A Survey on Hyperdimensional Computing aka Vector Symbolic Architectures, Part II: Applications, Cognitive Models, and Challenges

Denis Kleyko, Dmitri A. Rachkovskij, Evgeny Osipov, and Abbas Rahimi

Abstract

This is Part II of the two-part comprehensive survey devoted to a computing framework most commonly known under the names Hyperdimensional Computing and Vector Symbolic Architectures (HDC/VSA). Both names refer to a family of computational models that use high-dimensional distributed representations and rely on the algebraic properties of their key operations to incorporate the advantages of structured symbolic representations and vector distributed representations. Holographic Reduced Representations [1], [2] is an influential HDC/VSA model that is well-known in the machine learning domain and often used to refer to the whole family. However, for the sake of consistency, we use HDC/VSA to refer to the field.

Part I of this survey [3] covered foundational aspects of the field, such as the historical context leading to the development of HDC/VSA, key elements of any HDC/VSA model, known HDC/VSA models, and the transformation of input data of various types into high-dimensional vectors suitable for HDC/VSA. This second part surveys existing applications, the role of HDC/VSA in cognitive computing and architectures, as well as directions for future work. Most of the applications lie within the Machine Learning/Artificial Intelligence domain, however, we also cover other applications to provide a complete picture. The survey is written to be useful for both newcomers and practitioners.

Index Terms

Artificial Intelligence, Machine learning, Distributed representations, Cognitive architectures, Cognitive computing, Applications, Analogical reasoning, Hyperdimensional Computing, Vector Symbolic Architectures, Holographic Reduced Representations, Tensor Product Representations, Matrix Binding of Additive Terms, Binary Spatter Codes, Multiply-Add-Permute, Sparse Binary Distributed Representations, Sparse Block Codes, Modular Composite Representations, Geometric Analogue of Holographic Reduced Representations

We would like to thank three reviewers, the editors, and Pentti Kanerva for their insightful feedback as well as Linda Rudin for the careful proofreading that contributed to the final shape of the survey. The work of DK was supported by the European Union’s Horizon 2020 Programme under the Marie Skłodowska-Curie Individual Fellowship Grant (839179). The work of DK was also supported in part by AFOSR FA9550-19-1-0241 and Intel's THWAI program. The work of DAR was supported in part by the National Academy of Sciences of Ukraine (grant no. 0120U000122, 0121L000016, and 0117U002286), the Ministry of Education and Science of Ukraine (grant no. 0121U000228 and 0122U000818), and the Swedish Foundation for Strategic Research (SSF, grant no. UKR22-0024). Denis Kleyko and Dmitri A. Rachkovskij contributed equally to this work.

D. Kleyko is with the Redwood Center for Theoretical Neuroscience at the University of California, Berkeley, CA 94720, USA and also with the Intelligent Systems Lab at Research Institutes of Sweden, 16440 Kista, Sweden. E-mail: denkle@berkeley.edu
D. A. Rachkovskij is with International Research and Training Center for Information Technologies and Systems, 03680 Kiev, Ukraine. E-mail: dar@infrm.kiev.ua
E. Osipov is with the Department of Computer Science Electrical and Space Engineering, Luleå University of Technology, 97187 Luleå, Sweden. E-mail: evgeny.osipov@ltu.se
A. Rahimi is with IBM Research, 8803 Zurich, Switzerland. E-mail: abr@zurich.ibm.com
## CONTENTS

1 Introduction 3

2 Application areas 3
   2.1 Deterministic behavior with HDC/VSA 4
      2.1.1 Automata, instructions, and schemas 4
      2.1.2 Transmission of data structures 5
      2.1.3 String processing 6
      2.1.4 Factorization 6
   2.2 Similarity estimation with HVs 6
      2.2.1 Word embeddings: context HVs for words and texts 7
      2.2.2 Similarity estimation of biomedical signals 8
      2.2.3 Similarity estimation of images 8
   2.3 Classification 9
      2.3.1 Classification based on feature vectors 9
      2.3.2 Classification of images or their properties 11
      2.3.3 Classification of structured data 11

3 Cognitive computing and architectures 11
   3.1 Cognitive computing 12
      3.1.1 Holistic transformations 12
      3.1.2 Analogical reasoning 14
      3.1.3 Cognitive modeling 16
      3.1.4 Computer vision and scene analysis 17
   3.2 Cognitive architectures 18
      3.2.1 Semantic Pointer Architecture Unified Network 18
      3.2.2 Associative-Projective Neural Networks 18
      3.2.3 Hierarchical Temporal Memory 19
      3.2.4 Learning Intelligent Distribution Agent 19
      3.2.5 Memories for cognitive architectures 19

4 Discussion 20
   4.1 Application areas 20
      4.1.1 Context HVs 20
      4.1.2 Classification 20
      4.1.3 Real-world use-cases and new application areas 21
   4.2 Interplay with neural networks 21
      4.2.1 HVs as input to neural networks 21
      4.2.2 The use of neural networks for producing HVs 22
      4.2.3 HDC/VSA for simplifying neural networks 22
      4.2.4 HDC/VSA for explaining neural networks 22
      4.2.5 The use of HDC/VSA with spiking neural networks 23
      4.2.6 “Hybrid” solutions 23
   4.3 Open issues 23

5 Conclusion 25

References 25
TABLE 1: A qualitative assessment of existing HDC/VSA literature that have some elements of survey. ✓ means that the article overviewed the area rather comprehensively, ✗ means that the area was not covered at all while ± indicates that the article partially addressed a particular topic, but either new results were reported since then or not all related work was covered.

|                      | Deterministic behavior | Similarity estimation | Classification | Cognitive computing | Cognitive architectures |
|----------------------|------------------------|-----------------------|-----------------|---------------------|------------------------|
| Automated decision   | ✓                      | ✓                     | ✗               | ✓                   | ✓                      |
| Data transmission    | ✓                      | ✓                     | ✗               | ✓                   | ✓                      |
| Speech processing    | ✓                      | ✓                     | ✗               | ✓                   | ✓                      |
| Computer vision      | ✓                      | ✓                     | ✗               | ✓                   | ✓                      |
| Biomedical signals   | ✓                      | ✓                     | ✗               | ✓                   | ✓                      |
| Images               | ✓                      | ✓                     | ✗               | ✓                   | ✓                      |
| Textual vectors      | ✓                      | ✓                     | ✗               | ✓                   | ✓                      |
| Image of other properties | ✓ | ✓ | ✗ | ✓ | ✓ |
| Structural data      | ✓                      | ✓                     | ✗               | ✓                   | ✓                      |
| Holistic transformations | ✓ | ✓ | ✗ | ✓ | ✓ |
| Biological modeling  | ✓                      | ✓                     | ✗               | ✓                   | ✓                      |
| Cognitive modeling   | ✓                      | ✓                     | ✗               | ✓                   | ✓                      |
| Cognitive analysis   | ✓                      | ✓                     | ✗               | ✓                   | ✓                      |
| Computer vision and scene analysis | ✓ | ✓ | ✗ | ✓ | ✓ |
| Semantic Pointer     | ✓                      | ✓                     | ✗               | ✓                   | ✓                      |
| Associative-Projective Neural Networks | ✓ | ✓ | ✗ | ✓ | ✓ |
| Hierarchical Memories for cognitive architectures | ✓ | ✓ | ✗ | ✓ | ✓ |
| Cognitive Distribution Agent | ✓ | ✓ | ✗ | ✓ | ✓ |
| Measuring for cognitive architecture | ✓ | ✓ | ✗ | ✓ | ✓ |

This survey, Part II
Section # 2.1.1 2.1.2.1 2.1.3 2.2.1 2.2.2 2.2.3 2.3.1 2.3.2 2.3.3 3.1.1 3.1.2 3.1.3 3.1.4 3.2.1 3.2.2 3.2.3 3.2.4 3.2.5

1 INTRODUCTION

This article is Part II of the survey of a research field known under the names Hyperdimensional Computing, HDC (the term was introduced in [4]) and Vector Symbolic Architectures, VSA (the term was introduced in [5]). As in Part I [3], below we will consistently use the joint name HDC/VSA when referring to the field. HDC/VSA is an umbrella term for a family of computational models that rely on mathematical properties of high-dimensional random spaces and use high-dimensional distributed representations called hypervectors (HVs) for a structured (“symbolic”) representation of data, while maintaining the advantages of traditional connectionist vector distributed representations.

First, let us briefly recapitulate the motivation for this survey. The main driving force behind the current interest in HDC/VSA is the global trend of searching for computing paradigms alternative to the conventional (von Neumann) ones. Examples of the new paradigms are neuromorphic and nanoscalable computing, where HDC/VSA is expected to play an important role (see [6] and references therein for perspective). Due to this surge of interest in HDC/VSA, the need for providing a broad overview of the field, which is currently missing, became evident. Therefore, this two-part survey extensively covers the state-of-the-art of the field in a form that is accessible to a wider audience.

There were no previous attempts to make a comprehensive survey of HDC/VSA, but there are articles that overview particular topics of HDC/VSA. Probably the first attempt to overview and unify different HDC/VSA models should be attributed to Plato [7]. The key idea for the unification was to consider the existing (at that time, four) HDC/VSA models as different schemes for implementing two key operations: binding and superposition (see Section 2.2.3 in [3]). However, since that time numerous HDC/VSA models have come to prominence. A more recent summary of the most frequently used models was provided in [8]. In [12], the HDC/VSA models were compared in terms of their realizations of the binding operation. Both articles, however, missed some of the models. These and other gaps have been filled in Part I of this survey [3].

As for applications of HDC/VSA – the topic covered in this article – in Table 1 we identified the following substantial application domains, which reflect the structure of Sections 2 and 3: deterministic behavior, similarity estimation, classification, cognitive computing, and cognitive architectures. The columns in Table 1 list more fine-grained application clusters within these larger domains.

There is no previous article, which would account for all currently known applications, though there are recent works overviewing either a particular application area (as in [9], where the focus was on biomedical signals), or certain application types (as in [11], [13], where solving classification tasks with HDC/VSA was the main theme). The topic of machine learning is also omnipresent in this survey, and due to its ubiquity we dedicated Section 2.3 to classification tasks. However, the scope of the survey is much broader as it touches on all currently known applications. Table 1 contrasts the coverage of Part II of this survey with the previous articles (ordered chronologically). We use ± to indicate that an article partially addressed a particular topic, but either new results were reported since then or not all related work was covered.

In Part I of this survey [3], we considered the motivation behind HDC/VSA and basic notions, summarized currently known HDC/VSA models, and presented the transformation of various data types into HVs. Part II of this survey covers existing applications (Section 2) and the use of HDC/VSA in cognitive modeling and architectures (Section 3). The discussion and challenges, as well as conclusions, are presented in Sections 4 and Section 5, respectively.

2 APPLICATION AREAS

HDC/VSA have been applied across different fields for various tasks. For this section, we aggregated the existing applications into several groups: deterministic behavior (Section 2.1), similarity estimation (Section 2.2), and classification...
2.1 Deterministic behavior with HDC/VSA

In this section, we consider several use-cases of HVs designed to produce some kind of deterministic behavior. Note that due to the capacity limitations of HVs (see Section 2.4 in [3]), achieving deterministic behavior depends on several design choices. These include the dimensionality of HVs as well as, e.g., the number of atomic HVs and the kind of rules used for constructing compositional HVs, such as the number of arguments in the superposition operation. Note also that strictly speaking, not all application areas listed here are perfectly deterministic (in particular, communications in Section 2.1.2.1), but the determinism is a desirable property in all the areas collected in this section.

2.1.1 Automata, instructions, and schemas

2.1.1.1 Finite-state automata and grammars: A deterministic finite-state automaton is specified by defining a finite set of states, a finite set of allowed input symbols, a transition function (defines all transitions in the automaton), a start state, and a finite set of accepting states. The current state can change in response to an input. The joint current state and input symbol uniquely determine the next state of the automaton.

An intuitive example of an automaton controlling the logic of a turnstile is presented in Fig. 1. The set of input symbols is \{“Token”, “Push”\} and the set of states is \{“Unlocked”, “Locked”\}. The state diagram in Fig. 1 can be used to derive the transition function.

HDC/VSA-based implementations of finite-state automata were proposed in [14], [15]. Random HVs are assigned to represent states (\(u\) for “Unlocked”; \(l\) for “Locked”) and input symbols (\(t\) for “Token”; \(p\) for “Push”). These HVs are used to form a compositional HV \(a\) for the transition function. The transformation is similar to the one used for the directed graphs in Section 3.5.1 in [3]. However, the HV representing the input symbol for the automaton is bound to the edge HV that corresponds to the binding of the HVs for the current and the next state. For instance, going from “Locked” to “Unlocked” upon receiving “Token”, is represented as:

\[
\rho^{-1}(a \circ p \circ l) = l + \text{noise}. \tag{3}
\]

This noisy HV is used as the query for the item memory to obtain the noiseless atomic HV \(l\).

Transformations of pushdown automata and context-free grammars into HVs have been presented in [16]. A proposal for implementing Turing machines and cellular automata was given in [6].

In [17], the Holographic Reduced Representations (HRR) model was used to represent Fluid Construction Grammars, which is a formalism that allows designing construction grammars and using them for language parsing and production. Another work related to parsing is [18], which presented an HDC/VSA implementation of a general-purpose left-corner parsing with simple grammars. An alternative approach to parsing with HVs using a constraint-based parser has been presented in [19].

A related research direction was initiated by beim Graben and colleagues [16], [20]-[22]. It concerns establishing a mathematically rigorous formalism of Tensor Product Representations using the concept of a Fock space [23]. The studies focus largely on use-cases in computational linguistics, semantic processing, and quantum logic. The usage of the Fock space formalism for formulating minimalist grammars was presented in [20]. Syntactic language processing as part of phenomenological modeling was reported in [21]. An accessible and practical entry point to the area can be found in [22].

1. It is worth recalling that this and other applications use a common design pattern, relying on the unbinding operation (see Section 2.2.3.2 in [3]) that allows recovering one of the arguments. In the case of permutations, it is due to the fact that \(a = \rho^{-1}(\rho(a))\), while in the case of multiplicative binding \(a = b \otimes (a \circ b)\), where for the implementations with self-inverse binding \(\otimes = \circ\).
2.1.1.2 Controllers, instructions, schemas: In [24], using the Multiply-Add-Permute model (MAP), it was demonstrated how to manually construct a compositional HV that implements a simple behavior strategy of a robot. Sensor inputs as well as actions were represented as atomic HVs. Combinations of sensor inputs as well as combinations of actions were represented as a superposition of the corresponding atomic HVs. HVs of particular sensor input combinations were bound to the corresponding HV of action combinations. The bound HVs of all possible sensor-action rules were superimposed to produce the compositional HV of the robot controller. Unbinding this HV with the current HV of sensor input combinations results in the noisy HV of the proper action combination. This idea was extended further in [25] through a proposed algorithm to “learn” a compositional HV representing a robot’s controller using the sensor-actuator values obtained from successful navigation runs. This mode of robot operation is known as “learning by demonstration”.

