Classification and Recall With Binary Hyperdimensional Computing: Tradeoffs in Choice of Density and Mapping Characteristics

Denis Kleyko, Abbas Rahimi, Dmitri A. Rachkovskij, Evgeny Osipov, and Jan M. Rabaey, *Fellow, IEEE*

***Abstract*— Hyperdimensional (HD) computing is a promising paradigm for future intelligent electronic appliances operating at low power. This paper discusses tradeoffs of selecting parameters of binary HD representations when applied to pattern recognition tasks. Particular design choices include density of representations and strategies for mapping data from the original representation. It is demonstrated that for the considered pattern recognition tasks (using synthetic and real-world data) both sparse and dense representations behave nearly identically. This paper also discusses implementation peculiarities which may favor one type of representations over the other. Finally, the capacity of representations of various densities is discussed.**

***Index Terms*— Associative memory (AM), gesture recognition, hyperdimensional (HD) computing, pattern recognition, sparse distributed representation (SDR), vector symbolic architectures (VSAs).**

I. Introduction

THE major challenge for intelligent electronic appliances is to turn the raw data into information and knowledge with minimum power consumption. Hyperdimensional (HD) computing [1] also known as vector symbolic architectures (VSAs) [2], which is the focus of this paper, answers this quest having in mind an emerging class of imprecise computational elements operating at ultralow voltages with stochastic devices that are prone to bit errors [3]–[5].

Manuscript received March 29, 2017; revised September 18, 2017 and January 9, 2018; accepted March 5, 2018. Date of publication April 5, 2018; date of current version November 16, 2018. This work was supported in part by the Swedish Research Council under Grant 2015-04677 and in part by Systems on Nanoscale Information fabriCs, one of the six SRC STARnet Centers, sponsored by MARCO and DARPA. The work of D. Kleyko was supported by Stiftelsen Seth M Kempes Stipendiefond for his research visit to UC Berkeley. *(Corresponding author: Denis Kleyko.)*

D. Kleyko and E. Osipov are with the Department of Computer Science Electrical and Space Engineering, Luleå University of Technology, 971 87 Luleå, Sweden (e-mail: denis.kleyko@ltu.se; evgeny.osipov@ltu.se).

A. Rahimi is with the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA 94720 USA, and also with the Department of Information Technology and Electrical Engineering, ETH Zurich, 8092 Zürich, Switzerland (e-mail: abbas@eecs.berkeley.edu; abbas@ee.ethz.ch).

D. A. Rachkovskij is with the International Research and Training Center for Information Technologies and Systems, Department of Neural Information Processing Technologies, 02000 Kiev, Ukraine (e-mail: dar@infrm.kiev.ua).

J. M. Rabaey is with the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA 94720 USA (e-mail: jan@eecs.berkeley.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2018.2814400

VSAs are a bio-inspired family of methods for representing concepts (letters, phonemes, and features), their meanings and allows implementing sophisticated reasoning based on simple operations. The core of the method is HD distributed data representation. In HD computing everything is represented by vectors of very high dimensionality (i.e., vectors of several thousand bits). High-dimensional vectors (HD vectors) represent data in a distributed manner, i.e., individual positions do not have specific meaning in contrast to the traditional (localist) data representation in computers. Over the past few years, it has been demonstrated that the principles of HD computing can be applied to create systems capable of solving cognitive tasks, for example, Raven's progressive matrices [6]–[8] or analogical reasoning [9]–[11]. In [12] and [13], it is advocated that VSA is one of the suitable candidates for implementing functionality of general artificial intelligence.

This paper focuses on engineering aspects of the VSA-based technical systems. In particular, binary HD representations are considered. When it comes to using binary VSA in practical applications, for example, solving classification problems (e.g., [14], [15]) or creating one-shot reinforcement learning pipeline [16]; several important design choices have to be made as follows.

1) What density of the binary HD vectors to choose?
2) How to map the data from the original representation to the HD space?

While the interest in HD computing is currently increasing, these design choices are often made *ad hoc* without proper justification or even understanding of the consequences of for example power consumption, and alternative choices. This paper fills this gap by presenting its major contribution: a consideration of the tradeoffs of selecting major parameters of binary distributed representations.

This paper is structured as follows. Section II introduces fundamentals of HD computing and presents notations and terminology used in this paper. Section III presents a concise survey of the related work utilizing HD computing and studying its properties. Section IV defines the problem of classification using HD representations, describes the showcase scenarios used for evaluation of the tradeoffs, and outlines the approach. The tradeoffs of selecting hyperparameters of HD representations are described in Section VI. The capacity of HD representations of various densities is discussed in Section VII. Conclusions are drawn in Section VIII.

II. Fundamentals of Binary Hyperdimensional Distributed Representations

In a localist representation, which is used in all general purpose computers, single bits or small groups of bits can be interpreted without reference to the other bits. In a distributed representation, on the contrary, it is only the total set of bits that can be interpreted. Computing with distributed representations utilizes statistical properties of representation spaces with very high dimensionality, which is fixed for a given application. The operations in HD computing often return noisy approximations to the correct answers. Therefore, item memory, also referred to as clean-up memory, is needed to recover clean representations assigned to specific concepts. There are several flavors of HD computing with distributed representations depending on the numerical type of HD vector elements, which can be real numbers [17]–[20], complex numbers [21], binary numbers [1], [22], or bipolar [18], [23]. This paper considers engineering aspects of binary distributed representations only.

In this paper, binary HD computing is considered purely for solving classical tasks of supervised machine learning (classification and pattern matching) leaving the more "cognitive" aspects of VSA (analogical reasoning, semantic generalization, relational representation, and analysis) outside the scope. For the sake of further discussion the term pattern is used to refer to a set of features. Distributed representations of features could be either learned [24], [25] or produced through a function of mapping.[1] To highlight the engineering tradeoffs this paper uses the it mapping method of producing distributed representations of the features. For example, symbolic representations can often be represented by dissimilar distributed representations, while numeric data in turn require preservation of the similarity between points in the original space (see examples of such mappings for sparse distributed representations (SDRs) in [26]). Therefore, there is no single right way to perform the mapping and engineering choices must be made about what properties of the input features are to be preserved in the mapped representations.

