *distance preservation* and, *dissimilarity* w.r.t. the vectors being multiplied. These properties make it possible to encode compositional structures into hypervectors and to analyze the contents of composed hypervectors;

permutation that amounts to rearranging the vector components and it can be represented as a function or as the result of multiplying a vector with a *permutation matrix*, i.e., one filled with 0s except for exactly one 1 for each row and column.

Further related operations are: *weighting* with a constant, aX, where each component of the vector X is multiplied with the same number a, and the result is a vector; *subtraction* (*difference*) which is accomplished by adding to the first the second vector's *complement*, that is obtained by multiplying each component by -1 (or just flipping the bits in case of binary vectors); two vectors can be multiplied to give their so-called *inner product*, that can be used as a measure of similarity (e.g., the cosine); their *outer product*, which yields matrices, is a form of multiplication that is employed extensively for fitting the weights of ANNs. Multiplying a vector by a matrix, which is another common operation with ANNs, results in a vector that may require normalization.

The choices of the representation for the atomic symbols and of the implementation of the basic operations determine also the definition of a measure of **similarity** on the hyperdimensional space for the *comparison* of hypervectors. Examples of such measures are: *dot-product*, *cosine*, *Pearson's correlation*, and the *Hamming distance* [9, 10].

For example, considering a space of binary hypervectors, their similarity can be measured by the *Hamming distance*, which amounts to counting the number of bits that differ. Then the maximum distance is 10^4 and this measure can be expressed as relative to the number of dimensions. The (binomial) distribution of the distances makes them as highly concentrated halfway around 5000-bits, the mean, with a standard deviation of 50. Thus, a sphere of limited radius (say, 550-bits) centered at the mean distance will contain most of the space. If we consider two arbitrary random hypervectors, they are likely to differ by about 5000 bits and the same would hold also for each in a sequence of randomly chosen hypervectors for the representation. This makes them *unrelated* and the representation *robust*, since a very large number of bits in a noisy vector must change (more than 1/3) to make it unrelated to the original. From the opposite perspective, we can consider the relation of *similarity* between vectors, which remains strong until their distance approaches 0.5 (i.e., 5000-bits): the neighborhood volume within 3333-bits is quite limited with respect to the total space while unrelated vectors abound after a distance approaching 0.5. The neighborhoods of any two unrelated vectors have points in common: points that turn out to be very similar to any two unrelated points. Vectors can be related *by transformation*, i.e., by the way one is transformed into another or by the combination of several vectors in a new one.

4. A Structured Knowledge Representation Model

Given a content-addressable memory for hypervectors, and hyperdimensional arithmetic, a representation model for entities of various kinds can be devised. Models are based on the representation of atomic entities and on hypervector transformations defined in terms of the basic operations introduced above.

For illustrative purposes we will refer to simple models based on binary hypervectors with related operators and similarity measures. An extensive detailed presentation of the HDC/VSA models can be found in [10] (§2.3).

---

[2] We will adopt the notation used in [9]: lowercase letters will be used for scalars, variables, relations, and functions (e.g., a, x, f), uppercase letters for vectors (e.g., A, B), and Greek letters for (permutation) functions (e.g., ρ) or matrices (e.g., Π).

[3] By default in the order of the operators permutation precedes multiplication, and finally addition.

[4] e.g., by applying a threshold to get binary vectors,

Item Memory. An ideal architecture with a very large number of addresses would require storage for a huge number of locations. A pattern (a hypervector) X may be stored in a location addressed with a pattern A (or any A' similar to A). This solution is called *heteroassociative*, as opposed to the *autoassociative* memory where each X is stored in such a way that X itself is used as its address (*content addressable memory*), which makes retrieval much easier. A pattern becomes meaningful when it is selected to represent an entity. For later reference, they are stored in an autoassociative memory that forms a catalog of meaningful patterns that can be recognized even in the presence of noise. When searching with a noisy pattern, the corresponding noise-free stored pattern is retrieved (hence the name *clean-up memory*) in a kind of nearest neighbor search among the stored patterns. Arithmetic operations on patterns produce approximate (noisy) results that require cleaning up to recover the original pattern. Some operations produce meaningful patterns that are very similar to each other making them difficult to retrieve. For example, the sum pattern $S = A + B$ is similar to both A and B. Therefore, it is advantageous to transform S prior to its storage, mapping it into a different region of space, provided that the transformation is invertible.

Symbols for Basic Entities. A formal system can be built up from a set of basic atomic entities. The smallest meaningful unit of the cognitive code is a hypervector. The atomic entities, or *individuals*, are thus represented by random points of the hyperspace. To represent something new that does not consist of entities already represented in the system, one can simply randomly draw a new vector that must be stored in item memory for later reference. The new vector will be unrelated to all the vectors in memory (its distance from all of them is very close to 0.5), i.e., it will be approximately orthogonal to each of them. For example, a 10^4-dimensional space has 10^4 orthogonal vectors and a huge number of nearly orthogonal vectors [4].

Sets and Superposition. Both the set and its elements can be represented using hypervectors. The simplest commutative operation (such that the order does not matter) is vector addition. A sum-vector (or a the mean-vector) has the property of being similar to the added vectors. Thus, the elements are said to be *visible* in the representation of the set, and sets that share elements are represented by similar vectors. To distinguish the sum-vector representing the set from the vectors of its elements, it must to be mapped into a different region prior to its storage in memory. The mapping should be invertible so that the original sum-vector can be retrieved, and it should preserve distances so that the memory can be searched even with partial or noisy sums. Elements are recovered from a stored sum-vector by first restoring the sum (with the inverse mapping) and then searching the memory for the best match with it. The other elements can be retrieved from the *difference-vector* resulting from subtracting the former [4]. Large sets can be better analyzed by accumulating (partial) sums from the vectors recovered and by subtracting them from the original sum to get the various elements. However, if the unmapped sum has been stored in the item memory, this method fails. It is also possible to find previously stored sets (i.e., sums) that contain a specific element by searching the memory with that element (with its vector). Yet, preliminarily, the element must be mapped into the same part of space, i.e., via the same mapping, used before storing sum-vectors. Also a *multiset* (a *bag*) can be represented as the sum of their elements, which can be extracted from the sum in the same way.

