The Blessing of Dimensionality

Perspectives of Reasoning and Learning with Hyper-Dimensional Vector-Symbolic Representations

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 that represent an alternative approach to neural computing with interesting specific properties. As hyperdimensional patterns are suitable to encode complex knowledge structures these new architectures open perspectives for reasoning and learning models with a closer correspondence to cognitive activities in human brains. We review a variety of approaches to these tasks reviewing recent methods for hyperdimensional representations delineating perspective lines for future investigations.

1. Introduction

The artificial neural networks (ANNs) that underpin successful deep-learning models are known to be too complex for their activity and decisions to be explained as they lack of *transparency* and they can be also extremely *power-hungry*. Their complexity makes it extremely hard to explain exactly how they produce their output and, pairwise difficult to map the mechanisms to used by human brains to *reason* and *learn* using own symbols for objects, abstractions, and the relationships between them.

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.

The 90s saw the emergence of *cognitive models* [26, 51] that depend on very high dimensionality and randomness. A framework called *Vector Symbolic Architectures* (VSAs) [9] is essentially a family of theories for cognitive representation that provide operations for building data structures based on high-dimensional vectors [50] and, in combination with neural networks, suggest a path toward systems for symbolic computation that are able to learn from data. In *hyperdimensional computing*[1] (HDC) [26, 28] 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 objects and the algebra to manipulate them are flexible and powerful enough to address the mentioned limitations fostering a new approach to 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 [31, 32] for a recent comprehensive survey).

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

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 hyper-dimensional 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 [28, 32].

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 *IoT* and *Edge Computing*.

Several applications of HDC/VSA have been proposed in the last decade (see [31] for a recent comprehensive survey). As complex structures, such as those in scenes, episodes, images, can be represented as single hypervectors that contain information about all the objects in the image, including their properties. Specific algorithms for such representations have been developed to replicate typical tasks for deep neural networks such as image classification, the recognition of handwritten digits, for example. An algorithm analyzes the features of each image using some predetermined scheme. It then creates a hypervector for each image. Next, the algorithm adds the hypervectors for all images representing a given digit to create a hypervector for the idea of that digit. Thus 10 class hypervectors are created. Now given a new unlabeled image, a hypervector is created for it and then compared the hypervector against the stored class hypervectors. This comparison determines the digit that the new image is most similar to.

The strength of hyperdimensional computing lies in the ability to compose and decompose hypervectors for reasoning. This strength has been recently demonstrated [17] 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). Further applications and lines for novel investigations can be found in second Part of the mentioned survey [31] and will be also delineated in this paper.

2. Neural Representations and their Properties

Symbolic representations [45] 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 the simpler ones ans 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 such models that often require complex sequential operations.

Two main types of *connectionist representations* are distinguished: localized and distributed [58]. *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 already represented objects may require the allocation of a potentially infinite number of new additional elements / connections, which is neuro-biologically questionable. Also, localized representations similarly to symbolic representations, lack of enough semantic basis, i.e. of immediate explicit similarity between the representations. Different neurons representing different objects are dissimilar and estimating object similarity requires additional computation.

Distributed representations [18, 50] were inspired by the idea of a *holographic* representation as an alternative connectionist representation based on modeling information in the brain as *distributed* over 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 in isolation usually cannot be defined. However, for engineering applications and cognitive modeling, distributed representations of similar objects should be similar according to some similarity measure of the corresponding vector representations. According to [32], distributed representations should have a number of desirable properties:

- High *capacity*: i.e. it 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 immediate use of features in the representations and the possibility of representing their similarity in their vector representations;
- The ability to *recover* the original representations of objects;
- The ability to cope with noise, malfunction, and uncertainty (enforcing neuro-biological plausibility);
- A *direct access* to the representation of an object: in a vector, a distributed representation of a compositional structure can be processed directly (not requiring 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 [60], since any error might result in a fatal fault [32]. 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 [26, 28].

Computers use a binary representation: which means that an individual circuit component has two possible states, denoted by 0 and 1: electronic components are most reliable when they are *bistable*. The representation must discriminate: the binary patterns for different things must differ. However the choice of the patterns is often a compromise aimed at having 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. However 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*" [26]:

Hyperdimensionality brains contain massive numbers of neurons and synapses. Computing with very long words very 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, whereas the patterns for `bike` in different systems need not exhibit any similarity to each other. Randomness can be thought as the *path of least assumptions* that make systems easy to design.