There are two major hyperparameters to choose when using binary distributed representations in the supervised learning context: the density of HD vectors and the mapping function. With respect to the density, the possibilities include computing with sparse or dense distributed representations, where dense binary representation is a vector in which the "1" and "0" vector elements are roughly equiprobable. With respect to the mapping function the possibilities are to map features with preservation of their similarity in the original (low dimensional) domain or without it. While the implications of the choice of the mapping function are discussed in Section V-B, the sections below introduce the major operations of sparse and dense HD computing.

[1]In the literature, the term "projection" from an initial representation to a distributed representation is sometimes used as a synonym of mapping

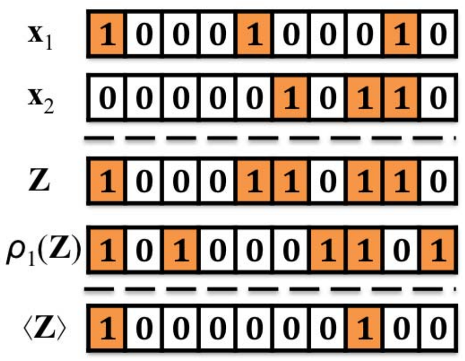


Fig. 1. Toy example of the CDT procedure according to (4) with one iteration, $T = 1$. Two atomic HD vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ ($G = 2$) form their superposition vector $\mathbf{Z}$. Next, the elements of $\mathbf{Z}$ are permuted to form $\rho_1(\mathbf{Z})$. The permutation is done via the cyclic shift operation by two elements. Finally, the thinned vector $\langle\mathbf{Z}\rangle$ is the result of the elementwise conjunction between $\mathbf{Z}$ and $\rho_1(\mathbf{Z})$.

A. Computing With Binary Sparse Distributed Representations

From the point of view of the biological plausibility, SDRs are the closest ones to neural representation of information in brains of biological organisms. That is, in biological brains only a small proportion of neurons is simultaneously active. While this may be so in biological brains for reasons of evolutionary accident, here we are concerned with engineering reasons for choosing a level of sparsity. The fundamentals of SDR computing are described in [13] and [22]. In SDR, the number of "1" elements is much less than that of "0." Example of sparse representation is vector dimensionality $N = 100\,000$ and the number of "1" in an HD vector about $M = 1000$. So the probability of "1" $p_1 \ll 0.5$, $p_0 = 1 - p_1$.

The actual density (empirical probability of "1") of an HD vector $\mathbf{x}$ is determined as its Hamming weight $|\mathbf{x}|_1$ divided by the dimensionality N, i.e.,

$$p_1^* = |\mathbf{x}|_1/N \approx M/N \tag{1}$$

where M/N is the design target while $|\mathbf{x}|_1/N$ is an approximate realization via the vector $|\mathbf{x}|$. The important properties of the SDR hold when the realized sparsity is an adequate approximation to the target sparsity, which is easily achievable given that large N is used.

The similarity between two sparse representations is estimated by their overlap d of "1" elements and determined as the Hamming weight of elementwise conjunction (AND, denoted by $\wedge$) of two HD vectors

$$\text{sim} = |\mathbf{x} \wedge \mathbf{y}|_1. \tag{2}$$

It is equivalent to the dot product of $\mathbf{x}$ and $\mathbf{y}$. The result d can then be normalized, e.g., to the Hamming weight of one of the compared HD vectors. Note that due to the asymmetric treatment of "0" and "1" elements in SDR, the geometric vectors by this definition have origin 0^N and are, therefore, all in the same hyperquadrant. This means that the similarities between them must be nonnegative. On average, unrelated vectors will be only slightly similar to each other. In particular, the average value of the dot product for two random binary

HD vectors equals $N(M/N)^2 = M^2/N$. No vector can ever have negative similarity to another.

Atomic SDR vectors, i.e., vectors representing unrelated, unstructured entities, are generated randomly and independently. Several atomic vectors $\mathbf{x}_i$ can be combined together (superimposed or bundled)[2] by elementwise disjunction (OR, denoted by $\vee$). This operation preserves similarity to its components (unstructured similarity). The density p_1 of the resultant HD vector increases with the number of atomic vectors in the superposition. It is often convenient to maintain the density of the result to be equal to the density of the individual components. The density of the resultant vector can be controlled through the context-dependent thinning (CDT) [22] procedure. While there are several options for implementing CDT here the additive CDT procedure [22] is used.

The procedure takes SDR vector $\mathbf{Z}$ as an input. $\mathbf{Z}$ is a superposition of G vectors computed by their elementwise disjunction

$$\mathbf{Z} = \vee_{i=1}^{G} \mathbf{x}_i. \tag{3}$$

Vector $\mathbf{Z}$ is thinned as follows:

$$\langle \mathbf{Z} \rangle = \vee_{k=1}^{T} (\mathbf{Z} \wedge \rho_k(\mathbf{Z})) = \mathbf{Z} \wedge \left(\vee_{k=1}^{T} \rho_k(\mathbf{Z}) \right) \tag{4}$$

where $\rho_k(\mathbf{Z})$ denotes kth permutation of elements of $\mathbf{Z}$; each kth permutation must be fixed, random, and independent. In [13], it is mentioned that a single random permutation can be recursively applied T times. In practical implementations, cyclic shift operations can be applied using randomly chosen (and then fixed) number of unit shifts for each k. Number of permutations T controls the density of the thinned HD vector. The equations describing how the density of the thinned HD vector depends on the choice of p_1, G, and T are provided in Appendix A. The CDT procedure when $N = 10$, $G = 2$, and $T = 1$ is exemplified in Fig. 1. The CDT procedure can be thought of [22] as a similarity preserving (actually, similarity transforming, unlike, e.g., random sampling), hash function of $\mathbf{Z}$ that controls the sparsity of the result and also performs binding of $\mathbf{x}_i$ (unlike random sampling). More details can be found in [28].