Multiplications and Related Mappings. Without loss of generality, two simple forms of mapping will be recalled for hyperspaces of $(0,1)$-binary vectors (or also in $(1,-1)$-binary *bipolar* form) [4]. The first is the *multiplication of vectors* that will be denoted with $A*B$. In a bipolar binary system ordinary multiplication can be used. In the case of binary vectors can be defined as the *component-wise eXclusive-OR*, XOR (e.g., `1010...01` XOR `1001...11` = `0011...10`). This can also be thought of as a special case of Euclidean distance or a sum modulo 2, often indicated with $\oplus$. XOR is commutative and each A is its own product-inverse since $A*A = O$, where O is the vector with all 0s and it is the *unit* element, as $A*O = A$. It is easy to see that the Hamming distance between two vectors is simply the number of 1s in their product, denoted $d(A,B) = |A*B|$, where $|\cdot|$ represents the count. This distance is usually expressed relative to the number of dimensions. Multiplication can be thought of as a mapping: multiplying X by A maps it to the vector $X_A = X*A$ which is as far away from X as there are 1s in A (i.e., $d(X_A,X) = |A|$). A typical random vector A has about half of its bits set, so X_A is in the part of the space that is unrelated to X in terms of the distance criterion. This shows that multiplication randomizes. Moreover, it is also distance-preserving:

Example 4.1. *Considering the product-vectors $X_A = A*X$ and $Y_A = A*Y$, their product is $X_A*Y_A = (A*X)*(A*Y) = A*X*A*Y = X*Y$. Note that same product-vector implies same Hamming distance: $|X_A*Y_A| = |X*Y|$.*

Therefore, when a set of points is mapped via multiplication by the same vector, the relative distances are maintained (the cluster of points is just moved to a different part of space). High-level cognitive functions (e.g., analogy) can be defined in terms of these mappings where the relations between entities are more important than the entities themselves. In the previous example, A can be regarded as a specific mapping applied to vectors X and Y. The same argument applies if we consider two mappings A and B and examine their effect on the same vector X: X would be mapped onto X_A and X_B that are as far apart as A and B. Therefore, similar vectors lead to similar mappings. Similar mappings map a vector to similar vectors.

Permutation is another form of multiplication that reorders the vector components. The permutation of a vector A can be denoted either as the application of a function, $\rho(A)$, or as a multiplication by the corresponding *permutation matrix*, ΠA. Alternatively, it can be described as a list of integers (positions) in the permuted order. As a mapping, it has desirable properties: it is invertible, it distributes over addition (and over XOR multiplications) and the result is dissimilar to the vector being permuted. Moreover, distances are preserved:

Example 4.2. *Considering $\rho(A) * \rho(B) = \rho(A * B)$, one can observe that $d(\rho(A),\rho(B)) = |\rho(A) * \rho(B)| = |\rho(A * B)| = |A * B| = d(A, B)$.*

A *random permutation* is one where the order is chosen randomly from the $n!$ possible permutations, where n is the dimensionality of the hypervectors. Given random permutations ρ_1 and ρ_2, it can be shown [4] that they agree on the average in only one position, thus the distance between $\rho_1(A)$ and $\rho_2(A)$ is about 0.5, so they map A to different parts of the space.

Associations: Pairs and Bindings. A *pair* is a basic unit of association in which two elements A and B correspond to each other. They can be represented with a multiplication: $C = A * B$. Knowing C and one of its elements, one can find the other by multiplying C with the inverse of the known element. With XOR, any pair of two identical vectors will be represented by the 0-vector O. This can be avoided adopting a slightly different multiplication that neither commutes nor is a self-inverse. One can encode the order of the operands by permuting one of them before combining them. By permuting the first, we get: $C = A * B = \rho(A)$ XOR B. This kind of multiplication has all the desirable properties mentioned above. However it is not associative because of the permutation, i.e.: $\forall A, B, C : (A * B) * C \neq A * (B * C)$. In a holistic representation, a *binding* can be represented as a multiplication of the variable vector X by the value vector A: $X * A$. The *unbinding* requires a simple multiplication by one of the two elements to get the other.

Records as Sets of Bound Pairs. A complex individual can be represented as a record consisting of fields that are variables, *roles* in Description Logics (DLs) [18], each of which stands for a relationship to a value or to another individual. In a holistic representation, roles are explicitly represented as vectors. Vectors for unrelated roles, such as `name` and `address`, can be chosen at random. A role bound to its filler is represented by the product $X * A$ defined as shown above. A single entity can be defined with a record that combines multiple role–filler pairs:

Example 4.3. *Consider the entity $E = W * A + X * B + Y * C + Z * D$ where: $W =$ `name`, $A =$ `"Mickey Mouse"`, $X =$ `creator`, $B =$ `Walt_Disney`, $Y =$ `creation_year`, $C =$ `1928`, $Z =$ `starring_in`, and $D =$ `Fantasia`.*

A record-vector is self-contained as or it includes role-filler pairs explicitly. Moreover, it will be similar to all of its pairs, while it will be dissimilar to the single role and filler vectors because of the products. Unbinding can be used to extract the contained information:

Example 4.4. *In the example above, given the record-vector E, to get the filler of a role X (`creator`) it suffices to multiply E with (the inverse of) X and search the memory with the result, yielding B (`Walt_Disney`). Formally: $X * E = X * (W * A + X * B + Y * C + Z * D) = X * W * A + X * X * B + X * Y * C + X * Z * D = R_1 + B + R_2 + R_3$ Retrieving $X * E$ means retrieving the sum $R_1 + B + R_2 + R_3$, hence only B can be retrieved from the memory, since nothing similar to the three products is stored in memory.*

Adding more pairs (products) to a record-vector E, gives us a new sum-vector E' that is very similar to E because it shares four pairs with it. One might be interested in querying the whole database of records for some property or in processing some of its records. For example, knowing a filler of one of the roles in one record (the role is not known) can enable one to get the filler of that role in another record. Together with standard database operations, different type of computing become possible: they are known as *holistic mappings* or *transformation without decompositions* [11] that can be used to resolve proportional analogies (see §5).