It was realized as the superposition of the current “controller” HV with the HV corresponding to the binding of the sensor-actuator values - in case the current sensor HV was dissimilar to the ones already present in the “controller” HV. Another work studying robot navigation is [26], which investigated a number of ways to form compositional HVs representing sensory data and explored the integration of the resultant HVs with neural network instead of the “controller”.

In [27], using HRR, instructions were represented as a sequence of rules, and rules as a sequence of their antecedent and consequent elements. Multiplicative bindings with positions HVs were used to represent the sequence. Antecedents and consequents, in turn, were represented as HVs of their elements using binding and superposition operations. This approach was used as a part of instruction parsing in a cognitive architecture [28]. In [29], a proposal was sketched for a HDC/VSA-based processor, where both data and instructions were represented as HVs.

In [30], HVs for the representation of “schemas” in the form <context, action, result> were used. A more general approach for modeling the behavior of intelligent agents as “functional acts” was considered in [31] for Sparse Binary Distributed Representations (SBDR; see also Section 3.2.2). It is based on HVs representing triples <current situation, action, resulting situation> (which essentially correspond to “schemas”), with the associated evaluations and costs. Finally, it is worth recalling that, in general, data structures to be represented by HVs do not have to be limited to “schemas”. For example, a recent proposal in [32] suggested that HVs are well suited for forming representations of the JSON format that can include several levels of hierarchy.

2.1.1.3 Membership query and frequency estimation: Section 3.1.2 in [3] presented the transformation of sets and multisets into HVs. When implemented with the SBDR model, it becomes evident that Bloom Filters [33] are a special case of HDC/VSA [34], which have been used in a myriad of applications involving the membership query. It is beyond the scope of this survey to overview them all, therefore, the interested readers are refereed to a survey in [35]). When implementing the transformation of multisets via the Sparse Block Codes model, a similar connection can be made to count-min sketch [36] that is used commonly for estimating frequency distributions in data streams (some applications are presented in [36]). The use of the HDC/VSA principles for constructing hash tables has been recently considered in [37].

2.1.2 Transmission of data structures

2.1.2.1 Communications: The main motivation for using HDC/VSA in the communication context is their robustness to noise due to the distributed nature of HVs. Let us consider three similar but varying applications of HDC/VSA.

In [38], it was shown how to use Binary Spatter Codes (BSC) (Section 2.3.6 in [3]) in collective communication for sensing purposes. Multiple devices wirelessly sent their specific HVs representing some of their sensory data (the paper used temperature as a show-case). It was proposed to receive them in a manner that implements the superposition operation. This superposition HV was then analyzed by calculating the distance of the normalized superposition and the atomic HVs and comparing with a threshold. For instance, the case that a particular temperature was transmitted could be detected. Another version of analysis allowed checking how many devices had been exposed to a particular temperature. The proposed communication scheme does not require a control mechanism for getting multiple access to the medium. So, it can be useful in scenarios where there are multiple devices that have to report their states to some central node. Recently, it has been shown how such over-the-air superposition can be used for on-chip communications to scale up the architecture with multiple transmitters and receivers [39]. This has been done by carefully engineering modulation constellations, and it paves the way for a large number of physically distributed associative memories (as wireless-augmented receivers) to reliably perform the similarity search given a slightly different version of a query HV as their input.

In [40], BSC was used in the context of medium access control protocols for wireless sensor networks. A device was forming a compositional HV representing the device’s sensory data (by the superposition of multiplicative bindings), which was then transmitted to the communication medium. It was assumed that the receiver knew the atomic HVs and, thus, could recover the information represented in the received compositional HV. The application scope of this approach is for scenarios where the communication medium is very harsh so that a high redundancy of HVs is useful for reliably transmitting the data.

In [41], it was proposed to combine forward error correction and modulation using Fourier HRR (Section 2.3.5 in [3]). The scheme represented individual pieces of data by using complex-valued HVs that were then combined into a compositional HV using permutation and superposition operations. Unnormalized complex values of the compositional HV were transmitted to the communication medium. The iterative decoding of the received compositional HV significantly increased the code rate. The application scope of this scheme is robust communication in a low signal-to-noise ratio regime. The scheme at a lower coding rate was compared to the low-density parity check and polar codes in terms of achieved bit error rates, while featuring lower decoding complexity. To improve its signal-to-noise ratio gain, a soft-feedback iterative
decoding was proposed [42] to additionally take the estimation’s confidence into account. That improved the signal-to-noise ratio gain by 0.2 dB at a bit-error-rate of $10^{-4}$. In further works, the scheme has been applied to collision-tolerant narrowband communications [43], massive machine type communications [44], and near-channel classification [42].

2.1.2.2 Distributed orchestration: Another use-case of HDC/VSA in the context of transmission of data structures is distributed orchestration. The key idea presented in [45], [46] was to use BSC to communicate a workflow in a decentralized manner between the devices involved in the application described by the workflow. Workflows were represented and communicated as compositional HVs constructed using the primitives for representing sequences (Section 3.3 in [3]) and directed acyclic graphs (Section 3.5.2 in [3]). In [47], the approach was implemented in a particular workflow system – Node-RED. In [48], the approach was extended further to take into account the level of trust associated with various elements when selecting services.

2.1.3 String processing

In [49], it was proposed to obtain HVs of a word using permutations (cyclic shifts) of HVs of its letters to associate a letter with its position in the word. The conjunction was used to bind together all the obtained letter-in-position HVs. HVs of words formed in such a way were then used to obtain $n$-grams of sequences of words using the same procedure. The obtained HVs were used to estimate the frequencies of various word sequences in texts in order to create a model of human text reader interests. Note that here the conjunction result is somewhat similar to all input HVs.

An interesting property of sequence representation using permutations of its element HVs (Section 3.3 in [3]) is that the HV of a shifted sequence can be obtained by the permutation of the sequence HV as a whole [50]–[52]. This property was leveraged in [50] for searching the best alignment (shift) of two sequences, i.e., the alignment that provides the maximum number of coinciding symbols. This can be used, e.g., for identifying common substrings. Such a representation, however, does not preserve the similarity of symbols’ HVs in nearby positions, which would be useful for, e.g., spell checking. This can be addressed by, e.g., extending the permutation-based representations as in [53], where the resultant compositional HVs were evaluated on a permutated text, which was successfully reconstructed. An approach to transforming sequences into sparse HVs from [54], which preserves the similarity of symbols at nearby positions and is shift-equivariant, was applied to the spellchecking task.

An algorithm for searching a query string in the base string was proposed in [55] and modified in [6]. It is based on the idea of representing finite-state automata in an HV (see Section 2.1.1.1). The algorithm represented the base string as a non-deterministic finite-state automaton [56]. The symbols of the base string corresponded to the transitions between the states of the automaton. The automaton in turn was represented as a compositional HV. The automaton was initialized as a superposition of all atomic HVs corresponding to the states. The query substring was presented to the automaton symbol by symbol. If after the presentation of the whole substring the automaton appeared in one of the valid states, this indicated the presence of the query substring in the base string.

In [57], the MAP model was used for DNA string matching. The key idea was as in [58]: the base DNA string was represented by one or several HVs containing a superposition of all $n$-grams of predefined size(s). HVs of $n$-grams were formed by multiplicative binding of appropriately permuted HVs of their symbols (Section 3.3.4 in [3]). A query string was considered present in the base DNA string if the similarity between its HV and the compositional HV(s) of the base DNA string was higher than the predefined threshold. The threshold value determined the balance between true and false positives, similar to Bloom filters (see Section 3.1.2 in [3]). The approach was evaluated on two databases of DNA strings: escherichia coli and human chromosome 14. The main promise of the approach in [57] is the possibility to accelerate string matching with application-specific integrated circuits due to the simplicity and parallelizability of the HDC/VSA operations.

2.1.4 Factorization

The resonator networks [59], [60] were proposed as a way to solve factorization problems, which often emerge during the recovery procedure, within HDC/VSA (Section 2.2.5 in [3]). It is expected that the resonator networks will be useful for solving various factorization problems, but this requires formulating the problem for the HDC/VSA domain. An initial attempt to decompose synthetic scenes was demonstrated in [61]. A more recent formulation for the integer factorization problem was presented in [62]. The transformation of numbers into HVs was based on the fractional power encoding [63], [64] (Section 3.2.1 in [3]) combined with log transformation, and the resonator network was used to solve the integer factorization problem. The approach was evaluated on the factorization of semiprimes.

2.2 Similarity estimation with HVs

Transformations of original data into HVs allow constructing HVs in various application areas in a manner that preserves similarity relevant for a particular application. This provides a tool to use HDC/VSA for “similarity-based reasoning” that includes, e.g., similarity search and classification in its simplest form as well as the much more advanced analogical reasoning considered in Section 3.1.2. Due to the abundance of studies on classification tasks we devote another separate Section 2.3 to it. In this section, we primarily focus on “context” HVs, since for a long time these were the most influential application of HDC/VSA. We also present some existing efforts in similarity search.
Tables 2: Experiments with context vectors.

| Ref. | Task | Dataset | Method | Baseline(s) |
|------|------|---------|--------|-------------|
| [72] | Synonymy test | TOEFL | RI with word-document matrix | BEAGLE |
| [73] | Synonymy test | TOEFL, ESL | RI with word-word matrix | BEAGLE |
| [75] | Semantic similarity of word pairs | TOEFL, ESL | RI with word-word matrix | BEAGLE |
| [77] | Water drop in woman’s vagina simulation | Own data | RI with word-document matrix | BEAGLE |
| [79] | Synonymy test | MEDLARS; Criminal, time Magazine | RI with word-document matrix | [Generalized] Vector Space Model |
| [80] | Retrieval of cancer therapies | A set of predictions extracted from MEDLINE | FreHRR-based PSI | BSC-based PSI |
| [81] | Identification of agents active against cancer cells | SemMedDB | PSI | Reflective RI |
| [82] | Semantic similarity of word pairs | From [77] & [83] & [85] | BEAGLE: RI with permutations | BEAGLE |
| [84] | Taxonomic organization | From [74] & [77] | TIS | BEAGLE, LSA |
| [85] | Meaning shrinkage in contexts | From [74] | BEAGLE, TIS |
| [86] | Synonymy test | TOEFL | BEAGLE with experimental optimization | BEAGLE |
| [87] | Prediction of side-effects of drug combinations | From [91] | ESP | graph convolutional ANN |
| [88] | Semantic similarity of word pairs | From [77] & [83] & [85] | BEAGLE: RI with permutations | BEAGLE |
| [89] | Acculturation similarity among alphabetic characters | Own data | RI with permutations | BEAGLE |
| [90] | Influence of ethnic origin on lexical behavior | English Lexicon Project, British Lexicon Project, etc. | BEAGLE | N/A |
| [91] | Acculturation similarity among alphabetic characters | Own data | RI with characters | Word2vec [90] & EARP [96] |
| [92] | Changes in verbal fluency | Canadian Longitudinal Study of Aging | BEAGLE | N/A |

2.2.1 Word embeddings: context HVs for words and texts

The key idea for constructing context vectors is usually referred to as the “distributional semantics hypothesis” [65], suggesting that linguistic items with similar distributions have similar meanings. The distributions are calculated as frequencies of item occurrence in particular contexts by using a document corpus. For items-words the contexts could be, e.g., documents, paragraphs, sentences, or sequences of words close to a focus word. For example, the generalized vector space model [66] used documents as contexts for words in information retrieval. In computational linguistics and machine learning, vectors are also known as embeddings.

In principle, context vectors can be obtained in any domain where objects and contexts could be defined. Below, we will only focus on the context vector methods which are commonly attributed to HDC/VSA. They usually transform the frequency distributions into HVs of particular formats, which we will call context HVs.

Historically, the first proposal to form context HVs for words was that of Gallant, e.g., [67]–[70]. These studies, however, did not become known widely, but see [71]. In fact, two most influential HDC/VSA-based methods for context HVs are Random Indexing (RI) [72], [73] and Bound Encoding of the Aggregate Language Environment (BEAGLE) [74].

2.2.1.1 Random Indexing: The RI method [72], [73] was originally proposed in [72] as a simple alternative to Latent Semantic Analysis (LSA) [98]. Instead of the expensive Singular Value Decomposition (SVD) used in LSA for the dimensionality reduction of a word-document matrix, RI uses the multiplication by a random matrix, thereby performing random projection (Section 3.2.3 in [3]). The Random Projection (RP) matrix was ternary \((-1, 0, 1)) and sparse. Each row of the RP matrix is seen as a “random index” (hence RI) assigned to each context (in that case, the document). In implementation, the frequency matrix was not formed explicitly, and the resultant context HVs were formed by scanning the document corpus and adding the document’s random index vector to the context HV of each word in the document. The sign function can be used to obtain binary context HVs. The similarity between unnormalized context HVs was measured by \(\text{sim}_{\text{con}}\). The synonymy part of TOEFL was used as the benchmark to demonstrate that a performance comparable to LSA could be achieved, but at lower computational costs due to the lack of SVD. In [99], it was proposed to speed up LSA by using RP before SVD as a preprocessing step. In [100], similar to [68], [87], RI was modified to use a narrow context window consisting of only few adjacent words on each side of the focus word.

In [79], permutations were used to represent the order information within the context window. Further extensions of RI included generalization to multidimensional arrays (N-way RI) [101] and inclusion of extra-linguistic features [102]. The RI was also extended to the case of training corpora that include information about relations between words. This model is called Prediction-based Semantic Indexing (PSI) [103]–[105]. PSI has been mainly used in biomedical informatics for literature-based discovery, such as identification of links between pharmaceutical substances and diseases they treat (see [103] for more details). Later, PSI was extended to the Embedding of Semantic Predications (ESP) model that incorporates some aspects of “neural” word embeddings from [95] and similarity preserving HVs (Section 3.2 in [3]) for representing time periods [106].

It is worth mentioning that the optimization-based method [107] for obtaining similarity preserving HVs from co-occurrence statistics can be contrasted to RI. RI also uses co-occurrence statistics, but implicitly (i.e., without constructing the co-occurrence matrix). The difference, however, is that the RI is optimization-free and it forms HVs through a single pass via the training data. Thus, it can be executed online in an incremental manner, while the optimization required by [107] calls for iterative processing, which might be more suitable for offline operations.