Because the vectors belong to one hyperquadrant, as the density of a superposition vector increases it becomes more similar to its component vectors and similar to more non-component vectors. The distribution of similarity of random pairs of vectors is very sharply peaked just above zero. The practical impact of this is that any level of similarity above an extreme (but still low) value on that distribution can be treated as significant similarity. That is, the exact level of similarity is not practically important so long as it is above some low threshold between dissimilarity and similarity.

The recovery of atomic vectors from their thinned superposition is performed by the search of the atomic vectors most similar to the thinned HD vector (all atomic vectors are stored in the item memory). Note that there are several possible ways of organizing the item memory but this paper does not explore particular implementations. There is a limit on the number of reliably recoverable atomic vectors from their superposition, which is regarded as the capacity of the HD representation. The capacity of SDRs after the CDT procedure is discussed in Section VII. It is worth noting that this is the capacity of the superposition, which is not necessarily being used as a system memory, but more likely as a working storage of currently active item representations. Item memories can have much higher capacity. The capacity of the superposition is defined in terms of the ability to discriminate simultaneously, randomly generated, unrelated entities. This operationalization of capacity may not apply in other circumstances, e.g., when the entities are similar or when groups of entities are related via the item memory.

[2]In the context of VSA term "bundling" is often used when referring to the operation of superposition. The algebra on VSA includes other operations, e.g., binding, permutation [19], [27]. Since we do not use them in this paper, we omit their definitions and properties for both sparse and DDRs.

B. Computing With Dense Hyperdimensional Distributed Representations

Kanerva [1] proposed the use of dense vectors comprising $N = 10\,000$ binary elements. The values of each bit of an HD vector are independent and equally probable, hence they are called dense distributed representations (DDRs). Similarity between two binary DDR-vectors is characterized by Hamming distance, which (for two vectors) measures the number of elements in which they differ

$$\mathrm{dist}_{\mathrm{H}}(\mathbf{x}, \mathbf{y}) = \frac{1}{N} \|\mathbf{x} \oplus \mathbf{y}\|_1 = \frac{1}{N} \sum_{i=1}^{N} x_i \oplus y_i \tag{5}$$

where x_i and y_i are values ith element of vectors $\mathbf{x}$ and $\mathbf{y}$ of dimension N, $\oplus$ denotes the elementwise XOR operation.

In very high dimensions, Hamming distances (normalized by the dimensionality N) between any arbitrary chosen HD vector and all other vectors in the HD space are concentrated around 0.5. Interested readers are referred to [1] and [29] for comprehensive analysis of probabilistic properties of the HD representation space. Note that in the case of DDR "0" and "1" elements are treated symmetrically. This is closely related to the bipolar representation. If the origin of the geometric vectors is taken as 0.5^N rather than as 0^N then the geometric vectors can be interpreted as filling the whole hyperspace rather than being restricted to one hyperquadrant and the Hamming distance (5) is a linear rescaling of the dot product of the vectors (which can be interpreted as the cosine of the angle between the vectors).

Atomic DDR vectors are generated randomly and independently. Similar to the SDR case random vectors can be obtained from any other random vector by the cyclic shift operation. Hamming distance between such DDR vectors will be approximately 0.5. Several atomic vectors $\mathbf{x}_i$ can be bundled together. The simplest bundling operation is elementwise summation. When using elementwise summation, the vector space is no longer binary. But from the implementation point of view, it can be practical to restrict the resultant HD vector to binary elements, because the magnitudes of the elements can be thought of as the least significant digits of the representation and most of the information on the direction of each vector is conveyed by the zero/nonzero distinction, which can be thought of as the most significant digits.

Bundling with dense binary HD vectors is, therefore, implemented with elementwise majority rule operation. An elementwise majority rule of G vectors results in 0 when $G/2$ or more arguments are 0 and 1 otherwise. The number of vectors involved into majority rule must be odd. In the case of even number of component vectors a tie-breaking vector is randomly generated and added to the superpostion. Formally, when bundling G vectors, the value H_j in position j of the resultant HD vector $\mathbf{H}$ is determined as follows:

$$H_j = \begin{cases} 1 & \text{when } \sum_{i=1}^{G} x_{ij} > G/2 \\ 0 & \text{when } \sum_{i=1}^{G} x_{ij} < G/2 \\ B(1, 0.5) & \text{when } \sum_{i=1}^{G} x_{ij} = G/2 \end{cases} \tag{6}$$

where x_{ij} is the value of jth element of ith component vector $\mathbf{x}_i$, and $B(1, 0.5)$ denotes a draw of one value from the binomial distribution, which is used for the tie-breaking. The majority rule operation is further denoted as $[\mathbf{a}+\mathbf{b}+\mathbf{c}]$.

The result is a DDR vector with the same density, i.e., the number of "1" elements is approximately equal to the number of "0" elements. The result is similar to all vectors included in the sum. For comprehensive analysis of similarity properties of the majority rule in the context of supervised learning the interested reader is referred to [30].

Similar to SDR, the recovery of atomic vectors from their bundle is performed by the search of the most similar vectors written in the item memory. There is also a limit on the number of recoverable atomic vectors from the bundle, i.e., the capacity. The capacity of dense distributed representations is discussed in Section VII.