Graphs. A graph $\mathcal{G}$ consists of a set of *nodes* connected by *arcs* (or *edges*). Arcs can either be undirected or directed. In a simple transformation of graphs into hypervectors [10, 19] a random vector is assigned to each node of the graph and an arc is represented as the binding of vectors of the nodes it connects, e.g., the edge between nodes A and B is $A * B$. The whole graph is represented simply as the superposition of hypervectors representing all edges in the graph.

Example 4.5. *A pentagram, i.e., a star-shaped (undirected) graph $\mathcal{G}$ with 5 nodes $\{A, B, C, D, E\}$ and 2 edges per node can be represented as the following single hypervector: $G = A * C + A * D + B * D + B * E + C * E$.*

To represent directed graphs, the directions of the edges should be included into their vectors. This can be done by applying a permutation, so that the directed edge e.g., one from A to C, is represented as $A * \rho(C)$.

Example 4.6. *A directed graph $\mathcal{G}'$ can be derived from $\mathcal{G}$ in the previous example by specifying a direction to each edge as follows: $G' = A * \rho(C) + A * \rho(D) + B * D + \rho(B) * \rho(E) + E * \rho(C)$.*

This example and more complex graphs that represent scenes or episodes can be found in [10], §3.5.2.

5. Reasoning by Analogy

Reasoning by analogy depends on specific human capacities: the facility for conceptual abstraction and the flexible recombination of atomic components to describe new unseen situations.

Proportional Analogy Problems. Prior work has targeted so-called *proportional analogy* problems and leveraged large networks with relatively little structure to map an inferred relation between two entities onto a third one. These problems are described in the form: $A : B :: C : ?$, to be read *A is to B as C is to* ?, where A and B represent two entities between which a relation is computed and C is an entity onto which this relation is applied to produce the solution (another unknown entity). Typical such problems are related to the domain of images, *visual analogy problems*, tackled in several works (e.g., see [20]).

Computing such a relation in hyperdimensional representations is extremely simple, namely it suffices to subtract B from A. This lacks a full appreciation of the nature of analogy, namely the substitution of atomic components that may be embedded in larger data structures. This computation is not well-captured by the mere subtraction. However, in a HDC/VSA framework, substitution of atomic components within/between complex data structures can be both elegant and very efficient if the data structures are constructed by means of *binding*. The operation of *unbinding* reveals analogical relations that can be used in various applications as discussed in [6, 21].

Holistic transformations for solving these problems are often illustrated using the well-known case of the "Dollar of Mexico" [21] in which a simple proportional analogy is considered: `US : México :: Dollar : ?`. This kind of analogy is known to be solvable by addition and subtraction of the embeddings of the corresponding concepts [22, 23]. Similarly, in [24] these analogical retrieval problems are tackled via shallow ANNs using a dependency path of relations between terms in sentences. In conventional symbol manipulation (using geometric properties of the representation space) there is no direct analog to this kind of processing by holistic transformation. The holistic transformation of hypervectors can be seen as a parallel alternative to the conventional sequential search.

Learning systematic transformations from examples in the *Holographic Reduced Representation* (HRR) model [8] was investigated in [25, 26]. A transformation hypervector is obtained from several training pairs of hypervectors as a solution to an optimization problem found by *Gradient Descent*, iterating through all examples until convergence. The learned transformation hypervector has been empirically proven to be able to generalize to new compositional structures with novel elements in structures of higher complexity than those provided as training examples. Similar capabilities with *Binary Spatter Code* (BSC) representations are illustrated in [27]. The drawback with such holistic transformations is their bidirectionality, which is due to the fact that in BSC *unbinding* is equivalent to *binding*. This complication can be solved by employing either the permutation or an additional associative memory similarly to the binding operation. A potential shortcoming of the approaches to learning holistic transformations is that the objects (and relations) are assumed to be dissimilar. Yet learning might not work as expected given that there is some similarity structure between the objects (relations) used as training examples. This is a direction that is worthy of further investigation.

Analogical Reasoning. In Cognitive Science, *analogical reasoning* theories [28] deal with analogical *episodes* (or *analogs*) and usually comprise (models for) a process[5] comprising the following four basic steps:

1. The *description* step concerns the representation of episodes, which can be modeled as systems of hierarchical relations consisting of elements in the form of entities and relations of different hierarchical levels. Entities are described by attributes and relations between the elements in episodes. Arguments of relations may be objects, attributes, and other relations. It is assumed that a collection of (source) *base episodes* is stored in a memory.
2. The *retrieval* step searches the memory for the closest episodes to the given *query* episode.
3. The *mapping* step, once the base episodes are identified, determines the correspondences between the elements of the query and the base episodes.
4. The *inference* step transfers knowledge from the base analogical episode(s) to the target analogical episode. For example, the problem specified by the query episode may be solved by means of inferred knowledge about the target. Candidate inferences must be treated as hypotheses and must be evaluated and checked [31].

The processing of analogical episodes involves two types of similarity. *Structural similarity* reflects the relations between the various elements of the episodes. Episodes are also matched in terms of a *superficial similarity*, which is based on common elements in episodes, or in terms of a broader notion of *semantic similarity*, which may be based on the similarity of characteristic feature vectors or on common membership in a taxonomic category.

HDC/VSA models have been used for *analogical retrieval* (see [6] and references therein). In such models both the set of structure elements and their arrangements influence the similarity of the corresponding hypervectors: similar episodes produce similar hypervectors. In [11] various studies are described which prove that results obtained by similarity estimation on hypervectors are consistent with both the empirical results in psychological experiments and the leading traditional models of the analogical retrieval.

Regarding the task of *analogical mapping*, models of mapping have been proposed that use techniques based on holistic transformations (again, see [11]). One of the limitations of these models is scalability. The similarity of hypervectors of the analogical episodes can be considered for their mapping. However, so far this approach was proved to work only for straightforward mapping cases. Several alternative mapping techniques have been proposed (including direct similarity mapping, re-representation by substitution of identical hypervectors, and parallel traversing of structures using higher-level roles). Some of them have been demonstrated on complex analogies. However, the techniques are quite elaborate and use sequential operations.

Interestingly, the hyperdimensional vector models adopted for analogical reasoning are compatible with established formats of knowledge representation, such as KGs. This facilitates the unification of symbolic and subsymbolic approaches for cognitive modeling and AI. An interesting study in this direction has presented a proof of concept of the mapping between an RDF-Schema ontology and a HDC/VSA model [32].