Holism In human brains, 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 applies to representations at 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: they mean the same thing. Replication is a simple way to achieve redundancy but it is not energy-efficient (thus moving away from a possible correspondence to models for animal brains). Error-correcting codes can tolerate a number of errors. With a hyperdimensional representation the number of places at which equivalent patterns may differ becomes quite large: the proportion of *allowable errors* increases with dimensionality.

3. Hyperdimensional Representation Models

Primarily hyperdimensional representational spaces are characterized by the domain of the vector components (e.g. binary, real, or complex numbers) and their dimensionality that defines the set of *atomic hypervectors*. They are also characterized by their sparseness and probability distribution which defines a space of hypervectors with (sparse) i.i.d. components drawn from the distribution of choice (actually, different representational spaces can coexist in a cognitive system). For example, considering a typical representational space of 10^4-bit patterns contains a total of $2^{10^4} \approx$ 2E+3010 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 that is possible to encode and decode all data structures familiar from 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*, by weighting or via other transformations[4] of the vector components. The sum (and the mean) of random vectors is similar to each of the input vectors, making it a possible representation for their set.

[2] We will adopt the notation used in [28]: lowercase letters will be used for 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,

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*, that is accomplished by adding the second vector's *complement* to the first 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); another form can yield a matrix called, or *outer product* (which used extensively for fitting the weights of an ANN). Multiplying a vector with a matrix results in a vector is also common with ANNs (usually the result needs to be normalized).

The choice of the representation of the atomic symbols and of the implementation of th 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* [28, 32].

Considering, for example, a space of binary hypervectors, their similarity can be measured using 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 sees them as highly concentrated half-way around 5000-bits, the mean, with a standard deviation of 50. Hence a ball with a limited radius (of, say, 550-bits) centred at the mean distance contains most of the space. If we consider two random hypervectors it is likely that they differ in about 5000 bits and the same would apply also for anyone in a sequence of randomly chosen hypervectors for the representation. This makes them *unrelated*. This makes the representation *robust* as a very large number of bits have to change (more than 1/3) in a noisy vector to make it unrelated to the original one. Taking the opposite perspective, we may 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 w.r.t. 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 representational 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 [32] (§2.3).

Item Memory. An ideal architecture with a very large number of addresses would also require storage for a huge number of locations. A pattern X may be stored in a location addressed by a pattern A (or any A similar to A'). This solution is called *heteroassociative* as opposed to the *autoassociative* mode in which each X is stored so that X itself is used as its address (*content addressable memory*) that makes storage recovery much easier.

A pattern (a hypervector) becomes meaningful when it is selected to represent an entity. For later reference they are recorded in a autoassociative memory that makes a catalog of meaningful patterns and can be recognized even when noisy. When searched using a noisy pattern, the corresponding noise-free stored pattern is retrieved (hence the name *clean-up memory*) in a sort of nearest-neighbor search among the set of 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 their retrieval complicated.

For example, the sum pattern $S = A + B$ is similar to both A and B. In such cases, before storing S, it is convenient to map it into a different part of the space, provided that the mapping can be reversed when needed.

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 anything new that is not composed of things already represented in the system, one can simply randomly draw a new vector that has to be stored in the 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 them. For example, a 10^4-dimensional space has 10^4 orthogonal vectors and a huge number of nearly orthogonal vectors.

Sets and Superposition. Both the set and its elements can be represented with hypervectors. The simplest commutative operation (so that the order does not matter) is vector addition. The sum-vector (and the mean-vector) has the property of being similar to the addend vectors. Thus, the elements are said to be *visible* in the representation of the set, and sets that share elements give rise to similar sums.

To produce a vector for the set that be dissimilar from those of its elements, it can be mapped into a different part of space before storing it in memory. The mapping should be invertible so that the original sum-vector can be retrieved, and 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. Further elements can be searched after computing the difference-vector.

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.

A *multiset* (or *bag*) can also be represented by the sum of the elements, and elements can be extracted from the sum also in the same way.