C. Computing With Bipolar Hyperdimensional Distributed Representations

The previous sections introduced sparse and dense HD representations using binary vectors, i.e., where each vector's component is encoded as either "0" and "1." The description can naturally be extended to the case of bipolar representations, i.e., where each vector's component is encoded as "-1" and "$+1$." This definition is sometimes more convenient for purely computational reasons. Bipolar HD representations, however, possess a set of distinctive properties. The distance metric is dot product as in the case of SDR. The bundling operation is implemented by elementwise summation. When using elementwise summation, the vector space is no longer bipolar, the result of summation of each bit takes an integer value. When the vector elements take integer bipolar values they can be separated into the sign and magnitude of each value. The signs of the N elements uniquely identify one of 2^N hyperquadrants. The magnitudes of the vector elements determine the direction of the geometric vector within the selected hyperquadrant. Given that the dynamics of HD computing systems are driven by the angles between geometric vectors it can be seen that the dynamics are primarily driven by the signs of the elements rather than their magnitudes. Therefore, in practical implementations, it makes sense to restrict the bit value by a certain threshold. The restriction can result either in a bipolar-valued bits or bounded (clipped) integers. Note that as the elements of unrelated vectors are independent, the sum of some number of values at any element is the sum of G $\{-1, +1\}$ values which is approximately normally distributed with zero mean. Thus, the sums lie with high probability in a bounded interval centered on zero. This makes it possible to choose a clipping limit which is very rarely exceeded. More details on bipolar representations can be found in [18] and [19].

D. Constructing Hyperdimensional Representation of a Pattern

Let us define a class prototype as a bundle of HD encoded pattern exemplars ($\mathbf{H}_i$) belonging to this class. Now consider distributed representation of the pattern itself. It is constructed in a similar manner. Suppose a pattern consists of G features. The value of each feature is mapped to a distributed representation either reflecting the distance between the feature values in the original representation space or to purely dissimilar HD vectors. The tradeoffs of the different mapping approaches are described in the next section. Note that the distributed representations for different features are chosen dissimilar. Note that in the considered HD spaces randomly generated vectors will be dissimilar with overwhelmingly high probability. Thus, after mapping for a given pattern there are G HD vectors $\mathbf{x}_i, i = 1..G$ dissimilar to each other, i.e., normalized (by N) dot product between them is approximately p_1^2 in the case of unipolar representation and zero for bipolar representation. The HD representation of the pattern is constructed by combining G HD vectors into a single representation $\mathbf{H}$. This is done through the bundling operation. Note that the realization of bundling depends on the type of representations. In the case of bipolar HD vectors bundling is implemented as elementwise summation of components and the resultant HD vector is

$$\mathbf{H}_p = \sum_{i=1}^{G} \mathbf{x}_i. \tag{7}$$

When using binary dense representations, the resultant HD vector is formed by majority rule operation as

$$\mathbf{H}_b = \left[\sum_{i=1}^{G} \mathbf{x}_i\right]. \tag{8}$$

Note that the result of (7) can also be restricted to bipolar values when applying majority rule with the threshold equal to zero.

Finally, the CDT procedure (4) is used when forming the resultant HD vector with binary sparse representations

$$\mathbf{H}_s = \left\langle \vee_{i=1}^{G} \mathbf{x}_i \right\rangle. \tag{9}$$

III. Related Work

Some of the aforementioned HD representations were used for modeling of human's long-term associative memory (AM) [31], [32]. The two major principles of distributed associative memories are aggregation of similar stimuli based on statistical similarity of their encoded representations and reducing the overlap between the representations by storing them sparsely over a huge memory space. AMs were extensively applied to pattern recognition [33], as the model for implementing AM in the context of cognitive computing architectures [13]. Abbott *et al.* [34] has demonstrated their suitability for approximating Bayesian inference. One of the major challenges attributed to HD distributed AMs is the encoding problem, i.e., the problem of how to encode stimuli into the distributed representation [29].

VSAs are a family of bio-inspired representations of structured knowledge and related operations. Their development was stimulated by studies on brain activity that showed that the processing of even simple mental events involves simultaneous activity in many dispersed neurons [1]. Information in VSAs is similarly represented in a distributed fashion: a single concept is associated with a pattern of activation of many neurons. This is achieved by using HD vectors of very large dimensions (more than 1000 bits). Several different types of VSAs have been introduced, each using different representations, e.g., [1], [17], [18], [35]. Binary distributed representations enable a hardware-friendly implementation of HD computing machines. However, the HD mapping operations on these random dense HD vectors increases the power consumption: binding with XOR gates imposes a large amount of switching activity, and bundling requires dedicated integer counter and threshold unit per each dimension of HD vector. On the other hand, the sparse binary representation with simple and low-cost mapping operations can lower the switching activity and power consumption by maintaining the sparsity while exhibiting almost the same capacity of dense alternatives.

A simple mapping of patterns from heterogeneous sensory inputs into binary dense HD vectors [36] was described in [30]. The modified version of the mapping into sparse HD vectors was proposed in [37]. Conceptually, the methods first map a value of each feature of a pattern into an HD vector. These HD vectors are then bundled into the resultant HD vector, which represents the pattern in the high-dimensional space. Note that these methods map features of a single object. In HD computing, it is also possible create a distributed representation of a set or a multiset of objects. In this case, the additional operation of binding should be used for encoding features of the particular object. The binding operation, however, addresses a diferent class of problems connected to analogical reasoning, which is outside the scope for this paper.

Methods of formation of real-valued distributed representations using random projections are considered in [38] and [39]. These methods can be modified to produce dense and sparse binary distributed representations [40]–[43]. Other methods of similarity preserving hashing, including those producing sparse binary representations are also surveyed in [43].

Recent usages of VSAs for solving classification tasks in different domains (e.g., [4], [14], [15], [23], [44], [45]) report on a par or even better performance with the state-of-the-art machine learning methods. In particular, in [23], VSAs were applied to the task of hand gesture recognition using electromyography (EMG) signals. It was shown that HD computing with dense bipolar HD vectors ($p_{+1} \approx p_{-1} \approx 0.5$) surpasses the-state-of-the-art classifier [46] based on support vector machines (SVMs) both in terms of the average recognition accuracy and the required amount of training data.