2.2.1.2 Bound Encoding of the Aggregate Language Environment: There is an alternative method for constructing context HVs with HDC/VSA, known as BEAGLE [74]. It was proposed independently of RI and used HRR to form the HVs of n-grams of words within a sentence, providing a representation of the order and context HVs of words. Words were initially represented with random atomic HVs as in HRR. The context HV of a word was obtained from two “parts”.

random projection (Section 3.2.3 in [3]). The Random Projection (RP) matrix was ternary \((-1, 0, 1)) and sparse. Each row of the RP matrix is seen as a “random index” (hence RI) assigned to each context (in that case, the document). In implementation, the frequency matrix was not formed explicitly, and the resultant context HVs were formed by scanning the document corpus and adding the document’s random index vector to the context HV of each word in the document. The sign function can be used to obtain binary context HVs. The similarity between unnormalized context HVs was measured by \(\text{sim}_{\text{con}}\). The synonymy part of TOEFL was used as the benchmark to demonstrate that a performance comparable to LSA could be achieved, but at lower computational costs due to the lack of SVD. In [99], it was proposed to speed up LSA by using RP before SVD as a preprocessing step. In [100], similar to [68], [87], RI was modified to use a narrow context window consisting of only few adjacent words on each side of the focus word.

In [79], permutations were used to represent the order information within the context window. Further extensions of RI included generalization to multidimensional arrays (N-way RI) [101] and inclusion of extra-linguistic features [102]. The RI was also extended to the case of training corpora that include information about relations between words. This model is called Prediction-based Semantic Indexing (PSI) [103]–[105]. PSI has been mainly used in biomedical informatics for literature-based discovery, such as identification of links between pharmaceutical substances and diseases they treat (see [103] for more details). Later, PSI was extended to the Embedding of Semantic Predications (ESP) model that incorporates some aspects of “neural” word embeddings from [95] and similarity preserving HVs (Section 3.2 in [3]) for representing time periods [106].

It is worth mentioning that the optimization-based method [107] for obtaining similarity preserving HVs from co-occurrence statistics can be contrasted to RI. RI also uses co-occurrence statistics, but implicitly (i.e., without constructing the co-occurrence matrix). The difference, however, is that the RI is optimization-free and it forms HVs through a single pass via the training data. Thus, it can be executed online in an incremental manner, while the optimization required by [107] calls for iterative processing, which might be more suitable for offline operations.

2.2.1.2 Bound Encoding of the Aggregate Language Environment: There is an alternative method for constructing context HVs with HDC/VSA, known as BEAGLE [74]. It was proposed independently of RI and used HRR to form the HVs of n-grams of words within a sentence, providing a representation of the order and context HVs of words. Words were initially represented with random atomic HVs as in HRR. The context HV of a word was obtained from two “parts”.
The first part included summing the atomic HVs of the words in the sentence other than the focus word, for all corpus sentences. The second part was contributed by the word order HVs, which were formed as the superposition of the word $n$-gram HVs (with $n$ between 2 and 7). The $n$-gram HVs were formed with the circular convolution-based binding of the special atomic HV in place of the focus word and the atomic HVs of the other word(s) in the $n$-gram. The word order in an $n$-gram was represented recursively, first by binding the HVs of the left and the right word permuted differently, and then by binding the resultant HV with the next right word, again using permutations for the “left” and “right” relative positions. The total context HV was the superposition of the HVs for the two parts. The similarity measure was $\text{sim}_{\text{cos}}$.

Later, [108] presented a modified version of BEAGLE using random permutations. The authors found that their model was both more scalable to large corpora and gave better fits to semantic similarity than the circular convolution-based representation. A comprehensive treatment of methods for constructing context HVs of phrases and sentences as opposed to individual words was presented in [109]. In a similar spirit, in [110] the HRR model was used to construct compositional HVs that were able to discover language regularities resembling syntax and semantics. The comparison of different sentence embeddings (including BEAGLE) in terms of their ability to represent syntactic constructions was provided in [111]. It has been demonstrated that context HVs formed by BEAGLE can account for a variety of semantic category effects such as typicality, priming, and acquisition of semantic and lexical categories. The effect of using different lexical materials to form context HVs with BEAGLE was demonstrated in [89], while the use of negative information for context HVs was assessed in [112]. In [113], BEAGLE was extended to a Hierarchical Holographic Model by augmenting it with additional levels corresponding to higher-order associations, e.g., part-of-speech and syntactic relations. The Hierarchical Holographic Model was used further in [114] to investigate what grammatical information is available in context HVs produced by the model.

Due to high similarity between BEAGLE and RI, both methods were compared against each other in [82], [108]. It was shown that both methods demonstrate similar results on a set of semantic tasks using a Wikipedia corpus for training. The main difference was that RI is much faster as it does not use the circular convolution operation. For placing the methods in the general context of word embeddings, please refer to [115].

The work [116] aimed to address the representation of similarity, rather than relatedness represented in context HVs by BEAGLE and RI. To do so, for each word the authors represented its most relevant semantic features taken from a knowledge base ConceptNet. The context HV of a word was formed using BSC as a superposition of its semantic feature HVs formed by role-filler bindings. The results from measuring the semantic similarity between pairs of concepts were presented using the SimLex-999 dataset.

Table 2 provides a non-comprehensive summary of the studies that applied the BEAGLE and RI methods to various linguistic tasks. The interested readers are referred to [117], which is a survey describing the applications of RI, PSI, and related methods, including the biomedical domain. The range of applications described in [117] covers word-sense disambiguation, bilingual information extraction, visualization of relations between terms, and document retrieval. It is worth noting that there is a software package called “Semantic vectors” [118], [119], [120], that implements many of the methods mentioned above and provides the main building blocks for designing further modifications of the methods.

### 2.2.2 Similarity estimation of biomedical signals

In [121], [122], BSC was applied to biomedical signals: heart rate and respiration. The need for comparing these signals emerged in the scope of a deep breathing test for assessing autonomic function. HDC/VSA was used to analyze cardiorespiratory synchronization by comparing the similarity between heart rate and respiration using feature-based analysis. Feature vectors were extracted from the signals and transformed into HVs by using role-filler bindings (Section 3.1.3 in [3]) and representations of scalars (Section 3.2 in [3]). These HVs were in turn classified into different degrees of cardiorespiratory synchronization/desynchronization. The signals were obtained from the healthy adult controls, patients with cardiac autonomic neuropathy, and patients with myocardial infarction. It was shown that, as expected, the similarity between different HVs were lower for patients with the cardiac autonomic neuropathy and myocardial infarction patients, than for the healthy controls.

Another application of BSC was the identification of the ictogenic (i.e., seizure generating) brain regions from intracranial electroencephalography (iEEG) signals [123]. The algorithm first transformed iEEG time series from each electrode into a sequence of symbolic local binary pattern codes, from which a binary HV was obtained for each brain state (e.g., ictal or interictal). It then identified the ictogenic brain regions by measuring the relative distances between the learned HVs from different groups of electrodes. Such the identification was done by one-way ANOVA tests at two levels of spatial resolution, the cerebral hemispheres and lobes.

### 2.2.3 Similarity estimation of images

In [124], the HDC/VSA 2D image representations (Section 3.4 in [3]) were applied for an aggregation of local descriptors extracted from images. Local image descriptors were real-valued vectors whose dimensionality was controlled by RP (Section 3.2.3 in [3]). To represent a position inside an interval, the authors concatenated parts of two basis HVs and used several intervals, as in [125], [126], but using MAP. Position HVs for $x$ and $y$ were bound to represent $(x, y)$, see Section 3.4 in [3]. Subsequently, projected local image descriptors were bound with their position HVs, using component-wise multiplication, and the bound HVs were superimposed to represent the whole image. The image HVs obtained with different algorithms for extracting the descriptors could also be aggregated using the superposition operation.
When compared to the standard aggregation methods in (mobile robotics) place recognition experiments, HVs of the aggregated descriptors exhibited an average performance better than alternative methods (except the exhaustive pair-wise comparison). A very similar concept was demonstrated in [127] using an image classification task, see also Table 15. One of the proposed ways of forming image HV used the superposition of three binary HVs obtained from three different hashing neural networks. The HVs representing the aggregated descriptors provided a higher classification accuracy. Finally, similarity-preserving shift-equivalent representation of images in HVs using permutations was proposed in [128].

2.3 Classification

Applying HDC/VSA to classification tasks is currently one of the most common application areas of HDC/VSA. This is due to the fact that similarity-based and other vector-based classifiers are widespread and machine learning research is on the rise in general. The recent survey [11] of classification with HDC/VSA was primarily devoted to the transformation of input data into HVs. Instead, here we focus first on the types of input data (in the 2nd level headings), and then on the domains where HDC/VSA have been applied (in the 3rd level headings). Moreover, we cover some of the studies not presented in [11]. The studies are summarized in the form of tables, where each table specifies a reference, type of task, dataset used, format of HVs, operations to form HVs from data, the type of classifier, and baselines for comparison. For the sake of consistency, in this section we use a table even if there is only a single work in a particular domain.

### Table 3: HDC/VSA studies classifying languages

| Ref. | Task | Dataset | HV format | Primitives used in data transformation | Classifier | Baseline(s) |
|------|------|---------|-----------|----------------------------------------|------------|-------------|
| [58] | Language identification | Wortschatz Corpora & Europarl Corpus | binarized ANN | binding; permutation; superposition | centroids | vector centroids |
| [90] | Language identification | Wortschatz Corpora & Europarl Corpus | dense binary | binding; permutation; superposition | binarized centroids | vector centroids |
| [106] | Language identification | Wortschatz Corpora & Europarl Corpus | sparse binary | binding; permutation; superposition | self-organizing map | approach from [58] |
| [117] | Language identification | Wortschatz Corpora & Europarl Corpus | sparse binary | binding; permutation; superposition | available binarized centroids | context |

### Table 4: HDC/VSA studies classifying texts

| Ref. | Task | Dataset | HV format | Primitives used in data transformation | Classifier | Baseline(s) |
|------|------|---------|-----------|----------------------------------------|------------|-------------|
| [58] | Text classification | 20 Newsgroups | text-based | binding; superposition | SVM with context and part of speech HVs | SVM with context HVs |
| [91] | News classification | Reuters newswire | dense binary | binding; permutation; superposition | centroids | Reuters ANN |
| [106] | Text classification | Chorafis, Ask Omaha, Yale Applications | dense binary | binding; permutation; superposition | binarized ANN | machine learning algorithms |
| [108] | Text classification | Chorafis, Ask Omaha, Yale Applications | sparse binary | binding; permutation; superposition | machine learning algorithms | machine learning algorithms |
| [107] | Text classification | Reuters newswire | dense binary | binding; permutation; superposition | binarized centroids | ANN; SVM; ANN; Random Forest |

### Table 5: HDC/VSA studies classifying acoustic signals

| Ref. | Task | Dataset | HV format | Primitives used in data transformation | Classifier | Baseline(s) |
|------|------|---------|-----------|----------------------------------------|------------|-------------|
| [133] | Language identification | CMU MOSEI | integer-valued | binding; permutation; superposition | large margin perceptron, centroid associated model | N/A |
| [134] | Language identification | CMU MOSEI | integer-valued | binding; permutation; superposition | large margin perceptron, centroid associated model | N/A |
| [135] | Language identification | CMU MOSEI | integer-valued | binding; permutation; superposition | large margin perceptron, centroid associated model | N/A |
| [136] | Language identification | CMU MOSEI | integer-valued | binding; permutation; superposition | large margin perceptron, centroid associated model | N/A |
| [137] | Language identification | CMU MOSEI | integer-valued | binding; permutation; superposition | large margin perceptron, centroid associated model | N/A |
| [140] | Language identification | CMU MOSEI | integer-valued | binding; permutation; superposition | linear discriminant analysis | N/A |
| [141] | Language identification | CMU MOSEI | integer-valued | binding; permutation; superposition | linear discriminant analysis | N/A |
| [142] | Language identification | CMU MOSEI | integer-valued | binding; permutation; superposition | linear discriminant analysis | N/A |
| [143] | Language identification | CMU MOSEI | integer-valued | binding; permutation; superposition | linear discriminant analysis | N/A |
| [144] | Language identification | CMU MOSEI | integer-valued | binding; permutation; superposition | linear discriminant analysis | N/A |
| [145] | Language identification | CMU MOSEI | integer-valued | binding; permutation; superposition | linear discriminant analysis | N/A |
| [146] | Language identification | CMU MOSEI | integer-valued | binding; permutation; superposition | linear discriminant analysis | N/A |
| [147] | Language identification | CMU MOSEI | integer-valued | binding; permutation; superposition | linear discriminant analysis | N/A |
| [148] | Language identification | CMU MOSEI | integer-valued | binding; permutation; superposition | large margin perceptron, centroid associated model | N/A |
| [149] | Language identification | CMU MOSEI | integer-valued | binding; permutation; superposition | large margin perceptron, centroid associated model | N/A |
| [150] | Language identification | CMU MOSEI | integer-valued | binding; permutation; superposition | large margin perceptron, centroid associated model | N/A |
| [151] | Language identification | CMU MOSEI | integer-valued | binding; permutation; superposition | large margin perceptron, centroid associated model | N/A |
| [152] | Language identification | CMU MOSEI | integer-valued | binding; permutation; superposition | large margin perceptron, centroid associated model | N/A |
| [153] | Language identification | CMU MOSEI | integer-valued | binding; permutation; superposition | large margin perceptron, centroid associated model | N/A |
| [154] | Language identification | CMU MOSEI | integer-valued | binding; permutation; superposition | large margin perceptron, centroid associated model | N/A |
| [155] | Language identification | CMU MOSEI | integer-valued | binding; permutation; superposition | large margin perceptron, centroid associated model | N/A |
| [156] | Language identification | CMU MOSEI | integer-valued | binding; permutation; superposition | large margin perceptron, centroid associated model | N/A |
| [157] | Language identification | CMU MOSEI | integer-valued | binding; permutation; superposition | large margin perceptron, centroid associated model | N/A |

### Table 2.3.1 Classification based on feature vectors

2.3.1.1 Language identification with the vector of $n$-gram statistics of letters: In [58], it was shown how to form a compositional HV corresponding to $n$-gram statistics (see Section 3.3.4 in [3]). The work also introduced a task of identifying a language amongst 21 European languages. Since then, the task was used in several studies summarized in Table 3.

2.3.1.2 Classification of texts: Table 4 summarizes the efforts of using HDC/VSA for text classification. The works in this domain dealt with different tasks such as text categorization, news identification, and intent classification. Most of the works [133], [136], [137] used HVs as a way of representing data for conventional machine learning classification algorithms.

2.3.1.3 Classification of feature vectors extracted from acoustic signals: Classification of various acoustic signals using HVs is provided in Table 5. Tasks were mainly related to speech recognition, e.g., recognition of spoken letters or words.