Mappings and Multiplication. Mappings can be defined in terms of multiplications. 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). The multiplication of vectors 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) producing a new vector with 0s where the two agree and 1s where they disagree (e.g. `1010...01` XOR `1001...11` = `0111...10`). It can also be thought of as a special case of Euclidean distance or a sum modulo 2, often indicated also with $\oplus$. XOR is commutative and the product is its own inverse: $A * A = O$, where O is the vector with all 0s and it is the unit vector, as $A * O = A$.

It is easy to see that the Hamming distance between two vectors is simply the count of 1s in their product (denoted by $|\cdot|$): $d(A,B) = |A * B|$. Multiplication can be thought of as a mapping: multiplying X with A maps it to the vector $X_A = X * A$ which is as far 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 show that multiplication randomizes. Besides, multiplication preserves distance:

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 by multiplying with the same vector, the relative distances are maintained (the cluster of points is just moved to a different part of the space).

High-level cognitive functions such as analogy could 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 as a specific mapping applied to vectors X and Y. The same argument applies considering two mappings A and B, and examining their effect on the same vector X: X would be mapped onto X_A and X_B that are as distant as A and B. Therefore, similar vectors lead to similar mappings; similar mappings map any vector to similar vectors.

Permutation simply reorders the vector components. The permutation of a vector can be denoted as function application ($\rho(\cdot)$) or a multiplication by a corresponding *permutation matrix* Π), thus we can write $\rho(A) = \Pi A$.

Alternatively it may also 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. The distances between points 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 of is chosen randomly from the $n!$ possible permutations. Given random permutations ρ_1 and ρ_2, they can be shown to agree in only one position on the average, and so the distance between $\rho_1(A)$ and $\rho_2(A)$ is approximately 0.5, so they map the vector to different parts of the space.

Associations: Pairs and Bindings. A *pair* is a basic unit of association, when 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 with 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 desired properties discussed 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 in the same way as a multiplication of the variable-vector X and the value-vector A: $X * A$. Then unbinding amounts to simply multiply for one of the two to get the other element.

Records as Sets of Bound Pairs. A complex individual can be represented as a record composed of fields. Each field in the record represents a variable (a role) that is a relationship to a value or another individual.

In a holistic representation the roles are explicitly represented as vectors. Vectors for unrelated roles, e.g. `name` and `address`, can be chosen at random. Again, a role bound to its filler will then be represented by the product $X * A$ defined as shown above. A single entity can be defined with a record that combines several 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-value pairs explicitly. Moreover, it will be similar to all of its pairs, while it will be separated (dissimilar) by the single variable- and value-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, nothing similar to the three products is 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* [31] 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, 32] 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 classic star-shaped (undirected) graph $\mathcal{G}$ with 5 nodes $\{A, B, C, D, E\}$ can be represented as the following hyper-vector: $G = A * C + A * D + B * D + B * E + C * E$.*

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 from node, say A to B in G is represented as $A * \rho(B)$.

Example 4.6. *A directed graph $\mathcal{G}'$ similar to $\mathcal{G}$ in the previous example can be represented as follows:* $G' = A * \rho(C) + A * \rho(D) + B * D + \rho(B) * \rho(E) + E * \rho(C)$.

Graphs can be used to represent scenes or episodes for analogy reasoning (as shown in [32], §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 analogies are described in the form: $A : B :: C : ?$, 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 entity). Typical such problems are related to the domain of images, *visual analogy problems*, tackled in several works (e.g. see [55]).

The method of computing such a relation in hyperdimensional representations is extremely simple, namely the subtraction between the hypervectors for A and B. This lacks a full appreciation for the nature of analogy, namely the substitution of atomic components which may be embedded in larger data structures. This computation is not well-captured by the mere subtraction. However, in a HD/VSA framework, substitution of atomic components within/between complex data structures can be both elegant and very efficient when the data structures are constructed through *binding*. The operation of *unbinding* reveals analogical relations that can be used in various applications as discussed in [27, 51].

Holistic transformations for solving these problems are commonly illustrated using the well-known case of the "Dollar of Mexico" [27] in which a simple proportional analogy: `US : México :: Dollar : ?`. This kind of analogies is known to be solvable by addition and subtraction of the embeddings of the corresponding concepts [40, 49]. Similarly, in [48] 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 representational 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 HRR model was investigated in [43, 44]. A transformation hypervector is obtained from several training pairs of hypervectors as a solution of an optimization problem found through gradient descent, iterating through all examples until convergence. The learned transformation hypervector was 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 BSC representations are illustrated in [25]. 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 resolved by using either the permutation or an additional associative memory as 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 worth of further investigation.