HD Computing in the Scope of Machine Learning on Large Data Sets and Artificial Neural Networks

The strength of HD computing in selected prediction and classification tasks on large data sets were demonstrated in [47] and [4]. The first work used HD computing for fusing data streams from multiples sources and predicting the behavior of users of mobile devices. The HD-based model improved the prediction accuracy by 4% in comparison to a mixed-order Markov model. The classification of texts based on their language in the second work demonstrated 95% accuracy with a gain in power efficiency of computations compared to the traditional n-gram based models.

On the other hand, HD computing usually meets criticism from the adepts of the traditional machine learning methods mainly due to its manual engineering of encodings, tailored to specific tasks and the absence of conventional optimization procedures refining HD representations. The major implication of this observation is that the accuracy of HD-based classification methods is not comparable with the state-of-the-art deep learning artificial neural networks (ANNs) which can learn representations of raw data (e.g., an image) with multiple levels of abstraction.

Despite this, several recent works explain functionality of deep neural architectures within the framework of HD computing. In [20], functionality of the pooling layer in deep convolutional ANNs is linked to a positional binding natively implementable with VSA. The work in [48] elaborates on the functionality of binarized ANNs [49] in terms of HD geometry. In particular, it was shown that binarized ANNs work because of the properties of binary high-dimensional spaces. The binarized vectors of the convolutional layers approximately preserve similarity to the original continuous vectors the dot products are approximately preserved as well.

With respect to approaches other than deep learning ANNs, the recent work in [50] explores direct similarities between the main operations of the HD computing and the operations of a special kind of recurrent ANNs—echo state networks (ESNs) [51]. In particular, it was shown that the entire dynamics of the reservoir can be implemented by HD arithmetic on dissimilar vectors while matching the accuracy of the traditional ESN.

We regard these trends as definite indication of at least an important complementarity of HD computing to ANNs. Whether HD computing will become a performance-challenging competitor to ANNs depends on finding solutions to yet unsolved problems. One of the most important ones is defining optimization procedures directly in HD spaces or finding training methods that do not rely on optimization.

IV. Pattern Recognition With HD Computing: Problem Statement, Showcases, and Outline of Approach

Classification is one of the most common tasks in machine learning. The task is to correctly associate a new pattern with one of the discrete classes where the number of classes is finite. In order to perform the classification, models of classes are needed. The models are constructed from a training data set consisting of patterns labeled with their classes. Traditional machine learning approach include instance-based methods (k-NN) or model-based (linear and nonlinear classifiers such as SVM or artificial neural nets). The instance-based methods suffer obviously from the scalability problems associated with the memory size and the search times as the number of the training examples grow. As for the model-based methods, they all rely on complex optimization routines aimed at minimizing the number of incorrect predictions on the test data set. The complex optimization is usually restricted to the training phase and prediction is typically some feed-forward calculation that is quite fast.

Distributed representation of features in combination with SVMs is reported in [23], [24], and [52]. Other mentions of classification with distributed representations are given in [13], [39], and [43]. However, training classifiers by optimization is time consuming.

This paper presents the usage of distributed representations for classification tasks based on the native properties of HD computing without involving complex optimization procedures. This approach results in construction of a single HD vector representing one class (further on also called centroid or class prototype).[3] A fixed amount of memory is required per class prototype to represent each prototype. The amount of memory required per prototype in the recognition mechanism may be different. It might be linear in the number of prototypes in a matrix memory (effectively a lookup table) or less than linear in an autoassociatve memory. Also, a similar approach [54] can be used in order to accelerate the search among a collection of high-dimensional vectors. Note that this model can be generalized to multiple prototypes per class, e.g., when relatively dissimilar inputs must be mapped to the same class, but the article considers only the case of a single prototype.

This section describes the approach for constructing HD distributed class prototypes as well as discusses tradeoffs of choosing its major hyperparameters: the dimensionality, density, and the mapping methods.

The discussion is centered around two showcases: a synthetically constructed classification scenario and the real-world classification task.[4] Both showcases are chosen for illustrative demonstration of the major aspects of the approach. The showcases are outlined in the following sections.

A. Synthetic Showcase: Classification of 7 × 5 Pixels Characters

In the synthetic classification task, a set of patterns consisting of black and white images of letters (7 by 5 pixels) of the Roman alphabet is considered [30, Fig. 5]. In the recall phase, images of the same letters distorted with different levels of random distortions by bit flipping (from 1 bit corresponding to a distortion of 2.9%of the pattern's size to 5 bits equivalent to 14.3% distortion). An example of a noisy input can be found in [30, Fig. 7].

B. Real-World Showcase: Gesture Recognition

As a practical HD-based classification task, we consider a smart prosthetic application, namely, hand gesture recognition from a stream of electromyography signals. The EMG signal is the superposition of all the action potentials from the brain cortex to the target muscles. It is measured by a couple of metal electrodes, placed on the skin and aligned parallel to the muscle fibers. The maximum amplitude of this signal is 20 mV (-10 to $+10$) depending on the dimension of the muscle fibers, on the distance between the muscle and the electrodes and on the electrode properties. Signals of this kind are also very noisy and difficult to manage even if the maximum bandwidth does not exceed 2 kHz. The main causes are noise from motion artifacts, fiber crosstalk, electrical equipment, and the floating ground of the human body.

The data set used in this paper is based on the collection of the EMG signals of the most common hand gestures used in daily life available in [23]. The selected gestures are: closed hand, open hand, two-finger pinch, and point index collected for five subjects. The classification includes also the rest position of the hand, recorded between two subsequent gestures. Thus, the task is formulated as a classification problem with five possible classes.

The signals are collected from five subjects wearing an elastic strip with the four EMG sensors. The EMG data set is labeled using a threshold assigning the gesture labels to the input EMG data. The collected sequences of sensory signals are composed of 10 repetitions of muscular contraction three seconds each. Between each contraction there are three seconds of rest. The gestures are sampled at 512 Hz. Three repetitions of gestures of each type and the contiguous rest periods were used for training and seven repetitions for testing.