2.3.1.4 Fault classification: Studies on applying HDC/VSA to fault classification are limited. We are only aware of two such use-cases (summarized in Table 6) applied to the problems of anomaly detection in a power plant and ball bearings. An earlier work on micro machine-tool acoustic diagnostics was presented in [159].

2. For the sake of generality, it was decided to avoid going to in-depth details of data transformations, so tables only specify the HDC/VSA operations used to construct HVs from data.
TABLE 6: HDC/VSA studies classifying faults

| Ref   | Task                          | Dataset                  | HV format              | Primitives used in data transformation | Classifier | Baseline(s) |
|-------|-------------------------------|--------------------------|------------------------|----------------------------------------|------------|-------------|
| [159] | Acoustic diagnostics of micro-machine tools | Own data                | sparse binary          |                                       | SVM        | N/A         |
| [160] | Noise isolation               | Face data                | dense binary           | binding, superposition                | SVM        | N/A         |
| [161] | Rail bearing anomaly detection | BSS Bearing Dataset     | dense binary           | binding, permutation, superposition   | SVM        | N/A         |

TABLE 7: HDC/VSA studies classifying automotive data

| Ref   | Task                          | Dataset                  | HV format              | Primitives used in data transformation | Classifier | Baseline(s) |
|-------|-------------------------------|--------------------------|------------------------|----------------------------------------|------------|-------------|
| [162] | Identification of vehicle type | From [166]               | dense binary           | binding, superposition                | SVM        | N/A         |
| [163] | Prediction of vehicle's trajectory | Own data, NGSIM US-101 | real-valued            | binding, superposition                | LSTM       | LSTM without HVs |
| [164] | Identification of driving style | UCI HAR                  | complex-valued         | binding, superposition, fractional point encoding | ANN; SNN; SVM; ANN | LSTM without HVs |
| [165] | Detection of automotive sensor attacks | AEGIS Big Data Project | bipolar                | binding, superposition                | similarity between original and reconstructed samples | N/A |

TABLE 8: HDC/VSA studies classifying behavioral signals

| Ref   | Task                          | Dataset                  | HV format              | Primitives used in data transformation | Classifier | Baseline(s) |
|-------|-------------------------------|--------------------------|------------------------|----------------------------------------|------------|-------------|
| [172] | Activity recognition          | Face data                | bipolar                | binding, permutation, superposition   | SVM        | N/A         |
| [173] | Activity recognition          | UCI HAR                  | dense binary           | binding, superposition                | SVM        | N/A         |
| [174] | Activity recognition          | UCI HAR                  | dense binary           | binding, weighted superposition        | SVM        | N/A         |
| [175] | Detection of Parkinson's Disease | Parkinson's Disease digital biomarker | bipolar                | binding, superposition                | SVM        | N/A         |
| [176] | Activity recognition          | UCI HAR                  | bipolar                | binding, superposition                | SVM        | N/A         |
| [177] | Activity recognition          | UCI HAR                  | N/A                    | trainable permutation matrix            | ANN; SVM; Adolescent | N/A         |
| [178] | Activity recognition          | UCI HAR                  | bipolar                | trainable permutation matrix            | ANN; SVM; Adolescent | N/A         |
| [179] | Activity recognition          | UCI HAR                  | bipolar                | binding, superposition                | SVM        | N/A         |
| [180] | Activity recognition          | UCI HAR                  | dense binary           | binding, superposition                | SVM        | N/A         |
| [181] | Activity recognition          | UCI HAR                  | dense binary           | binding, superposition, scalar multiplication | SVM | N/A         |
| [182] | Activity recognition          | UCI HAR                  | dense binary           | binding, permutation, superposition   | SVM        | N/A         |
| [183] | Activity recognition          | UCI HAR                  | dense binary           | binding, superposition, scalar multiplication | SVM | N/A         |
| [184] | Activity recognition          | UCI HAR                  | dense binary           | binding, superposition                | SVM        | N/A         |
| [185] | Activity recognition          | UCI HAR                  | dense binary           | binding, superposition                | SVM        | N/A         |
| [186] | Activity recognition          | UCI HAR                  | dense binary           | binding, superposition                | SVM        | N/A         |
| [187] | Activity recognition          | UCI HAR                  | dense binary           | binding, superposition                | SVM        | N/A         |
| [188] | Activity recognition          | UCI HAR                  | dense binary           | binding, superposition                | SVM        | N/A         |
| [189] | Activity recognition          | UCI HAR                  | dense binary           | binding, superposition                | SVM        | N/A         |

TABLE 9: HDC/VSA studies classifying EMG signals

| Ref   | Task                          | Dataset                  | HV format              | Primitives used in data transformation | Classifier | Baseline(s) |
|-------|-------------------------------|--------------------------|------------------------|----------------------------------------|------------|-------------|
| [190] | Hand gesture recognition      | From [156]               | dense binary           | binding, permutation, superposition    | SVM        | N/A         |
| [191] | Hand gesture recognition      | Own data                 | dense binary           | binding, permutation, superposition    | SVM        | N/A         |
| [192] | Hand gesture recognition      | Own data                 | dense binary           | binding, permutation, superposition    | SVM        | N/A         |
| [193] | Hand gesture recognition      | Own data                 | dense binary           | binding, permutation, superposition    | SVM        | N/A         |
| [194] | Adaptive hand gesture recognition | Own data              | dense binary           | binding, permutation, superposition    | SVM        | N/A         |
| [195] | Adaptive hand gesture recognition | Own data              | dense binary           | binding, permutation, superposition    | SVM        | N/A         |

2.3.1.5 Automotive data: Table 7 presents studies where HDC/VSA was used with automotive data, mainly in autonomous driving scenarios.

2.3.1.6 Behavioral signals: Studies that used behavioral signals are summarized in Table 8. One of the most common applications was activity recognition, but other tasks were considered as well (see the table).

2.3.1.7 Biomedical data: Currently explored applications of HDC/VSA on biomedical data can be categorized into five types of data: electromyography (EMG) signals, electroencephalography (EEG) signals, cardiocography (CTG) signals, DNA sequences, and surface enhanced laser desorption/ionization time-of-flight (SELDI-TOF) mass spectrometry. Most of the works so far have been conducted on EMG and EEG signals. In fact, there is a study [9], which provides an in-depth coverage of applying HDC/VSA to these modalities, so please refer to this article for a detailed overview of the area.

EMG signals. HDC/VSA was applied to EMG signals for the task of hand gesture recognition. This was done for several different transformations of data into HVs and on different datasets. Refer to Table 9 for the summary.

EEG and iEEG signals. EEG and iEEG signals were used with HDC/VSA for human-machine interfaces and epileptic seizure detection. These efforts are overviewed in Table 10. It is worth mentioning systematization efforts in [194], [195], which reported an assessment of several HDC/VSA models and transformations used for epileptic seizure detection. Another recent work [196] provides a tutorial on applying HDC/VSA for iEEG seizure detection.

CTG signals. So far, there is only one work [178] where CTG signals were used. It is summarized in Table 11.

3. It should be noted that works [169], [170] were, strictly speaking, solving the regression problems while [171], [172] were concerned with anomaly detection. These studies are listed in this section for the sake of covering the applications within the automotive data.
2.3 Classification of structured data

Classification of structured data can be tricky with conventional machine learning algorithms since local representations of structured data might not be convenient to work when using vector classifiers, especially when the data involve some sorts of hierarchies. HDC/VSA should be well suited for structured data since they allow representing various structures of structured data might not be convenient to work when using vector classifiers, especially when the data involve some sorts of hierarchies. HDC/VSA should be well suited for structured data since they allow representing various structures of structured data.

### Multi-modal signals

Studies involving multi-modal signals are summarized in Table 14.

2.3.2 Classification of images or their properties

Table 15 provides an overview of the efforts involving images. Since using raw pixels directly would rarely result in a good performance, HVs were produced either from features extracted from images or using HVs obtained from neural networks (see Section 3.4.3 in [3]), which took images as an input.

2.3.3 Classification of structured data

Classification of structured data can be tricky with conventional machine learning algorithms since local representations of structured data might not be convenient to work when using vector classifiers, especially when the data involve some sorts of hierarchies. HDC/VSA should be well suited for structured data since they allow representing various structures (including hierarchies) as HVs. To the best of our knowledge, however, the number of such studies is very limited (see Table 16). One such study in [229] used SBDR to predict the properties of chemical compounds and provided the state-of-the-art performance. A more recent example was demonstrated in [230], where 2D molecular structures were transformed into HVs that were used to construct classifiers for drug discovery problems. The approach outperformed the baseline methods on a collection of 30 tasks. Finally, in [231] it was proposed to classify graphs using HDC/VSA. A graph was represented as a superposition of HVs corresponding to vertices and edges. The proposed approach was evaluated on six graph classification datasets; when compared to the baseline approaches, it demonstrated comparable accuracy on four datasets and much shorter training time on all of them.

3 Cognitive computing and architectures

In this section, we overview the use of HDC/VSA in cognitive computing (Section 3.1) and cognitive architectures (Section 3.2). Note that, strictly speaking, cognitive computing as well as cognitive architectures can also be considered to be application areas but we decided to separate them into a distinct section due to the rather different nature of tasks being pursued.

DNA sequences. Table 12 presents two studies that used DNA sequences in classification tasks. SELDI-TOF mass spectrometry. Table 13 summarizes a study that used SELDI-TOF mass spectrometry for classifying sensitivity of glioma to chemotherapy using SELDI-TOF.
**3.1 Cognitive computing**

##### 3.1.1 Holistic transformations

3.1.1.1 Holistic transformations in database processing with HVs: In this section, we consider simple examples of HDC/VSA processing of a small database. This can be treated as analogical reasoning (though analogy researchers might disagree) in the form of query answering using simple analogs (we refer to them as records) without explicitly taking into account the constraints on analogical reasoning mentioned in the following sections.

The work [232] introduced the transformation of database records into HVs as an alternative to their symbolic representations. Each record is a set of role-filler bindings and is represented by HVs using transformations for role-filler bindings (Section 3.1.3 in [3]) and sets (Section 3.1.2 in [3]). We might be interested in querying the whole base for some property or relationship.

Several ways were proposed to use HDC/VSA operations for querying the records by HVs. An example query to the database operation but as an example demonstrating substitution transformations.

**TABLE 15: HDC/VSA studies classifying visual images and their properties**

| Ref. | Task | Dataset | HV format | Primitives used in data transformation | Classifier | Baseline(s) |
|------|------|---------|-----------|---------------------------------------|------------|-------------|
| [211] | Texture classification | Own data | sparse binary | binding, superposition | SVM; ANN | N/A |
| [212] | Micro-object shape recognition | Own data | sparse binary | binding, superposition | SVM; ANN | N/A |
| [213] | Stereopsis | INRIA-CLUE2014 | dense binary | binding, superposition | SVM; ANN | N/A |
| [214] | Biological gender classification | BML from FER | bipolar | binding, superposition | SVM; ANN | N/A |
| [215] | Visual place recognition | ONTID; Helmut; STATA; CHU; Visual Localization | complex-valued | binding, superposition | SVM; ANN | N/A |
| [216] | Egomotion estimation | MVVEC | dense binary | binding, superposition | SVM; ANN | N/A |
| [217] | Egomotion estimation | MVVEC | dense binary | random sparse projection, CDT | SVM, ANN | N/A |
| [218] | Detection of pneumonia | NSRCC-UV; UC [101] & [102] | bipolar | binding, superposition | SVM; ANN | N/A |
| [219] | Object classification | CIFAR-10, Artiﬁcial Face | dense binary | position binding, superposition | SVM; ANN | N/A |
| [220] | Object recognition | Fashion-MNIST | dense binary | permutation, binding, superposition | SVM; ANN | N/A |
| [221] | Character recognition | Fashion-MNIST | dense binary | permutation, binding, superposition | SVM; ANN | N/A |
| [222] | Character recognition | MNIST | sparse binary | LIRA | large margin perception | conventional classifiers |
| [223] | Character recognition | MNIST | sparse binary | BLUHV, permutation, superposition | large margin perception | conventional classifiers |
| [224] | Character recognition | MNIST | sparse binary | LIRA, HV, superposition | large margin perception | feature selection |
| [225] | Character recognition | MNIST | sparse binary | N/A | classification error | random forests |
| [226] | Character recognition | MNIST | sparse binary | binding, superposition | non-quantized centroids | MNIST, KNN, Adaboost |
| [227] | Character recognition | MNIST | sparse binary | binding, superposition | quantized centroids | SVM, KNN, Adaboost |
| [228] | Character recognition | MNIST | sparse binary | binding, superposition | non-quantized centroids | SVM, KNN, Adaboost |
| [229] | Character recognition | MNIST | sparse binary | binding, superposition | non-quantized centroids | SVM, KNN, Adaboost |
| [230] | Few-shot character recognition | Omniglot | dense binary | 5-Layer CNNs | weighted ANN | various CNNs |
| [231] | Few-shot character recognition | Omniglot | dense binary | 5-Layer CNNs | linear product-based associative memory | approach from [229] |
| [232] | Few-shot continual training for image classification | CIFAR-10, SVHN, SVHN, Omniglot | dense binary | Pre-trained CNN and a retrainable linear layer | deep convolutional, loss-optimized modular networks | various deep CNNs |
| [233] | Few-shot continual training for image classification | CIFAR-10, SVHN, SVHN, Omniglot | dense binary | Pre-trained CNN and a retrainable linear layer | deep convolutional, loss-optimized modular networks | various deep CNNs |
| [234] | Character recognition | ImageNet | sparse binary | LIRA, BLUHV, permutation, superposition | large margin perception | conventional classifiers |
| [235] | Face recognition | CAFACE | dense binary | binding, superposition | multiple linearized centroids | SVM |
| [236] | Face recognition | CAFACE | dense binary | binding, superposition | quantized centroids | SVM, KNN, Adaboost |
| [237] | Face recognition | CAFACE | dense binary | binding, superposition | quantized centroids | SVM, KNN, Adaboost |
| [238] | Face recognition | CAFACE | dense binary | binding, superposition | quantized centroids | SVM, KNN, Adaboost |

**TABLE 16: HDC/VSA studies classifying structured data**

| Ref. | Task | Dataset | HV format | Primitives used in data transformation | Classifier | Baseline(s) |
|------|------|---------|-----------|---------------------------------------|------------|-------------|
| [220] | Prediction of chemical compound properties | INTAS-9-397 | sparse binary | binding, superposition | SVM | DISCOVERY, ANALOGY |

4. This solution relies on knowing that there is only one Lee & female record in the database, so it should not be considered as a sensible database operation but as an example demonstrating substitution transformations.
above, age ⊗ LF6 after the clean-up procedure returns 66. Note that such a transformation is intended for HDC/VSA models where the binding operation is self-inverse (e.g., BSC).