Transformations of original data into hypervectors allow constructing hyperdimensional models for various application areas preserving the form of similarity that is relevant for a particular application [11]. This provides a tool to use HDC/VSA for *similarity-based reasoning* comprising (unsupervised) *similarity indexing/search* and (supervised) *classification* in its simplest form as well as much more advanced analogical reasoning techniques.

6. Learning

Classification is currently one of the most common application areas for of HDC/VSA, especially applied to images, signals and biomedical data in general (see [11, 33] for detailed surveys). Similarity-based classification for vectorial representations of the instances are widespread in machine learning. A taxonomy of classification methods with HD models can be based on the headings level they focus on: primarily (first-level headings) we have methods devoted to the transformation of *input data* into hypervectors. Higher-level headings suggest further directions for applying HDC to classification, focusing on the *types* of input data (second-level headings) and on the *domains* (third-level headings), respectively.

[5] In the context of Machine Learning this process is also known as *case-based reasoning* [29]. Actually *instance-based learning*, *similarity-based* predictions made by memorizing prototypes may be ascribed to this form of learning [30].

In a basic classification methodology [33], during the *training* phase, an *encoder* employs randomly generated hypervectors (stored in the *item memory*) to map the training data into the hyperdimensional space. Then *K class* hypervectors are learned and stored in the associative memory. In the inference phase, the encoder generates the *query* hypervector for each test data. Then, a similarity comparison is performed in associative memory involving the query and each class hypervector, to finally predict the most similar class.

Simple models based on centroids are known to fall short in terms of generalization [34]. Improvements can be obtained by weighting newly assigned instances to the centroids [35]. It has been observed that conventional classifiers assume that the input vector components can be interpreted independently. This is reasonable when components are features endowed with meaningful interpretations. However, it is not generally applicable to HD representations [11]. Sparse HDC/VSA representations may be adequate for classifiers that benefit from sparsity, such as *Winnow* [36] and sparse *Support Vector Machines* [37].

Devising suitable encodings for good classifiers is a challenging task [7] worth of specific investigations: an appropriate choice of the encoding method is crucial for an accurate classification. For example, specific efficient encodings for biosignal processing are presented in [2]. Alternatively, different encoding methods can be integrated together to achieve higher accuracy [38]. Compared to single-pass training, *retraining* iteratively has been demonstrated to improve the accuracy of the classification models [39]. This suggests the direction of investigating ensemble (hierarchical) models of encoders (e.g., see [40]).

Competitive learning models are neural methods in which a number of groups/units compete to gain the responsibility of representing the incoming instances. They implement an online clustering algorithm, such as online *k-Means* and *Leader Cluster*, via neural extensions like *Adaptive Resonance Theory* [41] or *Self-Organizing Maps* [42]. These can be thought as local models for defining the input density which could be interestingly transposed to an HD representation. Probability density estimation (and other tasks such as kernel smoothing) have been shown to be efficiently tackled resorting to *fractional power encoding* [5]. An efficient algorithm in the context of such vector spaces has been presented in [43].

Once the instances are localized, a higher layer of units can be considered in the network architecture to perform supervised learning, leading to the definition of neural models, such as the *Radial Basis Function networks* [44] or the *Mixture of Experts* [45] This layer implements a local model for the generation of the output (classification/regression). Centroids have been shown to be easily combined with generalized *Learning Vector Quantization* classifiers [46]. Besides, moving from the idea of *multiple-class hypervectors*, disjunctive or mixture hierarchical representations of the classes may be targeted to better represent them [33].

Resorting to a HD representation, instances and cluster prototypes may be embodied by hypervectors and the mentioned clustering and classification/regression models may be implemented on HDC/VSAs. In the spirit of explainable AI, methods for the induction of (probabilistic) rules based on local models (basis functions) [47] could also be implemented on HDC/VSAs.

Structured data are generally more difficult to handle with conventional learning models, since local representations of structured data might not be convenient to use with vector classifiers, especially when hierarchies are involved. HDC/VSA models may be well suited for structured data, since they allow representing various structures (including hierarchies). Related to this task, we will deal with the case of problems regarding embedding models and knowledge graphs in §6.2.

An example of application field for such methods is chemio-informatics where classification models have been proposed to predict the properties of chemical compounds demonstrating state-of-the-art performance (e.g., see [48]). As an example, *MoleHD* [49], based on a representation of 2D molecular structures, demonstrated comparable results, on average, with respect to baseline drug discovery methods at much lower computational costs. More generally, the problem of classifying graphs using HDC/VSA is another promising direction for further investigations. In [50] it was shown that representing a graph as a superposition of hypervectors corresponding to vertices and edges, the resulting approach could achieve comparable results to standard baseline approaches on four out of six graph classification datasets while requiring much shorter training time.

6.1. Rule Induction from Examples

Learning logic axioms, especially in the form of rules, from examples has a long tradition [30]. Deductive inference with these statements is possible by devising a mechanism to *apply* them to specific cases (assertions). Such mechanisms can be expressed as holistic mappings based on the hypervector arithmetic.

It is possible to express logic *atoms* as tuples. Without loss of generality, we will focus on *binary* relations $p(h,t)$ encoded as *triples* $\langle h,p,t\rangle$ (which is common in the representations for KGs): $P_{ht} = P_1 * H + P_2 * T$:

Example 6.1. *An atom expressing the relation* $\langle x, \texttt{brother\_of}, y\rangle$ *can be represented by* $B_{xy} = B_1 * X + B_2 * Y$. *An atom* $\langle y, \texttt{has\_child}, z\rangle$ *can be represented by* $C_{yz} = C_1 * Y + C_2 * Z$. *An atom* $\langle x, \texttt{uncle\_of}, z\rangle$ *can be represented by* $U_{xz} = U_1 * X + U_2 * Z$.

To represent implication in a rule, the *antecedent* is a conjunction of atoms with the AND operator represented with the addition, e.g., $P_{wx} + Q_{xy} + R_{yz}$. As the *consequent*, say G_{xz}, is implied by the antecedent, the expression mapping the antecedent to the consequent may be represented as a product-vector:

Example 6.2. *The mapping* $R_{xyz} = U_{xz} * (B_{xy} + C_{yz})$ *translates the rule: IF* $\langle x, \texttt{brother\_of}, y\rangle$ *AND* $\langle y, \texttt{has\_child}, z\rangle$ *THEN* $\langle x, \texttt{uncle\_of}, z\rangle$.