Analogical Reasoning. In Cognitive Science *analogical reasoning* theories [13] deal with analogical *episodes* (or *analogs*) and usually comprise (models for) a process[5] comprising the following four basic steps:

[5] In the context of Machine Learning this process is also known as *case-based reasoning* [36]. Actually *instance-based learning*, *similarity-based* predictions made by memorizing prototypes may be ascribed to this form of learning [41].

1. The *description* of episodes which can be modeled as systems of hierarchical relations consisting of elements in the form of entities and relations of various 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. Once such base episode/s is/are identified, the *mapping* step determines the correspondences between the elements of query and base episodes.
4. The *inference* step transfers knowledge from the base analogical episode(s) to the target analogical episode. This new knowledge obtained about the target may, e.g., provide the solution to the problem specified by the query episode. Candidate inferences have to be treated as hypotheses and must be evaluated and checked [12].

Two types of similarity concern the processing of analogical episodes. *Structural similarity* reflects the relations between the various elements of the episodes. Episodes are also matched in terms of a *superficial similarity* based on common elements in episodes or in terms of a more *semantic similarity*, that is based, for example, on the similarity of characteristic feature vectors or on joint membership in a taxonomic category.

HDC/VSA models have been used for *analogical retrieval* [51] 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 [31] various studies are described proving that results obtained by a similarity estimation on hypervectors are consistent with both the empirical results in psychological experiments as well as the leading traditional models of the analogical retrieval.

As regards *analogical mapping*, models of mapping based on HRR and BSC were proposed that use techniques based on holistic transformations (again, see [31]). 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 were demonstrated on complex analogies. However, the techniques are rather sophisticated and used sequential operations.

Interestingly, the hyperdimensional vector models adopted for analogical reasoning feature a compatibility with the existing well-established formats of knowledge representation. This facilitates the unification of symbolic and sub-symbolic approaches for cognitive modeling and AI. A interesting study in this line presented a proof of concept of the mapping between RDF-Schema ontology and a HDC/VSA model [39].

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 [31]. 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 the 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, 31] 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. Then higher-level headings have suggested further directions for the application of HDC to classification, focusing, respectively, on the *types* of input data (second-level headings) and on the *domains* (third-level headings).

In a basic classification methodology [11], 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 check is carried out in the associative memory comparing the query hypervector to each class hypervector. Finally, the label of the class with the smallest distance is returned.

Simple models based on centroids are known to fall short in terms of generalization [29]. Improvements can be obtained by weighting newly assigned instances to the centroids [15]. It has been pointed out that conventional classifiers assuming that any component of the input vector can be interpreted on its own, a reasonable assumption when components are features endowed with meaningful interpretations, generally not holding for HD representations [31]. Sparse HDC/VSA representations may be adequate for classifiers that benefit from sparsity such as *Winnow* [35] and *sparse SVMs* [6].

Devising suitable encodings for good classifiers is a challenging task [9] 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 [53]. Alternatively, different encoding methods can be integrated together to achieve higher accuracy [20]. Compared to single-pass training, *retraining* iteratively has been demonstrated to improve the accuracy of the classification models [21]. This suggests the direction of investigating ensemble (hierarchical) models of encoders (for a recent ensemble learning algorithm see [61]).

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* (ART) [2] or *Self-Organizing Map* (SOM) [33]. These can be thought as local models for the input density which may be interesting to transpose to a HD representation. Probability density estimation (and other tasks such as kernel smoothing) have been shown to be efficiently tackled resorting to *fractional power encoding* [7]. An efficient algorithm in the context of such vector spaces has been presented in [16].

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* (RBF-nets) [42] or the *Mixture of Experts* (MoE) [23]. This layer implements a local model for the generation of the output (classification/regression). Centroids have been shown to be easily combined with generalized LVQ classifiers [5]. Besides, moving from the idea of *multiple-class hypervectors*, disjunctive or mixture hierarchical representations of the classes may be targeted to better represent them [11].