Case 3. Only 33 and PF3 are known, and the task is to find an analog of 33 in LF6, that is, 66. A trivial solution would be to get age as the result of the clean-up procedure for 33 ⊗ PF3, and then get 33 by age ⊗ PF3 and the clean-up procedure. But there is a more interesting way, which can be considered as an analogical solution. Solution 3. One step solution is LF6 ⊗ (33 ⊗ PF3). This exemplifies a possibility of processing without the intermediate use of a clean-up procedure.

In some HDC/VSA models (e.g., BSC), however, the answer for this solution will be ambiguous, being equally similar to 33 and 66. This is due to the self-inverse property of the binding operation in BSC. Note that both LF6 and PF3 include gender ⊗ female as the part of their records. Unbinding 33 with PF3 creates 33 ⊗ gender ⊗ female (since ⊗ = ◦) among other bindings. When unbinding the result with LF6, the HV gender ⊗ female will cancel out, thus releasing 33, which will interfere with the correct answer, which is 66. This effect would not appear if the records would had different fillers in their role-filler bindings. For example, if instead of LF6 we consider LM6, then 33 ⊗ PF3 ⊗ LM6 produces the correct answer.

In models with self-inverse binding, the result of PF3 ⊗ LM6 can be seen as an interpretation of PF3 in terms of LM6 or vice versa, because the result of this operation is Pat ⊗ Lee + male ⊗ female + 33 ⊗ 66 + noise (since ⊗ = ◦). This allows answering queries of the form “which filler in PF3 plays the same role as (something) in LM6?” by unbinding PF3 ⊗ LM6 with the required filler HV, resulting in the noisy answer HV.

Note that Solution 1 resembles the standard processing of records in databases. We first identify the record and then check the value of the role of interest. Solutions 2b and 3 are examples of a different type of computing sometimes called “holistic mapping” [232] or “transformation without decomposition” [233]. We call it “holistic transformation” in order not to confuse it with the transformation of input data into HVs or with mapping in analogical reasoning. This holistic transformation of HVs is commonly illustrated by an example well-known under the name “Dollar of Mexico” [234]–[236].

In essence, the “Dollar of Mexico” show-case solves some simple “proportional analogies” of the form A:B :: C:D (i.e., United States/Mexico :: Dollar?). These analogies are also known to be solvable by addition and subtraction of the “neural” word embeddings of the corresponding concepts [95], [237]. Following a similar approach, the authors in [238] proposed to improve the results by training shallow neural networks using a dependency path of relations between terms in sentences.

It should be noted that there is no direct analog to this kind of processing by holistic transformation (using geometric properties of the representational space) in the conventional symbol manipulation [233]. The holistic transformation of HVs can be seen as a parallel alternative to the conventional sequential search.

3.1.1.2 Learning holistic transformations from examples: Learning systematic transformations from examples was investigated in [239], [240] for HRR. Previously, this capability was shown in [233] only for manually constructing the so-called transformation HV. In [239], [240], the transformation HV was obtained from several training pairs of HVs. One of the proposed approaches to obtaining the transformation HV was to use the gradient descent by iterating through all examples until the optimization converged. The experiments demonstrated that the learned transformation HVs were able to generalize to previously unseen compositional structures with novel elements. A high level of systematicity was indicated by the ability of transformation HVs to generalize to novel elements in structures of a complexity higher than the structures provided as training examples. The capability of BSC to learn holistic transformations was also presented in [235], [241]. However, the disadvantage of such holistic transformations is their bidirectionality, which is due to the fact that the unbinding operation in BSC is equivalent to the binding operation. This complication can be resolved by using either the permutation or an additional associative memory as the binding operation as proposed in [242].

The holistic transformation of the kind considered above was used to associate, e.g., sensory and motor/action data via binding HVs. For example, in [243], [244], BSC was applied to form a scene representation in the experiments with honey bees. It was shown to mimic learning in honey bees using the transformation as in [235], [241]. In [245], an HDC/VSA-based approach for learning behaviors, based on observing and associating sensory and motor/action data represented by HVs, was proposed. A potential shortcoming of the approaches to learning holistic transformations presented above is that the objects/relations are assumed to be dissimilar to each other. The learning might not work as expected given that there is some similarity structure between the objects/relations used as training examples. This direction deserves further investigation.
3.1.2 Analogical reasoning

We begin with a brief introduction to analogical reasoning by summarizing basic analogical processes as well as some of their properties. Two final sections discuss applications of HDC/VSA to modeling analogical retrieval and mapping.

3.1.2.1 Basics of analogical reasoning modeling: Modeling analogical reasoning in humans is an important frontier for Artificial Intelligence. Because it allows analogical problem solving as well as even more universal cognitive processes taking the problem structure into account and applying knowledge acquired in different domains. Analogical reasoning theories [246]–[250] usually consider and model the following basic processes: description; retrieval (also known as access or search); mapping; and inference.

Any model for analogical reasoning usually works with analogical episodes (or simply analogs). The description process concerns the representation of analogical episodes. The analogical episodes are usually modeled as systems of hierarchical relations (predicates), consisting of elements in the form of entities (objects) and relations of various hierarchical levels. Entities belong to some subject domain (e.g., the sun, planet, etc.) and are described by attributes (features, properties), which in essence are relations with a single argument (e.g., mass, temperature, etc.). Relations (e.g., attract, more, cause, etc.) define relations between the elements of analogical episodes. Arguments of relations may be objects, attributes, and other relations. It is assumed that a collection of base (source) analogical episodes is stored in a memory.

The retrieval process searches the memory (in models, the base of analogical episodes) to find the closest analog(s) for the given target (input, query) analog. The similarity between episodes is used as a criterion for the search. Once the base analogical episode(s) closest to the target is identified, the mapping process finds the correspondences between the elements of two analogous episodes: the target and the base ones. The inference process concerns the transfer of the knowledge from the base analogous 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 target analogical episode. Because the analogical reasoning is not a deductive mechanism, the candidate inferences are only hypotheses and must be evaluated and checked (see, e.g., [248] and references therein).

Cognitive science has identified quite a few properties demonstrated by subjects when performing analogical reasoning. Two types of similarity that influence processing of analogical episodes are distinguished. Structural similarity (which should not be confused with the structural similarity in HDC/VSA) reflects how the elements of analogs are arranged with respect to each other, that is, in terms of the relations between the elements [249]–[251]. Analogs are also matched by the “surface” or “superficial” similarity [246], [252] based on common analogs’ elements or a broader “semantic” similarity [251], [253], [254], based on, e.g., joint membership in a taxonomic category or on similarity of characteristic feature vectors. Experiments based on human assessment of similarities and analogies confirmed that both surface (semantic) and structural similarity are necessary for sound retrieval [252]. The structural similarity in the retrieval process is considered less important than that in the mapping process, however, the models of retrieval that take into account only the surface similarity are considered inadequate. These properties are expected to be demonstrated by the computational models of analogy [255]. In the following sections we discuss applications of HDC/VSA for analogical reasoning.

3.1.2.2 Analogical retrieval: It is known that humans retrieve some types of analogs more readily than others. Psychologists identified different types of similarities and ordered them according to the ease of retrieval [246], [252], [256], [257]. The similarity types are summarized in Table 18, relative to the base analogical episode. Simple examples of animal stories (adapted by Plate from [254]) with those similarity types are also presented. All analogical episodes have the same animal episodes should be taken into account in their similarity estimation. This requires alignment and

| Similarity type | Common low-order relations | Common high-order relations | Common object attributes | Examples using animal episodes |
|-----------------|---------------------------|----------------------------|-------------------------|-------------------------------|
| Literal Similarity (LS) | ✓ | ✓ | ✓ | bite(Spot, Jane), cause(bite, flee) |
| Surface Features (SF) | ✓ | ✗ | ✓ | cause(bite(Fido, John), cause(flee(Fido, John)) |
| True Analogy (AN) | ✓ | ✓ | ✗ | cause(bite(Felix, Mort), cause(flee(Mort), human(John)) |
| First-order Only Relations (FOR) | ✓ | ✗ | ✓ | cause(bite(Felix, Mort), cause(flee(Mort), human(John)) |

TABLE 18: Types of analogical similarity.
finding correspondences between elements of the analogical episodes, as in the mapping (Section 3.1.2.3 below), which is computationally expensive. Moreover, unlike for mapping, where only two analogical episodes are considered, in the retrieval process alignment should be repeated for the target analogical episode and each of the base analogical episodes, making such an implementation of the retrieval prohibitive.

To reduce the computational costs, a two-stage filter and refine (F&R) approach is used in the traditional models of analogical retrieval. At the filtering step, the target analogical episode is compared to all the base analogical episodes using a low-cost similarity of their feature vectors (that only counts the frequency of symbolic names in the analogical episodes, without taking into account the structure). The most similar base analogical episodes are selected as prospective candidates. At the refining step, the candidates are then compared to the target analogical episode by the value of their structural similarity. Computationally expensive mapping algorithms (Section 3.1.2.3 below) are used for calculating the structural similarity. As the final result, the analogical episodes with the highest structural similarity are returned.

HDC/VSA have been applied to modeling of the analogical retrieval by Plate (see, e.g., [2], [258]–[260] and references therein). In HDC/VSA, both the set of structure elements and their arrangements influence their HV similarity, so that similar structures (in this case, analogical episodes) produce similar HVs. Because the HV similarity measures are not computationally expensive, the two-stage F&R approach of the traditional models is not needed. Using HRR, it was shown that the results obtained by a single-stage HV similarity estimation are consistent with both the empirical results in psychological experiments as well as the aforementioned leading traditional models of the analogical retrieval. Note that Plate experimented with analogical episodes different from those tested in the leading models, but they still belong to the proper similarity types, as shown in Table 18. Similar results were also reported in [261] for SBDR using Plate’s episodes.

The study [262] applied SBDR to represent the analogical episodes in the manner of [261]. However, the performance was evaluated using the test bases of the most advanced models of analogical retrieval. The results demonstrated some increase in the recall and a noticeable increase in the precision compared to the leading traditional (two-stage) models. The authors also compared the computational complexity and found that in most of the cases the HDC/VSA approach had advantages over the traditional models.

### 3.1.2.3 Analogical mapping:

The most influential models of analogical mapping include the Structure Mapping Engine (SME) [263] and its versions and further developments [264], as well as the Analogical Constraint Mapping Engine (ACME) [265]. SME is a symbolic model that uses a local-to-global alignment process to determine correspondences between the elements of analogical episodes. SME’s drawback is a rather poor account of semantic similarity. Also, structure matching in SME is computationally expensive, so it is prohibitively expensive to use SME during retrieval by comparing input to each of (many) analogical episodes in the memory containing all analogical episodes for a structure-sensitive comparison.

ACME is a localist connectionist model that determines analogical mappings using a parallel constraint satisfaction network. Unlike SME, ACME relies not only on the structural information, but also takes into account semantic and pragmatic constraints. ACME is usually even more computationally expensive than SME.

Further models of mapping based on HRR and BSC were proposed that use techniques based on holistic transformations (Section 3.1.1.2). One of the limitations of these studies is that the approach was not demonstrated to be scalable to large analogical episodes. HRR was also used in another model for the analogical mapping [251] called DRAMA, where the similarity between HVs was used to initialize a localist network involved in the mapping.

In [261], similarity of HVs (formed with SBDR) of the analogical episodes was used for their mapping. However, this technique worked only for the most straightforward mapping cases. In [266], several alternative techniques for mapping with SBDR were proposed (including direct similarity mapping, re-representation by substitution of identical HVs, and parallel traversing of structures using higher-level roles) and some of them were demonstrated on complex analogies. However, the techniques are rather sophisticated and used sequential operations.

In [267], a kind of re-representation of an analog’s element HVs was proposed to allow the analogical mapping of the resultant HVs on the basis of similarity. The re-representation approach included the superposition of two HVs. One of those HVs was obtained as the HV for the episode’s element using the usual representational scheme of the episode, e.g., the HV used for the retrieval (compositional structure representation by HVs, see Section 2.2.4 in [3]). This HV took into account the semantic similarity. The other HV was the superposition of HVs of elements higher-level roles. This took into account the structural similarity. The proposed procedure was tested in several experiments on simple analogical episodes used in previous studies (e.g., in [2], [266], and on rather complex analogical episodes previously used only in the leading state-of-the-art models, e.g., “Water Flow-Heat Flow”, “Solar System-Atom”, and “Schools” [246], [265]. It produced the correct mapping results. The analogical inference was also considered. The computational complexity of the proposed approach was rather low and was largely affected by the dimensionality of HVs.

The problem with the current models of HDC/VSA for analogical mapping is that they lack interaction and competition of consistent alternative mappings. They could probably be improved by using an approach involving the associative memory akin to [268].

Finally, an important aspect of HDC/VSA usage for analogical mapping and reasoning is the compatibility with the existing well-established formats of knowledge representation. This will facilitate the unification of symbolic and subsymbolic approaches for cognitive modeling and Artificial Intelligence. The work in [269] presented a proof of concept of the mapping between Resource Description Framework Schema ontology and HDC/VSA.
3.1.2.4 Graph isomorphism: The ability to identify some form of “structural isomorphism” is an important component of analogical mapping [2], [263]. The abstract formulation of isomorphism is graph isomorphism. In [268], an interesting scheme was proposed for finding the graph isomorphism with HDC/VSA and associative memory. The scheme used the mechanism proposed in [270]. The paper presented the HDC/VSA-based implementation of the algorithm proposed in [271], which used replicator equations and treated the problem as a maximal-clique-finding problem. In essence, the HDC/VSA-based implementation transformed the continuous optimization problem into a high-dimensional space where all aspects of the problem to be solved were represented as HVs. The simulation study (unfortunately performed on a simple graph) showed that the distributed version of the algorithm using the mechanism from [270] mimics the dynamics of the localist implementation from [271].

3.1.3 Cognitive modeling

In this part, we briefly cover known examples of using HDC/VSA for modeling particular cognitive capabilities, such as sequence memorization or problem-solving, for cognitive tasks like the Wason task [272], n-back task [273], Wisconsin card sorting test [274], Raven’s Progressive Matrices [275], or the Tower of Hanoi [276].

3.1.3.1 HDC/VSA as a component of cognitive models: As argued in [277], HDC/VSA is an important tool for cognitive modeling. In cognitive science, HDC/VSA has been commonly used as a part of computational models replicating experimental data obtained from humans. For example, in [278] HVs were used as the representational scheme of a computational model. The model was tested using categorization studies considering three competing accounts of concepts: “prototype theory”, “exemplar theory”, and “theory theory”. The model was shown to be able to replicate experimental data from categorization studies for each of the accounts. It is also worth mentioning that there are numerous works working on HDC/VSA (Section 2.2.1) to form models replicating the results obtained in language-related cognitive studies, see, e.g., [74], [82], [86], [93], [94], [97], [279]–[281].