Substituting the names of specific persons for x, y, and z, a true statement about a specific uncle is obtained.

Example 6.3. *To apply the rule, we may encode* $\langle \texttt{Jim}, \texttt{brother\_of}, \texttt{Ann}\rangle$ *with* $B_{ja} = B_1 * J + B_2 * A$ *and* $\langle \texttt{Ann}, \texttt{has\_child}, \texttt{Ned}\rangle$ *with* $C_{an} = C_1 * A + C_2 * N$ *combine them into the antecedent* $B_{ja} + C_{an}$, *and map it with the rule* R_{xyz}: $R_{xyz} * (B_{ja} + C_{an}) = U_{xz} * (B_{xy} + C_{yz}) * (B_{ja} + C_{an})$. *The resulting vector,* U'_{jn} , *is more similar to* $U_{jn} = U_1 * J + U_2 * N$ *than to any other vector representing a relation involving these individuals, thus allowing the inference that* Jim *is uncle of* Ned. *Extending this mechanism, considering another example involving other variables / individuals, say u, v, and w, one can obtain another similar rule* $R_{uvw} = U_{uw} * (B_{uv} + C_{vw})$. *Now, combining the two rules by simple addition results in an improved rule based on two examples:* $R'' = R_{xyz} + R_{uvw}$. *The new rule is better than the previous: if applied to (i.e., multiplied by) the antecedent involving* Jim, Ann, *and* Ned, *as above, we get* U''_{jn} *which is closer to* U_{jn} *compared to* U'_{jn}.

This setting may be regarded as a form of learning by analogy via the random hypervectors arithmetic.

Forward and backward chaining reasoning procedures on cognitive representations have been discussed in [51]. HD models have been used to represent a knowledge base with (propositional) clauses, including also negation, allowing for deductive inference [52]. By extending the representation with the explicit representation of variables can lead to new methods the induction of First-Order Logic rule bases or more recent statistical relational models [53]. Devising appropriate transformations, also other types of kinds of knowledge bases may be targeted, such as ontologies and knowledge graphs expressed by axioms in DLs.

6.2. Embedding Models and Knowledge Graphs

The *distributional semantics hypothesis* [54], which suggests that linguistic items with similar distributions have similar meanings, is the basis for the idea of constructing *context vectors*. This was later extended to the semantics of more general entities such as concepts and relations. In principle, context vectors can be obtained in any domain where objects and contexts can be defined. Context vector methods associated with HDC/VSA usually transform the frequency distributions into specific *context hypervectors*.

Examples of successful *semantic indexing* methods based on context hypervectors are *Predication-based Semantic Indexing* [55] and its extension *Embedding of Semantic Predications* [56], as well as BEAGLE [57]. In [58], the authors focus on the representation of similarity: for each word, the most relevant semantic features that can be borrowed from *ConceptNet* are taken into account. The context hypervector for a word is defined as a superposition of its semantic feature hypervectors formed by role-filler bindings. Thus, moving from words to specific entities (resources) described in semantically rich knowledge bases (such as Web ontologies), similar HDC/VSA methods may be transposed from linguistics to more general domains for new downstream applications. The plethora of *2VEC models stemmed from the ideas presented in [22] are natural candidates for further suitable transpositions.

We have seen (§4) that graphs are general data structures that can be encoded in HD vector representations. *Graph Neural Networks* [59] are the typical models for graph representations that extend regular neural network operations to handle graphs. Specific learning models based on a HD representation have been proposed, one of the most recent being *GraphHD* [50], which achieves comparable effectiveness at lower computational cost compared to current deep neural models. Since methods based on graph kernels have been shown to deliver comparable results [60], it seems worthwhile to investigate such methods in combination with HDC.

KGs are specific types of multi-graphs that are intended to represent knowledge bases with a graph-structured data model including entities, attributes and properties and their semantics allowing for reasoning services involving related terminologies (schema-knowledge), that is expressed in DLs or related ontology languages [18]. Construction, refinement, and inference tasks for KGs have been tackled by exploiting the natural decomposition of KGs into triples, which has given rise to many *embedding models* mapping entities and properties to low-dimensional spaces, where such complex tasks can be approximated by geometric operations on embedding vectors and matrices.

Given suitable encodings from triples to hypervectors, as discussed in §6.1, it seems promising to target the development of further similar learning models based on HD representations to perform the mentioned tasks for KGs. Vector symbolic representations as encodings of semantic concepts (words, facts, properties of physical appearance, etc.) in high-dimensional vectors have been used to great effect in KGs (e.g., see [23]). Hypervectors of KGs can also be constructed from those of nodes and relations learned from data, as proposed in [61], where the HRR model was adopted by virtue of its differentiability. Representations for KGs have been further investigated in [62], where the use of a Cauchy distribution to generate atomic hypervectors, which are used as input to a neural network, has been shown to yield state-of-the-art results on the task of inferring missing links.

7. Further Perspectives for Future Work

Although HDC based on VSAs is no longer in its early stage it can still offer several directions for further improvement. These lines include (but are not limited to):

- Feature extraction and encoding methods: These are essential activities as hyperdimensional models cannot successfully handle complex data without appropriate encodings. Concerning the density of hypervectors, a choice between dense and sparse approaches should be made according to the application scenarios: a sparse representation requires less memory. *Nonlinearity* is another important aspect of transforming data into hypervectors. A lack of studies on the nonlinearity properties of hypervectors obtained from the compositional approach has been recognized (see [11], §4.1.2).
- *Similarity* assessment is also related to these objectives. New metrics should be developed to make a tradeoff between accuracy and complexity (also depending on the possible HW implementations). The choice of the kernel for kernel-based machines is also related to this issue. Kernels are related to the adopted HD transformation [5]. The transformation can be considered as another hyperparameter in an optimization problem which can be determined using standard statistical approaches such as *cross-validated* search strategies.
- *Learning strategies* such as *retraining* [33, 63] should be further explored to improve the accuracy of HD classification models. This goal can be pursued through *hybrid systems*, i.e., models that combine conventional learning methods with HDC models (see, e.g., [64]).
- Complex models such as deep neural networks used in combination with VSAs require specific eXplanable AI approaches to make such black boxes more transparent and interpretable, and decisions more comprehensible.
- Engagement with other *cognitive tasks*: These include, but are not limited to, other forms of *reasoning* (under uncertainty), *relational representation*, and *semantic generalization*. For example, *abduction* is a form of inference that deserves more attention since it can be exploited for various reasoning and learning tasks such as diagnosis, hypothesis generation, relevance detection, and explanation.
- The interplay with models for *uncertainty* and *causality*: Neural networks can be related to specific graphical models [53]; probabilistic causal models may also suggest further directions for studies on the transfer of forms of reasoning under uncertainty to HDC/VSAs (see, e.g., [65]).