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 also methods for the induction of (probabilistic) rules based on local models (basis functions) [59] may be also implemented on HDC-VSAs.

Structured data are generally more difficult to handle with conventional ML 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) as hypervectors. Related to this task, we will deal with the case of problems regarding embedding models and knowledge graphs in the following.

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 [24]). The approach presented in [37], showed that transforming 2D molecular structures it was possible to construct classifiers for drug discovery problems that outperformed the baseline methods on a collection of tasks. More generally, the problem of classifying graphs using HDC/VSA is another promising direction for further investigations. In [47] 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 in Machine Learning [41]. 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.

Without loss of generality, it is possible to express logic *atoms* (binary relations $p(h,t)$) as *triples* $\langle h, p, t\rangle$ (which is common in the representations for KGs). One can define it as a vector in the following way: $P_{xy} = P_1 * X + P_2 * Y$:

Example 6.1. *An atom expressing the relation* $\langle y, \texttt{brother\_of}, z\rangle$ *can be represented by* $B_{yz} = B_1 * Y + B_2 * Z$. *An atom* $\langle y, \texttt{child\_of}, z\rangle$ *can be represented by* $C_{yz} = C_1 * Y + C_2 * Z$. *An atom* $\langle x, \texttt{uncle\_of}, y\rangle$ *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 a 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, with 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 simply by addition, we get 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 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 [34]. HD models have been used to represent a knowledge base with (propositional) clauses for further performing deductive inference on them [57] including also negation. By extending the representation with the explicit representation of variables can lead to new methods the induction of FOL rule bases or more recent statistical relational models [4]. Devising appropriate transformations, also other types of kinds of knowledge bases may be targeted, such as ontologies and knowledge graphs expressed by axioms in Description Logics [1].

6.2. Embedding Models and Knowledge Graphs

The *distributional semantics hypothesis* [14], suggested that linguistic items with similar distributions have similar meanings is the basis for the idea for constructing context vectors. This has been later extended to the semantics of other entities such as concepts and relations. In principle, context vectors can be obtained in any domain where objects and contexts could be defined. Context vector methods that are commonly attributed to HDC/VSA usually transform the frequency distributions into hyper-vectors of particular formats, *context hypervectors*.

Examples of successful *semantic indexing* methods based on context hypervectors are *Predication-based Semantic Indexing* [62] and its extension *Embedding of Semantic Predications* [3], as well as BEAGLE [54]. In [52] the authors address the representation of similarity: for each word its most relevant semantic features taken from *ConceptNet* are represented. The context hypervector of a word are produced 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 the domain of linguistics to more general domains for new downstream applications. The plethora of *2VEC models stemmed from the ideas presented in [40] are natural candidates for such transpositions.

We have seen (§4) that graphs are general data structures that can be encoded in HD vector representations. *Graph Neural Networks* (GNNs) [56] are the typical models for graph representations extending 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* [47] which are achieving comparable effectiveness with lesser computational cost compared to current deep neural models. As methods based on graph-kernels have shown comparable results [63], it seems legit to investigate such methods combined with HD-computing.

Knowledge Graphs (KGs) [19] 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 on the underlying terminology (see [1]). Inference and refinement tasks on KGs have been tackled exploiting the natural decomposition of KGs in triples that has given rise to many *embedding models* (see §6.2) mapping entities and properties to low-dimensional spaces where such complex tasks can be approximated by geometrical operations on embedding vectors and matrices.

Given suitable encodings of triples to hypervectors, such as those used in §6.1, it is possible to devise similar methods to learn HD representations for KGs to perform the mentioned tasks. 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 [49]). Hypervectors of KGs can also be constructed from those of nodes and relations that are learned from data as proposed in [46], where the HRR model [50] was adopted due to its differentiability. Representations for KGs have been further studied in [38] where a Cauchy distribution is used to generate atomic hypervectors and demonstrated the state-of-the-art results on the task of inferring missing links using hypervectors for KGs as the input to a neural network.