C. Constructing Hyperdimensional Class Prototypes

The class prototype is an HD vector representing the entire class. The number of class prototypes equals the number of classes in the task. The prototype HD vector resembles significant similarity to its exemplars, i.e., to HD vectors in the training data set which belong to that class. The prototype HD vector for a given class is computed out of HD vectors for all presented patterns of this class in the training data set. The HD vectors are combined into the prototype using a specific realization of the bundling operation. The dimensionality of the prototype HD vector is the same as the dimensionality of the HD vectors in the data set.

The realization of the bundling operation depends on the choice of the type of used distributed representation. As outlined in Section II, the simplest realization of the bundling operation is elementwise summation (as proposed in the

[3]The term "class prototype" is actively used in cognitive science and semantic models [53] and refers to a holistic description of the class properties.

[4]MATLAB implementation of the experiments reported in this paper is available online via https://github.com/denkle/Binary-Hyperdimensional-Computing-Trade-offs-in-Choice-of-Density-and-Mapping.git.

Multiply–Add–Permute (MAP) VSA framework [19]). Note that in this case the prototype is literally the sum of the exemplars. So although the prototype is a single vector and treated as a single vector by the implementing hardware, a subsequent operation on the prototype may distribute over addition, in which case applying the distributive operation to the prototype is identical to applying the distributive operation to each of the exemplars and then summing over the results. This means that it is possible to save (potentially great) computational cost because of an effect like parallelism by virtue of the distributivity of the operations. Mathematically, the prototype HD vector for class c_i is constructed as

$$\mathbf{P}_p(c_i) = \sum_{i=1}^{n}(\mathbf{H}_i). \tag{10}$$

In (10), n is the number of the exemplars of class c_i in the training data set and $\mathbf{H}_i$ is the HD vector representing the exemplar i. Recall that when using elementwise summation, the vector space is no longer binary (or bipolar). From the implementation point of view, it can be practical to restrict prototype HD vectors to binary elements. When operating with dense HD vectors, majority rule is applied to the result of the elementwise summation. Then, the prototype HD vector for class c_i is formed as

$$\mathbf{P}_b(c_i) = \left[\sum_{i=1}^{n}(\mathbf{H}_i)\right]. \tag{11}$$

The square brackets in (11) indicate majority rule. The resultant HD vector $\mathbf{P}_b(c_i)$ representing the prototype is a binary HD vector. If component HD vectors $\mathbf{H}_i$ are dense with $p_1 \approx p_0$ then the prototype HD vector $\mathbf{P}_b(c_i)$ is also dense with $p_1 \approx p_0$ otherwise the density of ones is decreasing.

The majority rule cannot be applied to form prototypes with sparse HD vectors because the resultant HD vector will consist of all or almost all zeros. Therefore, for sparse HD vectors, the bundling is realized through the thresholded summation. The threshold controls the density of nonzero elements in the resultant HD vector. It, for example, can be set to ensure the required number of activated elements in the resultant prototype HD vector (see [55]).

D. Classification Using Class Prototypes

The classification using class prototypes is straightforward. An unseen pattern from the test data set is mapped to an HD vector using the same method as for the training process. The result of the mapping is used to calculate the value of similarity metric to all class prototypes formed during the training phase. The exemplar is assigned to the class for which similarity between its prototype HD vector and the pattern's HD vector is maximal. Note that this is a computation level description of the idealized classification task. Literal implementation of the task as described above would have a computational cost proportional to the number of class prototypes. Other implementations (e.g., an autoassociative memory) might have lower computational cost but a higher error rate.

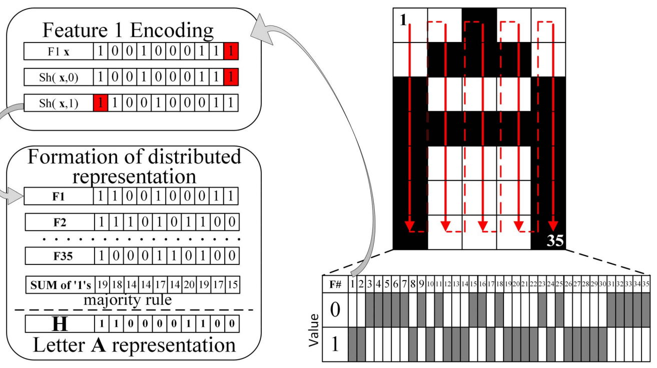


Fig. 2. Example of mapping letter "A" into binary dense representation. Value 0 corresponds to black color, while value 1 corresponds to white color. Gray filling in the table marks values of each pixel. For simplicity of presentation, the mapping is exemplified on 10-dimensional vectors.

V. Mapping Techniques

There are several approaches to mapping of features' values from the original representation space into the HD representation space. This section presents two classes of mapping: 1) the mapping by the orthogonal distributed representations and 2) the distance preserving mapping.

A. Mapping of a Pattern Using Dissimilar Distributed Representations of Feature Values

In the case when a feature can be described by a finite alphabet of independent symbols, the mapping to HD vectors is trivial. Each symbol is simply assigned a unique HD vector, which is randomly chosen initially, but fixed over the life of the system. This procedure is called orthogonal mapping. It can be implemented in a memory efficient way using only one HD vector per feature and the cyclic shift operation as a special case of permutation (other examples of permutation usage are considered in [27] and [56]) on it to derive a unique HD vector for the particular level (a discrete value of a feature). For illustration of orthogonal mapping consider encoding of image "A" from the synthetic showcase described above. It is demonstrated in Fig. 2. In this example, each pixel is a feature, thus we have ($7 \times 5 = 35$) features. Consider the case of representing a pattern using binary dense HD vectors with the orthogonal mapping of values. The mapping procedure consists of the following steps.