Acknowledgment. The research has been partially supported by the project FAIR - Future AI Research (PE00000013), spoke 6 - Symbiotic AI (https://future-ai-research.it/), under the NRRP MUR program funded by the NextGenerationEU and by the project HypeKG - Hybrid Prediction and Explanation with Knowledge Graphs (H53D23003700006), under PRIN 2022 program funded by MUR.

References

[1] G. Montavon, W. Samek and K.-R. Müller, Methods for interpreting and understanding deep neural networks, *Digital Signal Processing* **73** (2018), 1–15. doi:https://doi.org/10.1016/j.dsp.2017.10.011. https://www.sciencedirect.com/science/article/pii/S1051200417302385.

[2] A. Rahimi, P. Kanerva, L. Benini and J.M. Rabaey, Efficient Biosignal Processing Using Hyperdimensional Computing: Network Templates for Combined Learning and Classification of ExG Signals, *Proceedings of the IEEE* **107**(1) (2019), 123–143. doi:10.1109/JPROC.2018.2871163.

[3] T. Yu, B. Wu, K. Chen, G. Zhang and W. Liu, Fully Learnable Hyperdimensional Computing Framework With Ultratiny Accelerator for Edge-Side Applications, *IEEE Transactions on Computers* **73**(02) (2024), 574–585. doi:10.1109/TC.2023.3337316.

[4] P. Kanerva, Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors, *Cognitive Computation* **1**(2) (2009), 139–159. doi:10.1007/s12559-009-9009-8.

[5] E.P. Frady, D. Kleyko, C.J. Kymn, B.A. Olshausen and F.T. Sommer, Computing on Functions Using Randomized Vector Representations (in Brief), in: *Proceedings of the 2022 Annual Neuro-Inspired Computational Elements Conference, NICE'22*, ACM, 2022, pp. 115–122–. ISBN 9781450395595. doi:10.1145/3517343.3522597.

[6] T.A. Plate, Analogy retrieval and processing with distributed vector representations, *Expert Systems* **17**(1) (2000), 29–40. doi:https://doi.org/10.1111/1468-0394.00125. https://onlinelibrary.wiley.com/doi/abs/10.1111/1468-0394.00125.

[7] R.W. Gayler, Vector Symbolic Architectures answer Jackendoff's challenges for cognitive neuroscience, in: *Proceedings of the ICCS/ASCS Joint International Conference on Cognitive Science (ICCS/ASCS 2003)*, P. Slezak, ed., 2003, pp. 133–138. http://arxiv.org/abs/cs/0412059.

[8] T. Plate, *Holographic Reduced Representations: Distributed representation of cognitive structure*, CSLI Lecture Notes Vol. 150, CSLI Publications, Stanford, 2003.

[9] P. Kanerva, Computing with High-Dimensional Vectors, *IEEE Design & Test* **36**(3) (2019), 7–14. doi:10.1109/MDAT.2018.2890221.

[10] D. Kleyko, D.A. Rachkovskij, E. Osipov and A. Rahimi, A Survey on Hyperdimensional Computing Aka Vector Symbolic Architectures, Part I: Models and Data Transformations, *ACM Comput. Surv.* **55**(6) (2022). doi:10.1145/3538531.

[11] D. Kleyko, D. Rachkovskij, E. Osipov and A. Rahimi, A Survey on Hyperdimensional Computing Aka Vector Symbolic Architectures, Part II: Applications, Cognitive Models, and Challenges, *ACM Comput. Surv.* **55**(9) (2023). doi:10.1145/3558000.

[12] M. Hersche, M. Zeqiri, L. Benini, A. Sebastian and A. Rahimi, A neuro-vector-symbolic architecture for solving Raven's progressive matrices, *Nature Machine Intelligence* **5**(4) (2023), 363–375. doi:10.1038/s42256-023-00630-8.

[13] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. De Melo, C. Gutierrez, S. Kirrane, J.E. Labra Gayo, R. Navigli, S. Neumaier, A.-C. Ngonga Ngomo, A. Polleres, S.M. Rashid, A. Rula, L. Schmelzeisen, J. Sequeda, S. Staab and A. Zimmermann, Knowledge Graphs, *ACM Comput. Surv.* **54**(4) (2021). doi:10.1145/3447772.

[14] A. Newell and H.A. Simon, Computer Science as Empirical Inquiry: Symbols and Search, *Commun. ACM* **19**(3) (1976), 113–126–. doi:10.1145/360018.360022.

[15] S.J. Thorpe, Localized versus distributed representations, in: *Handbook of Brain Theory and Neural Networks*, MIT Press, 2003, pp. 643–646–.

[16] G.E. Hinton, J.L. McClelland and D.E. Rumelhart, *Distributed Representations*, in: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*, MIT Press, 1986, pp. 77–109–. ISBN 026268053X.

[17] N.J. Wang, J. Quek, T.M. Rafacz and S.J. Patel, Characterizing the effects of transient faults on a high-performance processor pipeline, in: *International Conference on Dependable Systems and Networks, 2004*, 2004, pp. 61–70. doi:10.1109/DSN.2004.1311877.

[18] F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi and P.F. Patel-Schneider (eds), *The Description Logic Handbook: Theory, Implementation and Applications*, 2nd edn, Cambridge University Press, 2007. doi:10.1017/CBO9780511711787.

[19] R.W. Gayler and S.D. Levy, A distributed basis for analogical mapping, in: *Proceedings of the 2nd International Analogy Conference*, 2009, pp. 165–174. https://redwood.berkeley.edu/wp-content/uploads/2021/08/Gayler2009.pdf.