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 improvements. These lines include (but are not limited to):

- Feature exaction and encoding methods: these are essential activities as hyperdimensional models cannot successfully handle complex data without proper encodings. Related to the density of hypervectors, a choice between dense and sparse approaches should be accordingly made based on the application scenarios: adopting sparse representation requires lower memory footprints. *Nonlinearity* is another important aspect of transforming data into hypervectors. A lack of studies on nonlinearity properties of hypervectors obtained from the compositional approach has been recognized (see [31], §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 type of kernel to be adopted for kernel-based machines is also related to this issue. Kernels are related to the choice of the HD transformation, as investigated in [7]. The transformation may be considered as another hyperparameter in a optimization problem which may be determined with the standard statistical approaches such as *cross-validation* on.
- *Learning strategies* like *retraining* [11, 22] should be further explored to improve the accuracy of the HD classification models. This objective can be pursued through *hybrid systems*, i.e. models that combine conventional ML methods with HD computing models (e.g. see [30]).
- Complex neural models such as deep neural architectures used in combination of VSAs call for specific XAI approaches to make such black-boxes more transparent and interpretable and their decisions more easily comprehensible.
- Engaging with other *cognitive tasks*: Apart from the engineering aspects of HD computing, more cognitive aspects of HD computing should be explored. Such tasks 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 which would require more attention as it can be exploited for various reasoning and learning tasks such as diagnostics, hypothesis creation, relevance detection and explanation.
- The interplay with models for *uncertainty* and *causality*: neural networks can be related to specific graphical models [4]; also probabilistic causal models may suggest further directions for studies on the transposition of forms of reasoning under uncertainty to HDC/VSAs (e.g. see [8]).
- *HW acceleration*: Rebuilding the specific implementation for HD computing to store and manipulate a large amount of hypervectors to improve speed and energy efficiency. A *general* HD computing *processor* has also been envisioned [11] to address different types of data with only one general processor containing a large word-length ALU.

References

[1] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2nd edition, 2007.

[2] G. A. Carpenter and S. Grossberg. The art of adaptive pattern recognition by a self-organizing neural network. *Computer*, 21(3):77–88, 1988.

[3] T. Cohen and D. Widdows. Embedding of semantic predications. *Journal of Biomedical Informatics*, 68:150–166, 2017.

[4] L. De Raedt, K. Kersting, S. Natarajan, and D. Poole. *Statistical Relational Artificial Intelligence: Logic, Probability, and Computation*. Morgan & Claypool, 2016.

[5] 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)*, pages 1–9, 2021.

[6] 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.

[7] 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*, page 115–122. ACM, 2022.

[8] 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*, page 259–266, 2022.

[9] R. W. Gayler. Vector symbolic architectures answer Jackendoff's challenges for cognitive neuroscience. In P. Slezak, editor, *Proceedings of the ICCS/ASCS Joint International Conference on Cognitive Science (ICCS/ASCS 2003)*, pages 133–138, 2003.

[10] R. W. Gayler and S. D. Levy. A distributed basis for analogical mapping. In *Proceedings of the 2nd International Analogy Conference*, pages 165–174, 2009.

[11] L. Ge and K. K. Parhi. Classification using hyperdimensional computing: A review. *IEEE Circuits and Systems Magazine*, 20(2):30–47, 2020.

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

[13] D. Gentner and K. D. Forbus. Computational models of analogy. *WIREs Cognitive Science*, 2(3):266–276, 2011.

[14] Z. Harris. *Mathematical Structures of Language*. Wiley, 1968.

[15] 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*, page 56–61, 2021.

[16] 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*, pages 1859–1864, 2021.

[17] 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):363–375, 2023.

[18] G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. *Distributed Representations*, page 77–109. MIT Press, 1986.

[19] 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.

[20] 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)*, pages 1–6, 2018.

[21] 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)*, pages 1–8, 2017.

[22] 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)*, pages 1–4, 2019.

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

[24] 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, 2023. arXiv: doi: 10.48550/arXiv.2303.15604 https://arxiv.org/abs/2303.15604.

[25] P. Kanerva. Large patterns make great symbols: An example of learning from example. In S. Wermter and R. Sun, editors, *Hybrid Neural Systems*, pages 194–203. Springer, 2000.

[26] P. Kanerva. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation*, 1(2):139–159, 2009.

[27] 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.

[28] P. Kanerva. Computing with high-dimensional vectors. *IEEE Design & Test*, 36(3):7–14, 2019.