1) Initialization of the mapping.
    a) Set the dimensionality of the HD vectors.
    b) Set the number of features. The image of a letter consists of 35 pixels. Each pixel is treated as a feature.
    c) For every feature i generate one binary dense random HD vector $\mathbf{x}_i$, that is 35 HD vectors are generated in total. Note that generated HD vectors for different features are dissimilar.
2) For each feature i shift its $\mathbf{x}_i$ by the value of the pixel (0-bit shift for black and 1-bit shift for white).
3) Form the distributed representation of the letter using shifted $\mathbf{x}_i$ vectors according to (8).

4) Store the letter's representation in the list of memorized patterns.

Note that the mapping example above is just one way to encode the feature values, not the only way. This method means that the representations for different features are unrelated, but the zero and one values for every feature are systematically related to each other by cyclic shifts—so the HD vector encodes the pixel ID, and the shift encodes the pixel value. For an alternative mapping, it would be possible to choose a random HD vector for every pixel-value pair. In this case there would be no systematic relationship between the zero and one values of the same pixel. Also note that because the mapping treats the pixel IDs as completely independent it does not capture the spatial similarity of neighboring pixels.

B. Mapping of a Pattern Using Distance-Preserving Distributed Representations of Feature Values

The task of mapping real numbers into HD vectors is of practical importance since features of this type are common in practical pattern recognition tasks. This section describes three methods of such mapping.

Consider a feature whose values are real numbers between 0 and m. The task is to represent the current value of the feature as an HD vector. All considered methods require a finite amount of values, therefore, real numbers are first quantized using a quantization scheme with the fixed number of levels (values). For example, the current value of the feature can be rounded to the closest integer. Then the considered feature is described by $m+1$ unique levels. Next, each unique level is associated with the corresponding HD vector.

1) Mapping Preserving "Linear Similarity" of Values: The "linear mapping" [23], [57] (this name does not imply mapping by matrix multiplication) preserves "linear similarity" between HD vectors corresponding to different levels of a feature. The method starts by initializing an HD vector for the first level. The HD vector for the next level is formed using the HD vector for the previous level by inverting several random "1" and "0" (active and inactive) elements such that the total number of activated elements stays the same, but some of their positions are changed. This step is repeated until HD vectors for all levels are generated. It is important that during the iterations only elements activated at the initialization step can be inverted and inactive elements to be inverted should be selected from the pool of vector elements unprocessed so far (sampling without replacement). It guarantees that the similarity between HD vectors decreases linearly due to the way the mapping scheme is designed. The number of elements inverted at each iteration of the method is a tunable parameter. In the simplest case, if the HD vector for the first level and the HD vector for the last level should be dissimilar (i.e., their normalized dot product equals p_1^2 in expectation) then for $m+1$ levels at each iteration $Np_1(1-p_1)/m$ activated elements (and the same number of inactive elements) should be inverted. If we do not want to repeat the HD generation procedure during feature encoding, all $m+1$ HD vectors should be stored in a memory which requires more resources than in the orthogonal mapping where only the seed HD vector can be stored.

2) Approximate "Linear Mapping": Approximate linear mapping [58] does not guarantee ideal linear characteristic of the mapping, but the overall decay of similarity between feature levels will be approximately linear. In contrast to the ideal linear mapping values of similarity between neighboring levels could slightly vary. Approximate linear mapping, however, requires storage of only two random HD vectors: one for the first level and one for the last level. HD vectors for intermediate levels are generated by the concatenation of parts of the two HD vectors. The lengths of the concatenated parts are defined by the value of the intermediate level. Alternatively, one could form the weighted sum of the two end points where the weighting reflects the position of the level between the end points. The sum is then binarized by choosing from {0, 1} with the probability proportional to each weighted sum element. If further bundling operations are expected, a fixed random permutation should be applied for each feature HD vector in order to form dissimilar vectors even if two features have the same or similar values. So an ideal linear mapping can be easily obtained as follows: take an HD vector with M first and M last elements assigned to "1" ($2M \leq N$). To encode level i, take $(m-i)M/m$ successive "1" from the "first M" and iM/m successive "1" from the "last M"; to get HD vector, apply a random permutation fixed for the particular feature (actually storing $2M$ numbers for a feature is required instead of full permutation matrix). In the case of the alternative method above, it is possible to randomly choose the ith element of intermediate value HD vector from the ith elements of the two endpoint vectors, with probability of choosing each source vector proportional to the position of the level between the two end points.

It is worth noting that approximate linear mapping is cheaper than the ideal one as it requires storage of only two random HD vectors. As it will be shown below in Section VI-B, the usage of approximate linear mapping in a classification task provides results comparable to ideal linear mapping. However, for regression problems (see, [50]) where prediction error is a key issue, ideal linear mapping would be a preferable option due to its exact linear characteristic.

3) Approximate "Nonlinear Mapping": The decreasing similarity between HD vectors is not necessarily a linear function. The approximate nonlinear mapping method described in [58] starts by initializing an HD vector for the first level. Next, in the iterative manner, it forms HD vectors for all other levels. In the version we use here, HD vector for the next level is created via deactivating several activated elements from the HD vector for the previous level and then activating some number of elements so that the density of HD vectors is maintained on approximately constant level. Unlike "linear mapping," multiple processing of the same vector elements is allowed (sampling with replacement). The similarity between HD vectors decays (approximately) as a bell-like curve.