[20] S.E. Reed, Y. Zhang, Y. Zhang and H. Lee, Deep Visual Analogy-Making, in: *Advances in Neural Information Processing Systems*, Vol. 28, C. Cortes et al., eds, Curran Associates, Inc., 2015. https://proceedings.neurips.cc/paper_files/paper/2015/file/e07413354875be01a996dc560274708e-Paper.pdf.

[21] P. Kanerva, What We Mean When We Say "What's the Dollar of Mexico?": Prototypes and Mapping in Concept Space, in: *Papers from the 2010 AAAI Fall Symposium*, AAAI Press, 2010. https://aaai.org/papers/02243-what-we-mean-when-we-say-whats-the-dollar-of-mexico-prototypes-and-mapping-in-concept-space/.

[22] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado and J. Dean, Distributed Representations of Words and Phrases and their Compositionality, in: *Advances in Neural Information Processing Systems*, Vol. 26, C.J. Burges et al., eds, Curran Associates, Inc., 2013. https://proceedings.neurips.cc/paper_files/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

[23] J. Pennington, R. Socher and C. Manning, GloVe: Global Vectors for Word Representation, in: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, 2014, pp. 1532–1543. doi:10.3115/v1/D14-1162. https://aclanthology.org/D14-1162.

[24] A. Paullada, B. Percha and T. Cohen, Improving Biomedical Analogical Retrieval with Embedding of Structural Dependencies, in: *Proceedings of the 19th SIGBioMed Workshop on Biomedical Language Processing*, ACL, 2020, pp. 38–48. doi:10.18653/v1/2020.bionlp-1.4. https://aclanthology.org/2020.bionlp-1.4.

[25] J. Neumann, Learning holistic transformation of HRR from examples, in: *Proceedings KES'2000, Fourth International Conference on Knowledge-Based Intelligent Engineering Systems*, Vol. 2, 2000, pp. 557–560. doi:10.1109/KES.2000.884110.

[26] J. Neumann, Learning the systematic transformation of holographic reduced representations, *Cognitive Systems Research* **3**(2) (2002), 227–235, Integration of Symbolic and Connectionist Systems. doi:https://doi.org/10.1016/S1389-0417(01)00059-6. https://www.sciencedirect.com/science/article/pii/S1389041701000596.

[27] P. Kanerva, Large Patterns Make Great Symbols: An Example of Learning from Example, in: *Hybrid Neural Systems*, S. Wermter and R. Sun, eds, Springer, 2000, pp. 194–203.

[28] D. Gentner and K.D. Forbus, Computational models of analogy, *WIREs Cognitive Science* **2**(3) (2011), 266–276. doi:https://doi.org/10.1002/wcs.105. https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wcs.105.

[29] B. López, *Case-Based Reasoning*, Synthesis Lectures on Artificial Intelligence and Machine Learning, Springer, 2013. ISBN 978-3-031-00434-6. doi:10.1007/978-3-031-01562-5.

[30] T. Mitchell, *Machine Learning*, McGraw Hill, 1997.

[31] D. Gentner and J. Colhoun, Analogical processes in human thinking and learning, in: *Towards a Theory of Thinking: Building Blocks for a Conceptual Framework*, B. Glatzeder et al., eds, Springer, 2010, pp. 35–48.

[32] C. Mercier, H. Chateau-Laurent, F. Alexandre and T. Viéville, Ontology as neuronal-space manifold: Towards symbolic and numerical artificial embedding, in: *proceedings of KRHCAI 2021 Workshop on Knowledge Representation for Hybrid & Compositional AI @ KR2021*, 2021. https://inria.hal.science/hal-03360307.

[33] L. Ge and K.K. Parhi, Classification Using Hyperdimensional Computing: A Review, *IEEE Circuits and Systems Magazine* **20**(2) (2020), 30–47. doi:10.1109/MCAS.2020.2988388.

[34] J. Karlgren and P. Kanerva, Semantics in high-dimensional space, *Front. Artif. Intell.* **4** (2021). doi:10.3389/frai.2021.698809.

[35] A. Hernandez-Cano, N. Matsumoto, E. Ping and M. Imani, OnlineHD: Robust, efficient, and single-pass online learning using hyperdimensional system, in: *Proceedings of the Design, Automation Test in Europe Conference Exhibition, DATE'21*, 2021, pp. 56–61–.

[36] N. Littlestone, Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm, *Mach. Learn.* **2** (1988), 285–318–.

[37] K. Eshghi, M. Kafai and H. Packard, Support Vector Machines with Sparse Binary High-Dimensional Feature Vectors, Technical Report, HPE-2016-30, Hewlett Packard Labs, 2016, https://www.labs.hpe.com/techreports/2016/HPE-2016-30.pdf. https://www.labs.hpe.com/techreports/2016/HPE-2016-30.pdf.

[38] M. Imani, C. Huang, D. Kong and T. Rosing, Hierarchical Hyperdimensional Computing for Energy Efficient Classification, in: *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, 2018, pp. 1–6. doi:10.1109/DAC.2018.8465708.

[39] M. Imani, D. Kong, A. Rahimi and T. Rosing, VoiceHD: Hyperdimensional Computing for Efficient Speech Recognition, in: *2017 IEEE International Conference on Rebooting Computing (ICRC)*, 2017, pp. 1–8. doi:10.1109/ICRC.2017.8123650.

[40] R. Wang, D. Ma and X. Jiao, EnHDC: Ensemble Learning for Brain-Inspired Hyperdimensional Computing, *IEEE Embed. Syst. Lett.* **15**(1) (2023), 37–40. doi:10.1109/LES.2022.3191641.

[41] G.A. Carpenter and S. Grossberg, The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network, *Computer* **21**(3) (1988), 77–88–. doi:10.1109/2.33.

[42] T. Kohonen, The self-organizing map, *Neurocomputing* **21**(1) (1998), 1–6. doi:https://doi.org/10.1016/S0925-2312(98)00030-7. https://www.sciencedirect.com/science/article/pii/S0925231298000307.