3.1.3.2 Modeling human working memory with HDC/VSA: A topic that was studied by different research groups working on HDC/VSA is sequence memorization and recall. For example, it was demonstrated in [282] (see Section 3.3 in [3]) that a HDC/VSA-based representation of sequences performed better than localist representations when compared on the standard benchmark for behavioral effects. Some studies [283]–[290] demonstrated how the recall of sequences represented in HVs (Section 3.3 in [3]), albeit with slightly different encoding, can reproduce the performance of human subjects on remembering sequences. This is profound as it demonstrates that simple HDC/VSA operations can reproduce basic experimental findings of human memory studies. An alternative model was proposed in [291]. Importantly, this work linked the neuroscience literature to the modeling of memorizing sequences with HDC/VSA.

3.1.3.3 Raven’s Progressive Matrices: Raven’s Progressive Matrices is a nonverbal test used to measure general human intelligence and abstract reasoning. Some simple geometrical figures are placed in a matrix of panels, and the task is to select the figure for an empty panel from a set of possibilities [292]. Note that the key work related to Raven’s progressive matrices goes back to the 1930s (see [293]). It was widely used to test human fluid intelligence and abstract reasoning since the 1990s [294]. The task was first brought to the scope of HDC/VSA in [275], [295] using HRR and its subsequent implementation in spiking neurons. Later studies [296], [297] demonstrated that other HDC/VSA models (BSC and MAP, respectively) can also be used to create systems capable of solving a limited set of Raven’s Progressive Matrices containing only the progression rule.

In all studies the key ingredient of the solution was the representation of a geometrical panel by its HV (e.g., giving access to the symbolic representation of the panel followed by using role-filler bindings of the shapes and their quantity present in the panel). Subsequently, the HV corresponding to the transformation between the HVs of the adjacent pairs of panels was obtained using the ideas from [240] (see Section 3.1.1.2). The transformation HV was then used to form a prediction HV for the blank panel in the matrix. The candidate answer with the HV most similar to the prediction HV was then chosen as the answer to the test. All of the previously described studies have two limitations: first, they assume the perception system provides the symbolic representations that support the reasoning for solving Raven’s Progressive Matrices test; and second, they only support the progression rule. A recent work [298] addressed these limitations by positioning VSA/HDC as a common language between a neural network (to solve the perception issue) and a symbolic logical reasoning engine (to support more rules). Specifically, it exploited the superposition of multiplicative bindings in a neural network to describe raw sensory visual objects in a panel and used Fourier Holographic Reduced Representations (FHRR) to efficiently emulate a symbolic logical reasoning with a rich set of rules [298].

3.1.3.4 The Tower of Hanoi task: The Tower of Hanoi task, which is a simple mathematical puzzle, is another example of a task used to assess problem solving capabilities. The task involves three pegs and a fixed number of disks of different sizes with holes in the middle such that they can be placed on the pegs. Given a starting position, the goal is to move the disks to a target position. Only one disk can be moved at a time and a larger disk cannot be placed on top of a smaller disk.

In [276], an HDC/VSA-based model capable of solving the Tower of Hanoi tasks was presented. The binding and superposition operations were used to form HVs of the current position and a set of goals identified by the model. The model implemented a known algorithm for solving the task given valid starting and target positions. The performance of the model was compared to that of humans regarding of time delays, which were found to be qualitatively similar.

3.1.3.5 Modeling the visual similarity of words: Modeling human perception of word similarity with HDC/VSA was based on the experimental data obtained for human subjects in [299]. The task was to model the human patterns
of delays in priming tasks with the similarity values of sequence HVs obtained from various HDC/VSA models and for various schemes for sequence representation (Section 3.3 in [3]).

In the task of modeling restrictions on the perception of word similarity, four types of restrictions (a total of 20 similarity patterns) were summarized in [282]. In [282], the BSC model was employed to represent symbols in their positions; various string representations with substrings were also used. Symbols in positions with correlated HVs to represent nearby positions were studied in [300] for the BSC, FHRR and HRR models. Their results demonstrated partial satisfaction of the restrictions. On the other hand, string representations from [301] with HRR, and symbols-at-correlated positions representations from [54] with SBDR using permutations as well as the ones from [302] with FHRR met all the restrictions for certain choices of the similarity measure and values of a scheme’s hyperparameters. In the task of finding correlations between human and model similarity data, [54], [302] demonstrated results were on a par with those of the string kernel similarity measures from [303].

3.1.3.6 General-purpose rule-based and logic-based inference with HVs: The study in [304] presented a spiking neurons model that was positioned as a general-purpose neural controller. The controller was playing a role analogous to a production system capable of applying inference rules. HDC/VSA and their operations played a key role in the model providing the basis for representing symbols and their relations. The model was demonstrated to perform several tasks: repeating the alphabet, repeating the alphabet starting from a particular letter, and answering simple questions similar to the ones in Section 3.1.1.1. Another realization of a production system with the Tensor Product Representations model was demonstrated in [305]. Several examples of hierarchical reasoning using the superposition operation on context HVs representing hyponyms to form representations of hypernyms were presented in [306].

In [307], it was demonstrated how HVs can be used to represent a knowledge base with clauses for further performing deductive inference on them. The work widely used negation for logical inference, which was also discussed in [308]. Reasoning with HRR using modus ponens and modus tollens rules was demonstrated in [309]. The works in [310]–[312] discussed the usage of VSA/HDC for the realization of context logic language and demonstrated the inference procedures.

3.1.4 Computer vision and scene analysis

This section summarizes different aspects of using HDC/VSA for processing visual data. This is one of the newest and least explored application areas of HDC/VSA.

3.1.4.1 Visual analogies: In [313], a simple analogy-making scenario was demonstrated on 2D images of natural objects (e.g., bird, horse, automobile, etc.). This work took an image representing a particular category, e.g., a bird. The HVs of images were obtained through using convolutional neural networks (Section 3.4.3 in [3]) and cellular automata computations (see [314] for the method description). Several (e.g., 50 in [313]) such binary HVs (e.g., for images of different birds) were superimposed together to form the HV of a category, e.g.,

\[
\text{land} = \text{animal} \odot \text{horse} + \text{vehicle} \odot \text{automobile};
\]

\[
\text{air} = \text{animal} \odot \text{bird} + \text{vehicle} \odot \text{airplane}.
\]

The category HVs were used to form some statements using the HDC/VSA operations. Inspired by the well-known example of “Dollar of Mexico?” (as in the techniques of Section 3.1.1.1) it was shown that one could perform queries of a similar form as “What is the Automobile of Air?” (AoA), but using HVs formed from the 2D images:

\[
\text{AoA} = \text{air} \odot (\text{land} \odot \text{automobile}).
\]

The system demonstrated a high accuracy (98%) of correct analogy-making on previously unseen images of automobiles.

3.1.4.2 Reasoning on visual scenes and Visual Question Answering: Visual Question Answering is defined as a task where an agent should answer a question about a given visual scene. In [315], a trainable model was presented that used HDC/VSA for this task. The model in [315] differs from the state-of-the-art solutions that usually include a combination of a recurrent neural network (handles questions and provides answers) and a convolutional neural network (handles visual scenes). The model included two parts. The first part transformed a visual scene into an HV describing the scene using a neural network. This part used only one feed-forward neural network, which took a visual scene and returned its HV. The second part of the model defined the item memory of atomic HVs as well as HVs of questions along with their evaluation conditions in terms of cosine similarity thresholds. The neural network was trained to produce HVs associated with a dataset of simple visual scenes (two figures in various combinations of four possible shapes, colors, and positions). The gradient descent used errors from the question answering on the training data, which included five predefined questions. It was shown that the trained network successfully produced HVs that answered questions for new unseen visual scenes. The five considered questions were answered with 100% accuracy. On previously unseen questions the model demonstrated an accuracy in the range of 60-72%.

Similar to [315], there were attempts in [298], [316], [317] to train neural networks to output HVs representing a structured description of scenes (see also Section 3.4.3 in [3]), which could then be used for computing visual analogies.

Another approach to Visual Question Answering with HDC/VSA was outlined in [318], [319], where a visual scene was first preprocessed to identify objects and construct a scene data structure called the causal matrix (it stored some object attributes including positions). This data structure describing a scene was transformed into a HV that could then be queried using HDC/VSA operations similar to those from [315]. In [320], it was applied to a dataset constructed to
facilitate visual navigation in human-centered environments. This approach was further extended from Visual Question Answering to Visual Dialog in [321].

Another application of HDC/VSA for representation and reasoning on visual scenes, similar in its spirit to the Visual Question Answering, was presented in [322]. The approach represented visual scenes in the form of HVs using Fourier HRR. The paper transformed continuous positions in an image to complex-valued HVs such that the spatial relation between positions was preserved in the HVs (see the “fractional power encoding” in Sections 3.2.1 and 3.4.2 in [3]). During the evaluation, handwritten digits and their positions were identified using a neural network with an attention mechanism. Then the identified information was used to create a complex-valued compositional HV describing the scene. Such scene HVs were used to answer relational queries like “which digit is below 2 and to the left of 1?”.

A further exploration of this task was presented in [323]. The approach in [322] also demonstrated solving a simple navigation problem in a maze. In [324], navigation tasks in 2D environments were further studied by using HVs as an input to a neural network producing outputs directing the movement. Neural networks trained with HVs demonstrated the best performance amongst methods considered. There was also a recent attempt to implement this continuous representation with spiking neurons [325].

### 3.2 Cognitive architectures

HDC/VSA have been used as an important component of several bio-inspired cognitive architectures. Here, we briefly describe these proposals.

#### 3.2.1 Semantic Pointer Architecture Unified Network

The most well-known example of a cognitive architecture using HDC/VSA is called “Spaun” for Semantic Pointer Architecture Unified Network (see its overview in [326] and a detailed description in [327]). Spaun is a large spiking neural network (2.5 million neurons). Spaun that uses HRR for data representation and manipulation. In the architecture, HVs play the role of “semantic pointers” [278] aiming to integrate connectionist and symbolic approaches. It has an “arm” for drawing its outputs as well as “eyes” for sensing inputs in the form of 2D images (handwritten or typed characters). Spaun (without modifying the architecture) was demonstrated in eight different cognitive tasks that require different behaviors. The tasks used to demonstrate the capabilities of Spaun were: copy drawing of handwritten digits, handwritten digit recognition, reinforcement learning on a three-armed bandit task, serial working memory, counting of digits, question answering given a list of handwritten digits, rapid variable creation, and fluid syntactic or semantic reasoning. The same principles were used in [328] to represent the WordNet knowledge base with HVs allowing enriching, e.g., Spaun, with a memory storing some prior knowledge.

#### 3.2.2 Associative-Projective Neural Networks

The cognitive architecture of Associative-Projective Neural Networks (APNNs) that use the SBDR model was proposed and presented in [31], [308], [329]–[331]. The goal was to show how to construct a complex hierarchical model of the world, that presumably exists in human’s and higher animals’ brains, as a step towards Artificial Intelligence.

Two hierarchy types were considered: the compositional (part-whole) as well as the categorization or generalization (class-instance or is-a) ones. An example of the compositional hierarchy is: letters → words → sentences → paragraphs → text. Another example is the description of knowledge base episodes in terms of their elements of various complexity, from attributes to objects to relations to higher-order relations (see Section 3.1.2). An example of the categorization hierarchy is: dog → spaniel → spaniel Rover or apple → big red apple → this big red apple in hand.

The proposed world model relies on models of various modalities, including sensory ones (visual, acoustic, tactile, motoric, etc.) and more abstract modalities (linguistics, planning, reasoning, abstract thinking, etc.) that are organized in hierarchies. The models are required for objects of different nature, e.g., events, real objects, feelings, attributes, etc. Models (their representations) of various modalities can be combined, resulting in multi-modal representations of objects and associating them with the behavioral schemes (reactions to objects or situations), see details in [31], [308], [329]–[331].

The APNN architecture is based on HDC/VSA (though it was proposed long before the terms HDC and VSA appeared), in particular, models are represented by SBDR (Section 2.3.8 in [3]). An approach to formation, storage, and modification of hierarchical models was proposed. This is facilitated by the capability to represent in HVs of fixed dimensionality (for items of various complexity and generality) various heterogeneous data types, e.g., numeric data, images, words, sequences, structures (Section 3 in [3]). As usual for HDC/VSA, the model HVs can be constructed on-the-fly (without learning). APNNs have a multi-module, multi-level, and multi-modal design. A module forms, stores, and processes many HVs representing models of objects of a certain modality and of a certain level of compositional hierarchy. A module’s HVs are constructed from HVs obtained from other modules, such as lower-level modules of the same modality, or from modules of other modalities. The lowest level of the compositional hierarchy consists of modules providing a representation grounding (atomic HVs).

For SBDR, a HV is similar to the HVs of its elements of a lower compositional hierarchy level, as well as to the HVs of the higher level, of which the HV is an element. So, using similarity search (in the item memory of levels) it is possible to recover both the lower-level element HVs and compositional HVs of the higher level.
Each module has a long-term memory where it stores its HVs. A Hopfield-like distributed auto-associative memory [332]–[334] was suggested as a module memory. It performs the clean-up procedure for noisy or partial HVs by a similarity search. However, its unique property is the formation of the second main hierarchy type, i.e., of generalization (class-instance) hierarchies. It is formed when many similar (correlated) HVs are stored (memorized), based on the idea of Hebb’s cell assemblies including cores (subsets of HV 1-components often occurring together, corresponding to, e.g., typical features of categories and object-prototypes) and fringes (features of specific objects), see [31], [332].

It is acknowledged that a world model comprising domain(s) specific knowledge as well as information about an agent itself is necessary for any intelligent behavior. Such a model allows comprehension of the world by an intelligent agent and assists it in its interactions with the environment, e.g., through predictions of action outcomes.

The main problem with the APNN architecture is that not all its aspects have been modeled. For example, there are the following questions, which do not have exact answers:

- how to extract objects and their parts of various hierarchical levels?
- how to determine the hierarchical level an object belongs to?
- how to work with an object that may belong to different hierarchy levels and modules?
- how to represent objects invariant of their transformations?

Also, modeling cores and fringes formation in distributed auto-associative memory is still fragmentary as of now. Finally, it worth noting that similar ideas are currently being developed in the context of deep neural networks [335], [336].