[29] J. Karlgren and P. Kanerva. Semantics in high-dimensional space. *Front. Artif. Intell.*, 4, 2021.

[30] 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):3777–3783, 2021.

[31] 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.

[32] 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.
[33] T. Kohonen. The self-organizing map. *Neurocomputing*, 21(1):1–6, 1998.
[34] V. Kvasnička and J. Pospíchal. Deductive rules in holographic reduced representation. *Neurocomputing*, 69(16):2127–2139, 2006.
[35] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Mach. Learn.*, 2:285–318, 1988.
[36] B. López. *Case-Based Reasoning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Springer, 2013.
[37] D. Ma, R. Thapa, and X. Jiao. MoleHD: Efficient drug discovery using brain inspired hyperdimensional computing. In D. A. Adjeroh et al., editors, *IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2022*, pages 390–393. IEEE, 2022.
[38] 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*, pages 403–413. AUAI Press, 2018.
[39] 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.
[40] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In C. Burges et al., editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.
[41] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.
[42] J. Moody and C. J. Darken. Fast Learning in Networks of Locally-Tuned Processing Units. *Neural Computation*, 1(2):281–294, 1989.
[43] J. Neumann. Learning holistic transformation of hrr from examples. In *Proceedings KES'2000, Fourth International Conference on Knowledge-Based Intelligent Engineering Systems*, volume 2, pages 557–560, 2000.
[44] J. Neumann. Learning the systematic transformation of holographic reduced representations. *Cognitive Systems Research*, 3(2):227–235, 2002. Integration of Symbolic and Connectionist Systems.
[45] A. Newell and H. A. Simon. Computer science as empirical inquiry: Symbols and search. *Commun. ACM*, 19(3):113–126, 1976.
[46] M. Nickel, L. Rosasco, and T. Poggio. Holographic embeddings of knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.
[47] I. Nunes, M. Heddes, T. Givargis, A. Nicolau, and A. Veidenbaum. GraphHD: Efficient graph classification using hyperdimensional computing. In C. Bolchini et al., editors, *Proceedings of DATE '22, Conference on Design, Automation & Test in Europe*, pages 1485–1490. IEEE, 2022.
[48] 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*, pages 38–48. ACL, 2020.
[49] 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)*, pages 1532–1543. Association for Computational Linguistics, 2014.
[50] T. Plate. *Holographic Reduced Representations: Distributed representation of cognitive structure*. Number 150 in CSLI Lecture Notes. CSLI Publications, Stanford, 2003.
[51] T. A. Plate. Analogy retrieval and processing with distributed vector representations. *Expert Systems*, 17(1):29–40, 2000.
[52] J. I. Quiroz-Mercado, R. Barrón-Fernández, and M. A. Ramírez-Salinas. Semantic similarity estimation using vector symbolic architectures. *IEEE Access*, 8:109120–109132, 2020.
[53] 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):123–143, 2019.
[54] 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.
[55] S. E. Reed, Y. Zhang, Y. Zhang, and H. Lee. Deep visual analogy-making. In C. Cortes et al., editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.
[56] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.
[57] D. Summers-Stay. Propositional deductive inference by semantic vectors. In Y. Bi et al., editors, *Intelligent Systems and Applications*, pages 810–820. Springer, 2020.
[58] S. J. Thorpe. Localized versus distributed representations. In *Handbook of Brain Theory and Neural Networks*, page 643–646. MIT Press, 2003.
[59] V. Tresp, J. Hollatz, and S. Ahmad. Representing probabilistic rules with networks of Gaussian basis functions. *Machine Learning*, 27:173–200, 1997.
[60] N. Wang, J. Quek, T. Rafacz, and S. Patel. Characterizing the effects of transient faults on a high-performance processor pipeline. In *International Conference on Dependable Systems and Networks, 2004*, pages 61–70, 2004.
[61] R. Wang, D. Ma, and X. Jiao. EnHDC: Ensemble learning for brain-inspired hyperdimensional computing. *IEEE Embed. Syst. Lett.*, 15(1):37–40, 2023.
[62] D. Widdows and T. Cohen. Reasoning with vectors: A continuous model for fast robust inference. *Logic Journal of the IGPL*, 23(2):141–173, 11 2014.
[63] 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.