[43] A. Hernández-Cano, Y. Kim and M. Imani, A Framework for Efficient and Binary Clustering in High-Dimensional Space, in: *Proceedings of the Design, Automation Test in Europe Conference Exhibition, DATE'21*, 2021, pp. 1859–1864. doi:10.23919/DATE51398.2021.9474008.

[44] J. Moody and C.J. Darken, Fast Learning in Networks of Locally-Tuned Processing Units, *Neural Computation* **1**(2) (1989), 281–294. doi:10.1162/neco.1989.1.2.281.

[45] R.A. Jacobs, M.I. Jordan, S.J. Nowlan and G.E. Hinton, Adaptive Mixtures of Local Experts, *Neural Computation* **3**(1) (1991), 79–87. doi:10.1162/neco.1991.3.1.79.

[46] C. Diao, D. Kleyko, J.M. Rabaey and B.A. Olshausen, Generalized Learning Vector Quantization for Classification in Randomized Neural Networks and Hyperdimensional Computing, in: *proceedings of the International Joint Conference on Neural Networks (IJCNN 2021)*, 2021, pp. 1–9. doi:10.1109/IJCNN52387.2021.9533316.

[47] V. Tresp, J. Hollatz and S. Ahmad, Representing Probabilistic Rules with Networks of Gaussian Basis Functions, *Machine Learning* **27** (1997), 173–200. doi:10.1023/A:1007381408604.

[48] D. Jones, J.E. Allen, X. Zhang, B. Khaleghi, J. Kang, W. Xu, N. Moshiri and T. Simunic Rosing, HD-Bind: Encoding of Molecular Structure with Low Precision, Hyperdimensional Binary Representations, arXiv, 2023, arXiv: doi: 10.48550/arXiv.2303.15604 https://arxiv.org/abs/2303.15604. doi:10.48550/arXiv.2303.15604.

[49] D. Ma, R. Thapa and X. Jiao, MoleHD: Efficient Drug Discovery using Brain Inspired Hyperdimensional Computing, in: *IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2022*, D.A. Adjeroh et al., eds, IEEE, 2022, pp. 390–393. doi:10.1109/BIBM55620.2022.9995708.

[50] I. Nunes, M. Heddes, T. Givargis, A. Nicolau and A. Veidenbaum, GraphHD: Efficient Graph Classification Using Hyperdimensional Computing, in: *Proceedings of DATE '22, Conference on Design, Automation & Test in Europe*, C. Bolchini et al., eds, IEEE, 2022, pp. 1485–1490. ISBN 9783981926361. doi:10.23919/DATE54114.2022.9774533.

[51] V. Kvasnička and J. Pospíchal, Deductive rules in holographic reduced representation, *Neurocomputing* **69**(16) (2006), 2127–2139. doi:https://doi.org/10.1016/j.neucom.2005.09.011. https://www.sciencedirect.com/science/article/pii/S0925231205003474.

[52] D. Summers-Stay, Propositional Deductive Inference by Semantic Vectors, in: *Intelligent Systems and Applications*, Y. Bi et al., eds, Springer, 2020, pp. 810–820.
[53] L. De Raedt, K. Kersting, S. Natarajan and D. Poole, *Statistical Relational Artificial Intelligence: Logic, Probability, and Computation*, Morgan & Claypool, 2016. doi:10.2200/S00692ED1V01Y201601AIM032.
[54] Z.S. Harris, *Mathematical Structures of Language*, Wiley, 1968.
[55] D. Widdows and T. Cohen, Reasoning with vectors: A continuous model for fast robust inference, *Logic Journal of the IGPL* **23**(2) (2014), 141–173. doi:10.1093/jigpal/jzu028.
[56] T. Cohen and D. Widdows, Embedding of semantic predications, *Journal of Biomedical Informatics* **68** (2017), 150–166. doi:https://doi.org/10.1016/j.jbi.2017.03.003. https://www.sciencedirect.com/science/article/pii/S1532046417300527.
[57] G. Recchia, M. Sahlgren, P. Kanerva and M.N. Jones, Encoding Sequential Information in Semantic Space Models: Comparing Holographic Reduced Representation and Random Permutation, *Intell. Neuroscience* **2015** (2015). doi:10.1155/2015/986574.
[58] J.I. Quiroz-Mercado, R. Barrón-Fernández and M.A. Ramírez-Salinas, Semantic Similarity Estimation Using Vector Symbolic Architectures, *IEEE Access* **8** (2020), 109120–109132. doi:10.1109/ACCESS.2020.3001765.
[59] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner and G. Monfardini, The Graph Neural Network Model, *IEEE Transactions on Neural Networks* **20**(1) (2009), 61–80. doi:10.1109/TNN.2008.2005605.
[60] K. Xu, W. Hu, J. Leskovec and S. Jegelka, How Powerful are Graph Neural Networks?, in: *Proceedings of the 7th International Conference on Learning Representations, ICLR 2019*, 2019. https://openreview.net/forum?id=ryGs6iA5Km.
[61] M. Nickel, L. Rosasco and T. Poggio, Holographic Embeddings of Knowledge Graphs, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30, 2016. doi:10.1609/aaai.v30i1.10314. https://ojs.aaai.org/index.php/AAAI/article/view/10314.
[62] Y. Ma, M. Hildebrandt, V. Tresp and S. Baier, Holistic Representations for Memorization and Inference, in: *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018*, AUAI Press, 2018, pp. 403–413. http://auai.org/uai2018/proceedings/papers/163.pdf.
[63] M. Imani, J. Morris, S. Bosch, H. Shu, G. De Micheli and T. Rosing, AdaptHD: Adaptive Efficient Training for Brain-Inspired Hyperdimensional Computing, in: *2019 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, 2019, pp. 1–4. doi:10.1109/BIOCAS.2019.8918974.
[64] D. Kleyko, M. Kheffache, E.P. Frady, U. Wiklund and E. Osipov, Density Encoding Enables Resource-Efficient Randomly Connected Neural Networks, *IEEE Transactions on Neural Networks and Learning Systems* **32**(8) (2021), 3777–3783. doi:10.1109/TNNLS.2020.3015971.
[65] P.M. Furlong and C. Eliasmith, Fractional binding in vector symbolic architectures as quasi-probability statements, in: *Proceedings of the Annual Meeting of the Cognitive Science Society, CogSci'22*, 2022, pp. 259–266–.