### 3.2.3 Hierarchical Temporal Memory

An interesting connection between HDC/VSA and a well-known architecture called Hierarchical Temporal Memory (HTM) [337] was presented in [338]. The work showed how HTM can be trained in its usual sequential manner to support basic HDC/VSA operations: binding and superposition of sparse HVs, which are natively used by HTM. Even though permutations were not discussed, it is likely that they also could be implemented, so that HTM could be seen as another HDC/VSA model, which additionally has a learning engine in its core.

### 3.2.4 Learning Intelligent Distribution Agent

A version of the well-known symbolic cognitive architecture LIDA (Learning Intelligent Distribution Agent [339]), working with HVs, was presented in [340]. In particular, the Modular Composite Representations model was used [341]. Moreover, memory mechanisms in the proposed architecture were also related to HDC/VSA: an extension of Sparse Distributed Memory [342], known as Integer Sparse Distributed Memory [343], was used. The usage of HVs allowed resolving some of the issues with the original model [339], which relies on directed graph-like structures such as representation capability, flexibility, and scalability.

### 3.2.5 Memories for cognitive architectures

Memory is one of the key components of any cognitive architecture and for modeling cognitive abilities of humans. There is a plethora of memory models. For example, MINERVA 2 [344] is an influential computational model of long-term memory. However, in its original formulation MINERVA 2 was not very suitable for an implementation in connectionist systems. Nevertheless, it was demonstrated in [345] that the Tensor Product Representations model (Section 2.3.2 in [3]) can be used to formalize MINERVA 2 as a fixed size tensor of order four. Moreover, it was demonstrated that the lateral inhibition mechanism for HDC/VSA [268] and HRR can be used to approximate MINERVA 2 with HVs. HVs allowed to compress the exact formulation of the model, which relies on tensors, into several HVs, thus making the model more computationally tractable at the cost of lossy representation in HVs.

Another example of using HVs (with HRR) for representing concepts is a Holographic Declarative Memory [288], [346], [347] related to BEAGLE [74] (see Section 2.2.1.2). It was proposed as a declarative and procedural memory in cognitive architectures. It was shown that the memory can account for many effects such as primacy, recency, probability estimation, interference between memories, and others.

In [86], [348] BEAGLE (Section 2.2.1.2) was extended to store (instead of one context HV per word) episodic memory of the observed data as HVs of all the contexts. This extension was called the instance theory of semantics. Each word was represented by an atomic random HV. A word’s context (a sentence) HV is constructed as a superposition of its word HVs and is stored in the memory. The HV of some query word is constructed as follows. First, the $\text{sim}_{\text{cos}}$ of the query word HV and each context HV is calculated and raised to a power, producing a vector of “trace activations”. Then, context HVs are weighted by traces and summed to produce the retrieved (“semantic”) HV of the query word.

The study in [349] introduced a “weighted expectancy subtraction” mechanism that formed actual context HV as follows. First, the context HV produced the retrieved HV as explained above. Then, the retrieved HV was weighted and subtracted from the initial context HV. During the retrieval, the weighted HV of the second retrieval iteration was subtracted from the HV of the first retrieval iteration. This allowed flexibly controlling the construction of general versus specific semantic knowledge. The work in [350] proposed CogNGen - a core of a cognitive architecture that combines predictive processing based on neural generative coding and HDC/VSA models of human memory. The CogNGen architecture learns across diverse tasks and models human performance at larger scales.
4 DISCUSSION

4.1 Application areas

4.1.1 Context HVs

When it comes to context HVs, Random Indexing and Bound Encoding of the Aggregate Language Environment have appeared as improvements to Latent Semantic Analysis – e.g., they do not require Singular Value Decomposition and can naturally take order information into account. However, they were largely overshadowed after the introduction of “neural” word embeddings in Word2vec [95] or GloVe [237]. The latter are the result of an iterative process, which takes numerous passes via training data to converge. At the same time, an important fact is that distributional models such as Latent Semantic Analysis can in fact benefit from some techniques used in neural word embeddings [351]. Concerning, e.g., Bound Encoding of the Aggregate Language Environment, as recently demonstrated in [112], the method can benefit from negative information. Nevertheless, the current de facto situation in the natural language processing community is that Bound Encoding of the Aggregate Language Environment and Random Indexing methods are rarely the first candidates when it comes to choosing word embeddings. On the other hand, since Bound Encoding of the Aggregate Language Environment has been proposed within cognitive science community, it still plays an important role in modeling cognitive phenomena related to memory and semantics [86], [349]. Also, in contrast to the iterative methods, Random Indexing and Bound Encoding of the Aggregate Language Environment only require a single pass through the training data to form context HVs. In some situations, this could be an advantage, especially since the natural language processing community is becoming increasingly concerned about the computational costs of algorithms [352].

4.1.2 Classification

While right now the classification with HDC/VSA is flourishing, there are still important aspects that are often not taken into account in these studies.

4.1.2.1 Formation of HVs: An important aspect of the formation of HVs is the initial extraction of features from raw data such as 2D images or acoustic signals. Usually, directly transforming such data into HVs does not result in a good performance, so an additional step of extracting meaningful features is required.

Another important aspect is that, when constructing HVs from feature data for classification, in most cases the transformation of data into HVs is somewhat ad hoc. While there likely will not be a straightforward answer to how transforming data into HVs, it is still important to mention several issues.

It is a well-known fact that the advantage of nonlinear transformation is that classes not linearly separable in the original representation, might become linearly separable after a proper nonlinear transformation to a high-dimensional space (often called lift). This allows using not only k-Nearest Neighbor classifiers, but also well-developed linear classifiers to solve problems that are not linearly separable. So, nonlinearity is an important aspect of transforming data into HVs. All transformations of data into HVs that we are aware of seem to be nonlinear. However, there are no studies that scrutinize and characterize the nonlinearity properties of HVs obtained from the compositional approach. Moreover, most of the studies choose a particular transformation of numeric vector and stick to it. One of the most common choices is randomized “float” coding [9], [125], [185], [353]. There is, however, a recent study [63], [64] that established a promising connection between kernel methods [354], [355] and the fractional power encoding for representing numeric vectors as well as an earlier algorithm for approximating a particular type of kernels (tree kernels) with the HRR model [356]. In our opinion, the transformation of data into HVs is a hyperparameter of the model and using, e.g., cross-validation to choose the most promising transformation will likely be the best strategy when considering a range of different datasets.

4.1.2.2 Choice of classifier: As we saw in Section 2.3, centroids are probably the most common approach to forming a classifier in HDC/VSA. This is understandable, since centroids have an important advantage in terms of computational costs – they are very easy to compute. However, as pointed out in [357], the result of superposition does not provide generalization in itself, it is just a representation of combinations of HVs of training samples. Practically, it means that the centroids are not the best performing approach when it comes to classification performance. One way to improve the performance is to assign weights when including new samples into centroids [147], [183]. It was also shown that the perceptron learning rule in [141] and loss-based objective in [227] might significantly improve centroids. Earlier work on HV-based classifiers, e.g., [133], [358]–[360] also used linear perceptron and Support Vector Machine classifiers with encouraging results. Note that a large-margin perceptron usually trains much faster than a Support Vector Machine for big data, while providing classification quality at the same level as the Support Vector Machine and usually much higher than that of the standard perceptron. Another recent result [361] is that centroids can be easily combined with a known conventional classifier – generalized learning vector quantization [362]. Using an HDC/VSA transformation of data to HV, the authors obtained state-of-the-art classification results on a benchmark [363]. In general, we believe that when inventing new mechanisms of classification with HDC/VSA, it is important to report the results on collections of datasets instead of only a handful of datasets. For example, for feature-based classification the UCI Machine Learning Repository [364] and subsets thereof (e.g., [363]) are a common choice (examples of HDC/VSA using it are [361], [365], [366]). For univariate temporal signals, the UCR Time Series Archive [367] is a good option that was used, e.g., in [368]. If a reported mechanism targets a more specific application area, it would be desirable to evaluate it on a relevant collection for that area.

Many other types of classifiers such as k-Nearest Neighbors [225] are also likely to work with HVs as their input. When HVs are generated by a nonlinear transformation (in [225], using an HDC/VSA-guided convolutional neural network
4.2 Interplay with neural networks

4.2.1 HVs as input to neural networks

One of the most obvious ways to make an interplay between HDC/VSA and neural networks is by using HVs to represent input to neural networks. This is a rather natural combination of the two, because, in essence, neural networks often work with distributed representations. So, processing information distributed in HVs is not an issue for neural networks.

However, since HVs are high-dimensional, it is not always possible to use them as the input: the size of neural networks’ input layer should be set to $D$ (e.g., in [397]), whereby a fully-connected “readout” layer for a task of choice was trained on $D$-dimensional input HVs or even to a tensor comprised of HVs (e.g., to represent each position in the retina by its HV, without superposition of HVs). Moreover, the local structure of the input signal space may become different from that used, e.g., in convolutional neural networks. This could require very different neural network architectures compared to modern deep neural networks.

There are, nevertheless, scenarios where using HVs with neural networks appeared to be beneficial. First, HVs are useful in situations when the data to be fed to a neural network are high-dimensional and sparse. Then HVs can be used to form more compact distributed representations of these data. A typical example of using such high-dimensional and sparse data is $n$-gram statistics. There are works which studied tradeoffs between the dimensionality of HVs representing $n$-gram
in the previous section, randomized recurrent neural networks, known as reservoir computing/echo state networks, can be used to take care of forming fixed size HVs for input of variable size. This mode of a neural network interface has been demonstrated in an automotive context to represent either varying number of intersections being crossed by a vehicle [374] or the dynamically changing environment around a vehicle [169], [170]. Further promising avenues for this model are graphs and natural language processing, since there is a lot of structure in both, which can potentially be represented in HVs [102], [397]. Some investigations in this direction using Tensor Product Representations were presented in [398].

We foresee that this interface mode might expand the applicability of neural networks, as it allows relieving the pressure of forming the task either with fixed size input or in the form of, e.g., a sequence suitable for recurrent neural networks. However, it may require a replacement of the widely used convolutional layers. Although, there are new results [399] suggesting that fully connected neural networks might be a good architecture even for vision tasks.

4.2.2 The use of neural networks for producing HVs

Transforming data into HVs (see Section 3 in [3]) might be a non-trivial task, especially when data are unstructured and of non-symbolic nature as, e.g., in the case of images (Sections 3.4.1 and 3.4.2 in [3]). Also, those transformations are usually not learned. This challenge stimulates the interface between neural networks and HDC/VSA in the other direction, i.e., to transform activations of neural network layer(s) into HVs. For example, as mentioned in Section 3.4.3 in [3], it is very common to use activations of convolutional neural networks to form HVs of images. This is commonly done using standard pre-trained neural networks [124], [127], [400]. Two challenges here are to increase the dimensionality and change the format of the neural network representations to conform with the HV format requirements. The former one is generally addressed by expanding the dimensionality, e.g., by random projection, possibly with a subsequent binarization by thresholding [124], [401]. Some neural networks already produce binary vectors (see [127]), and the transformation into HVs was done by expanding the dimensionality, e.g., by random projection, possibly with a subsequent binarization by thresholding [124], [401]. Some neural networks already produce binary vectors (see [127]), and the transformation into HVs was done by expanding the dimensionality, e.g., by random projection, possibly with a subsequent binarization by thresholding [124], [401].

In [404], it was shown that it is possible to treat the functionality of binarized neural networks with the ideas from high-dimensional geometry. The paper has demonstrated that binarized networks work because of the properties of binary high-dimensional spaces, i.e., the properties used in Binary Spatter Codes [4]. While it is an interesting qualitative result, it did not provide a concrete way to make the two benefit from each other. This is not obvious, since in the standard neural networks all weights are trained via backpropagation, which is rather different from the HDC/VSA principles.

There is, however, a family of randomized neural networks [370] where a part of the network is initialized randomly and stays fixed. There are two versions of such networks: feed-forward (e.g., random vector functional link networks [405] aka extreme learning machines [406]) and recurrent (e.g., echo state networks [407] aka reservoir computing [408]). The way the randomness is used in these networks can be expressed in terms of HDC/VSA operations for both feed-forward [365] and recurrent [409] versions. Conventionally, randomized neural networks were used with real-valued representations. However, since it was realized that these networks can be interpreted in terms of HDC/VSA, it appeared natural to use binary/integer variants (as in Binary Spatter Codes and Multiply-Add-Permute) to produce activations of hidden layers of the networks. This opened the avenue for efficient hardware implementations of such randomized neural networks. Yet another connection between HDC/VSA and feed-forward randomized neural networks was demonstrated in [410] where it was shown that HRR’s binding operation can be approximated by such networks. Finally, in [411] it was shown that the address mechanism from the Sparse Distributed Memory [342] approximates the attention mechanism [412] used in modern neural networks.

4.2.3 HDC/VSA for simplifying neural networks

In [404], it was shown that it is possible to treat the functionality of binarized neural networks with the ideas from high-dimensional geometry. The paper has demonstrated that binarized networks work because of the properties of binary high-dimensional spaces, i.e., the properties used in Binary Spatter Codes [4]. While it is an interesting qualitative result, it did not provide a concrete way to make the two benefit from each other. This is not obvious, since in the standard neural networks all weights are trained via backpropagation, which is rather different from the HDC/VSA principles.

There is, however, a family of randomized neural networks [370] where a part of the network is initialized randomly and stays fixed. There are two versions of such networks: feed-forward (e.g., random vector functional link networks [405] aka extreme learning machines [406]) and recurrent (e.g., echo state networks [407] aka reservoir computing [408]). The way the randomness is used in these networks can be expressed in terms of HDC/VSA operations for both feed-forward [365] and recurrent [409] versions. Conventionally, randomized neural networks were used with real-valued representations. However, since it was realized that these networks can be interpreted in terms of HDC/VSA, it appeared natural to use binary/integer variants (as in Binary Spatter Codes and Multiply-Add-Permute) to produce activations of hidden layers of the networks. This opened the avenue for efficient hardware implementations of such randomized neural networks. Yet another connection between HDC/VSA and feed-forward randomized neural networks was demonstrated in [410] where it was shown that HRR’s binding operation can be approximated by such networks. Finally, in [411] it was shown that the address mechanism from the Sparse Distributed Memory [342] approximates the attention mechanism [412] used in modern neural networks.

4.2.4 HDC/VSA for explaining neural networks

It was discussed in Section 2.4 in [3] that the capacity theory [413] applies to different HDC/VSA models. As mentioned in the previous section, randomized recurrent neural networks, known as reservoir computing/echo state networks, can
be formulated using HDC/VSA. Therefore, capacity theory can also be used to explain memory characteristics of reservoir computing. Moreover, using the abstract idea of dissecting a network into mapping and classifier parts [414], it is possible to apply capacity theory for predicting the accuracy of other types of neural networks (such as deep convolutional neural networks) [415].

In [416], it was shown that Tensor Product Representations approximate representations of structures learned by recurrent neural networks.

4.2.5 The use of HDC/VSA with spiking neural networks

Another direction of the interplay between HDC/VSA and neural networks is their usage in the context of spiking neural networks (SNN). It is especially important in the context of emerging neuromorphic platforms [417], [418]. The main advantage HDC/VSA can bring into the SNN domain is the ease of transformation to spiking activities, either with rate-based coding for HDC/VSA models with scalar components or phase-to-timing coding for HDC/VSA models with phasor components. The Spaun cognitive architecture overviewed in Section 3.2.1 is one of the first examples where the HRR model was used in the context of SNN. The latest developments [419], [420] use FHRR and HRR to implement associative memory and k-Nearest Neighbor classifiers on SNN. Further, these memories were proposed as building blocks for the realization of a holistic HDC/VSA-based unsupervised learning pipeline on SNN [376]. While in [421] the Sparse Block Codes model was mapped to an SNN circuit. In other related efforts, an event-based dynamic vision sensor [51], [216] or an SNN [422], [423] was used to perform the initial processing of the input signals that were then transformed to HVs to form the prediction model.

These works provide some initial evidence of the expressiveness of HDC/VSA on the one hand, and compatibility with SNNs on the other. We therefore foresee that using HDC/VSA as a programming/design abstraction for various cognitive functionalities will soon manifest itself in the emergence of novel SNN-based applications.

4.2.6 “Hybrid” solutions

By “hybrid” in this context we refer to solutions that use both neural networks and some elements of HDC/VSA. Currently, a particularly common primitive used in such hybrid solutions is the representation of a set of role-filler bindings, or superposition of multiplicative bindings. For example, in [424] the weights of several neural networks were stored jointly by using the superposition operation, which alleviated the problem of “catastrophic forgetting”. In [425], activations of layers of a deep neural network were used as filler HVs. They were bound to the corresponding random role HVs and all role-filler bindings were aggregated in a single superposition HV that in turn was used to successfully detect out-of-distribution data. Similarly, in [124], [215], [426], activations of several neural networks were combined together via HDC/VSA operations. In [124], [215], this idea was used to form a single HV compactly representing the aggregated neural networks-based image descriptor while in [426] outputs of multiple neural networks were fused together to solve classification problems. In [427], the superposition of role-filler bindings was used to simultaneously represent the output of a deep neural network when solving multi-label classification tasks. In [298], the activations of the last layer generate a query HV that resembles the superposition of the visual objects available in a panel, whereby each object is uniquely represented by multiplicative binding of its attributes’ HVs. In addition, a review of hybrid solutions combining Tensor Product Representations and neural networks such as Tensor Product Generation Networks [428] can be found in [429]. Finally, it is worth noting that all these solutions in some way relied on the idea of “computing in superposition” [6] suggesting that HVs can be used to simultaneously manipulate several pieces of information.

4.3 Open issues

As introduced at the beginning of this survey, HDC/VSA originated from proposals of integrating the advantages of the symbolic approach to Artificial Intelligence, such as compositionality and systematicity, and those of the neural networks approach to Artificial Intelligence (connectionism), such as vector-based representations, learning, and grounding. There is also the “neural-symbolic computing” [430], [431], or “neurosymbolic AI” [432] community that suggests hybrid approaches to Artificial Intelligence. The key idea is to form an interface so that symbolic and connectionist approaches can work together. At present, HDC/VSA and neural-symbolic computing seem to be rather separate fields that can benefit from synergy. Moreover, the works developing cognitive architectures and Artificial Intelligence with HDC/VSA are rather limited [31], [326], [327].

So far, a major advantage of the HDC/VSA models has been their ability to use HVs in a single unified format to represent data of varied types and modalities. Moreover, the use of HDC/VSA operations allows introducing compositionality into representations. The prerequisite, however, is that the representation of the input data to be transformed into HVs should be able to specify the compositionality explicitly. Nevertheless, despite this advantage, HDC/VSA is usually overlooked in the context of neural-symbolic computing, which calls for establishing a closer interaction between these two communities.

In fact, most of the works use HDC/VSA to reimplement the symbolic Artificial Intelligence primitives or the machine learning functionality with HVs, in a manner suitable for emerging unconventional computing hardware. At the same time, when transforming data into HVs, machine learning is used rarely, if at all. Learning is used mainly for training a classifier based on the already constructed HVs. In most recent studies, the narrative is often to demonstrate solutions to
some simple classification or similarity search problems. In so, the quality of the results is comparable to the state-of-the-art solutions, but the energy/computational costs required by an HDC/VSA-based solution are only a fraction of those of the baseline approaches. These developments suggest that HDC/VSA might find one of their niches in application areas known as “tiny machine learning” and “edge machine learning”. Nevertheless, there is an understanding that manually designing transformations of data into HVs for certain modalities (e.g., 2D images) is a challenging task. This stimulates attempts that use modern learning-based approaches, such as deep neural networks, for producing HVs (see Section 4.2.2). The current attempts, however, are rather limited since they focus heavily on using HVs formed from the representations produced by neural networks to solve some downstream machine learning tasks (e.g., similarity search or classification).

Another approach would be to discover general principles for combining neural networks and HDC/VSA, but currently there are only few such efforts [298], [380], [398], [402], [424], [433] (see also Section 4.2.6). For example, the Tensor Product Representations operations in [398], and the HDC/VSA operations in [298], are introduced into the neural network machinery. These attempts are timely for the connectionist approach to Artificial Intelligence since, despite recent uptick of deep neural networks [434], there is a growing awareness within the connectionist community that reaching Artificial General Intelligence is going to require a much higher level of generalization than that available in modern neural networks [435]–[438]. A recent proposal in [429] could be considered more HDC/VSA-oriented.

One of the milestones toward reaching Artificial General Intelligence is solving the problem of compositionality. For example, [436] stressed the importance of various aspects of binding for achieving compositionality and generalization. The framework of HDC/VSA has a dedicated on-the-fly operation for binding (Section 2.2.3.2 in [3]), which does not require any training. The neural implementation of binding in the context of SNNs is still an open issue. There are, however, two recent proposals [421], [439] aiming at addressing this issue.

Further, [437] suggested that various inductive biases are required for human-level generalization, including compositionality and discovery of causal dependencies. HDC/VSA have a potential to achieve this through analogical reasoning (Section 3.1.2). However, the progress is held back by the lack of mechanisms to build the analogical episodes, e.g., by observing the outer world or by just reading texts. The analogical episodes should include two major types of hierarchy, the compositional (“part-whole”) one, and the generalization (“is-a”) hierarchy. We believe that associative memories [332] may provide one way to form “is-a” hierarchies (see the discussion in [31]), but this topic has not yet been studied extensively in the context of HDC/VSA. In terms of forming part-whole hierarchies from 2D images, a recent conceptual proposal was given in [438]. The essence of the proposal is learning to parse 2D images by training a neural network and using the similarity of the learnt high dimensional vector representations for analogical reasoning. An interesting direction for future work is to see how such representations can be paired with the analogical reasoning capabilities of HDC/VSA. However, all the above and other proposals from the connectionist community rely on learning all the functional lacking in neural networks. But there are also discussions on the innate structure of neural networks [435]. For the sake of fairness, it should be noted that current HDC/VSA also lack ready implementations for most of the mentioned above functionality. There are a lot of open problems that should be addressed to build truly intelligent systems. Below we list some of them. Some problems related to the internals of HDC/VSA are the following:

- **Recovery.** Recovering element HVs from compositional HVs. Section 2.2.5 in [3] presented some ideas for recovering the content of HVs. However, for most of the HDC/VSA models, knowledge of all but one bound HV is required. This makes the recovery problem combinatorial (but see a proposal in [59], [60]).
- **Similarity.** In many of the HDC/VSA models, the results of the binding operation are not similar as soon as a single input HV is dissimilar. While it is often convenient, or even desired, that the result of the binding operation is dissimilar to its input HVs, the price to be paid is weak or no similarity in the resultant HV that includes the same HVs in slightly different combinations. This might hinder the similarity search that is at the heart of HDC/VSA and required in many operations such as the recovery or clean-up procedures.
- **Memory.** Quantity and allocation of item memories. How many item memories and which HVs of all available HVs should be placed in each of the? Types of the item memories to be used? List memories provide reliable retrieval, but are problematic for generalization. Distributed auto-associative memories have problems with dense HVs (but see [440]).
- **Generalization of HVs.** How to form generalized HVs containing typical features of objects and, is-a hierarchies of objects, but preserve HVs of specific objects as well? Distributed auto-associative memories have potential for generalization by unsupervised learning [332].
- **Generativity.** Is it possible to make a new meaningful compositional HV without constructing it from atomic HVs? Is it possible to produce meaningful input (e.g., fake image or sound, as in deep neural networks) from some generated generalized or specific compositional HV?
- **Similarity-based associations.** How to select the particular association needed in the context from myriads of possible associations? For example, between HVs of a part and of a whole, or between HVs of a class and of a class instance.
- **Parsing.** How to parse the raw sensory inputs into a meaningful part-whole hierarchy of objects?
- **Holistic representation.** HVs are holistic representations. However, for the comparison of objects we may need to operate with their parts. Part HVs can be found given a particular holistic HV. Is it possible to form holistic HVs of very different inputs of the same class so that they are similar enough to be found in memory?
- **Dynamics.** Representation, storage, similarity-based search, and replaying of spatial-temporal data (e.g., a kind of video).
• Probabilistic representation. Representation of object(s) reflecting the probabilities assigned to them.
• Learning. How to learn, for example, the most suitable transformation of input data into HVs for a given problem?
  Also, learning HV for behaviors, including reinforcement learning.

Let us also touch on problems specific not only to HDC/VSA:
• Representation invariance. To recognize the same object in various situations and contexts, we need some useful
  invariance of representation that makes similar diverse manifestations of the same object.
• Context-dependent representation. Representing data as objects and getting the proper representation of an object in
  a particular context.
• Context-dependent similarity. For example, depending on the context a pizza is similar to a cake but also to a frisbee.
  How can such a context-dependent similarity be implemented?
• Context-dependent processing. The type of processing to be applied to data should take into account the context, such
  as bottom-up and top-down expectations or system goals.
• Hierarchies. Forming “part-whole” and “is-a” hierarchies. Which level of “part-whole” hierarchy does an object belong
  to? An object can belong to various levels for different scenes of the same nature. This is also connected to the image
  scale. An object can belong to very many hierarchies for scenes of varied nature. Concerning “is-a” (class-instance)
  hierarchy, an object can belong to various hierarchies of classes, subclasses, etc. in different contexts.
• Cause-effect. Cause-effect extraction in new situations could be done by analogy to familiar ones. Generalizations and
  specifications using is-a hierarchy are possible.
• Interface with symbolic representations. Designers of cognitive agents have to solve the dual problem of both building
  a world model and building it so that it can be expressed in symbols in order to interact with humans.
• The whole system control. Most of the solutions generally rely on conventional deterministic mechanisms for flow
  control. It is, however, likely that the control of the system should also be trainable, so that the system could adjust it
  for new tasks.

All the problems described above are rarely (if at all) considered in the scope of HDC/VSA. To the best of our
knowledge, one study that discussed some of these problems is [31]. Moreover, it is not fully clear which of these
problems are related to the general problems of building Artificial General Intelligence, and which ones are due to
the architectural peculiarities of neural networks and HDC/VSA. In other words, the separation provided above is
not necessarily unequivocal. We believe, however, that building Artificial General Intelligence will require facing these
problems anyway. Finally, we hope that insights from HDC/VSA, symbolic Artificial Intelligence, and neural networks
will contribute to the solution.

5 Conclusion
In this two-part survey, we provided comprehensive coverage of the computing framework known under the names
Hyperdimensional Computing and Vector Symbolic Architectures. Part I of the survey [3] covered existing models and
transformations of input data of various types into distributed representations. In this Part II, we focused on known
applications of Hyperdimensional Computing/Vector Symbolic Architectures including the use in cognitive modeling and
cognitive architectures. We also discussed the open problems along with promising directions for the future work. We hope
that for newcomers, this two-part survey will provide a useful guide of the field, as well as facilitate its exploration and
the identification of fruitful directions for research and exploitation. For the practitioners, we hope that the survey will
broaden the vision of the field beyond their specialization. Finally, we expect that it will accelerate the convergence of this
interdisciplinary field to discipline with common terminology and solid theoretical foundations.

References
[1] T. A. Plate, “Holographic Reduced Representations,” IEEE Transactions on Neural Networks, vol. 6, no. 3, pp. 623–641, 1995.
[2] ——, Holographic Reduced Representations: Distributed Representation for Cognitive Structures. Stanford: Center for the Study of Language and
  Information (CSLI), 2003.
[3] 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 Computing Surveys, vol. 55, no. 6, pp. 1–40, 2022.
[4] P. Kanerva, “Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random
  Vectors,” Cognitive Computation, vol. 1, no. 2, pp. 139–159, 2009.
[5] R. W. Gayler, “Vector Symbolic Architectures Answer Jackendoff’s Challenges for Cognitive Neuroscience,” in Joint International Conference
  on Cognitive Science (ICCS/ASCS), 2003, pp. 133–138.
[6] D. Kleyko, M. Davies, E. P. Frady, P. Kanerva, S. J. Kent, B. A. Olshausen, E. Osipov, J. M. Rabaey, D. A. Rachkovskij, A. Rahimi, and F. T.
  Sommer, “Vector Symbolic Architectures as a Computing Framework for Nanoscale Hardware,” arXiv:2106.05268, pp. 1–28, 2021.
[7] T. A. Plate, “A Common Framework for Distributed Representation Schemes as Compositional Structure,” in Connectionist Systems for
  Knowledge Representation and Deduction, 1997, pp. 15–34.
[8] A. Rahimi, S. Datta, D. Kleyko, E. P. Frady, B. Olshausen, P. Kanerva, and J. M. Rabaey, “High-dimensional Computing as a Nanoscale
  Paradigm,” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 64, no. 9, pp. 2508–2521, 2017.
[9] 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, vol. 107, no. 1, pp. 123–143, 2019.
[10] P. Neubert, S. Schubert, and P. Protzel, “An Introduction to Hyperdimensional Computing for Robotics,” KI - Künstliche Intelligenz, vol. 33,
  no. 4, pp. 319–330, 2019.
[440] H. Ramsauer, B. Schäfl, J. Lehner, P. Seidl, M. Widrich, T. Adler, L. Gruber, M. Holzleitner, M. Pavlović, G. K. Sandve et al., “Hopfield Networks Is All You Need,” in *International Conference on Machine Learning (ICML)*, 2021, pp. 1–95.