It is worth noting that for all distance-preserving mappings it is possible to have the situation where levels more than some threshold apart are maximally dissimilar whereas levels closer together are more similar. This might be useful for representing a variable where nearby levels are able to be

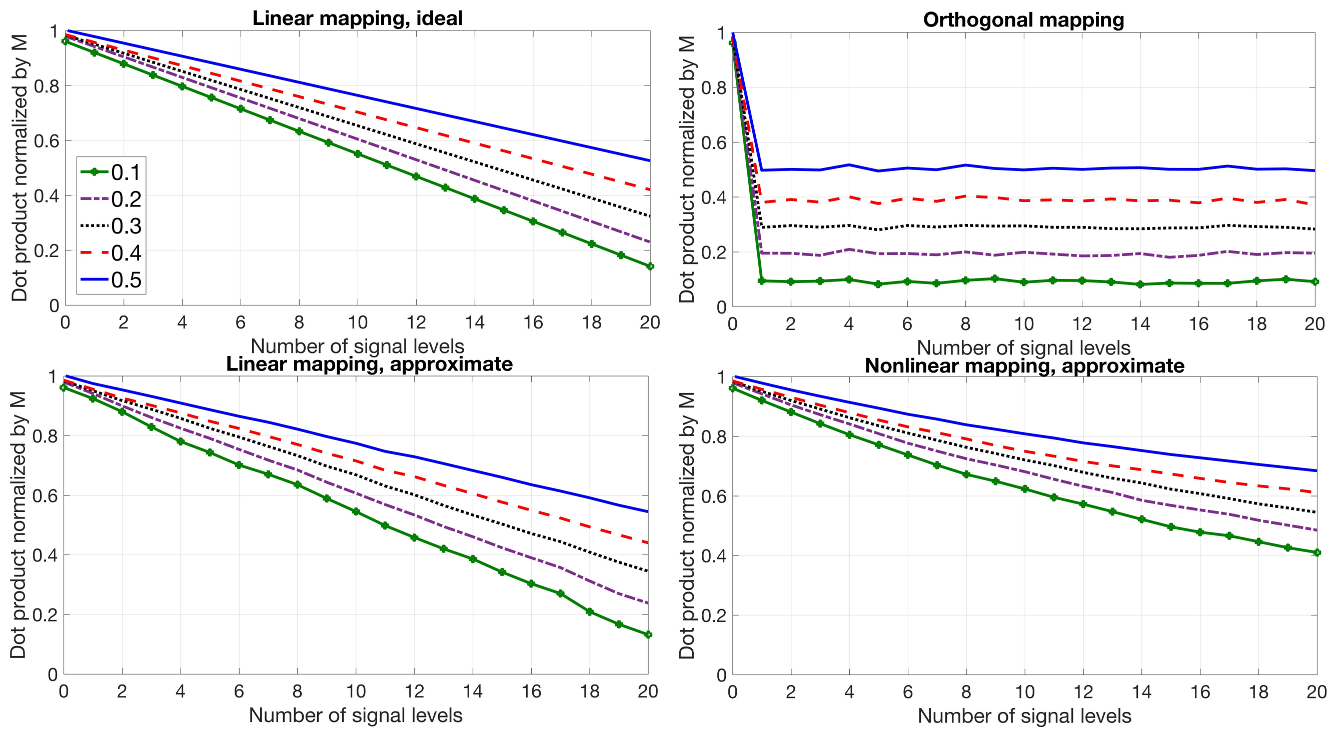


Fig. 3. Four schemes for mapping real numbers into HD vectors. During the experiments, N was fixed to $N = 10\,000$. Density of ones p_1 varied between 0.1 and 0.5 with step 0.1. Note that for all four panels the similarity value is the normalized dot product of the representation of each signal level with the representation for 0 mV. Also, the normalized dot products for the nonidentical pairs in the "orthogonal mapping" panel are the dot products corresponding to maximum dissimilarity (vectors are nearly orthogonal) and are, therefore, the values that the linear mappings tend to (by design).

judged by similarity, but all levels more than a certain distance apart are treated as maximally dissimilar.

For the implementation details of the presented mappings interested readers are referred to the simulation code used for Fig. 3. It illustrates similarity decay for all mapping methods when the feature is a voltage signal level from the EMG sensors showcase. The value range is between 0 and 20 mV (denoted by number of signal levels in Fig. 3). Five different densities of HD vectors ranging from 0.1 to 0.5 are considered. The signal is quantized into 21 levels (integers from 0 to 20). This makes dissimilarity linear in voltage differences. In some other situations (e.g., with wide input ranges), it may be better to have resolution proportional to value. In this case one could take the logarithm of the input before discretizing it. The similarity decay is characterized by the dot product between the HD vector for the first level (0 mV) and all 21 HD vectors. Dot products were normalized by the number of ones (Np_1). For all mapping methods normalized dot product for the first level is 1 because in this case the dot product is calculated for the same HD vectors. Normalized dot products for other levels in the orthogonal mapping are the same and approximately equals p_1. All other mappings demonstrate smooth decay. The approximate linear mapping behaves very similar to the ideal linear mapping. However, the curves for the ideal linear mapping are literally straight lines while the corresponding curves for the approximate linear mapping have local deviations. It is especially clear in the case of a low density of ones ($p_1 = 0.1$). The normalized dot products between the first and the last levels for both mappings are approximately p_1. It is also clear that for the same number of inverted elements dot product for the nonlinear mapping decays slower than for the linear mappings.

VI. Tradeoffs in Choice of Density and Mapping Characteristics

This section evaluates the performance of binary HD vectors in classification tasks. First, the effect of density on the classification accuracy is demonstrated for the synthetic showcase scenario of black and white 7×5 images. Second, the effects of different types of mappings are demonstrated on the example of EMG-based gesture recognition.

A. *Effect of Density on the Accuracy of Classification*

This section evaluates the effect of dense versus sparse binary distributed representations on the classification accuracy. It is evaluated in two major recall modes: the one-shot learning and the supervised learning (see sections later). The first recall mode is studied for varying density of ones in HD vectors for several dimensionalities. Next, more detailed results of experiments are presented for sparse HD vectors for three sets of parameters: $N = 100\,000$ and $M = 1000$ ($p(1) = 0.01$); $N = 10\,000$ and $M = 100$ ($p(1) = 0.01$); $N = 2048$ and $M = 40$ ($p(1) = 0.02$). $N = 100\,000$ and $M = 1000$ are mentioned as a typical values for sparse representations in [22]. The dimensionality $N = 10\,000$ is chosen to match the dimensionality of the dense HD vectors. The third set of parameters $N = 2048$ and $M = 40$ is motivated by the usage of

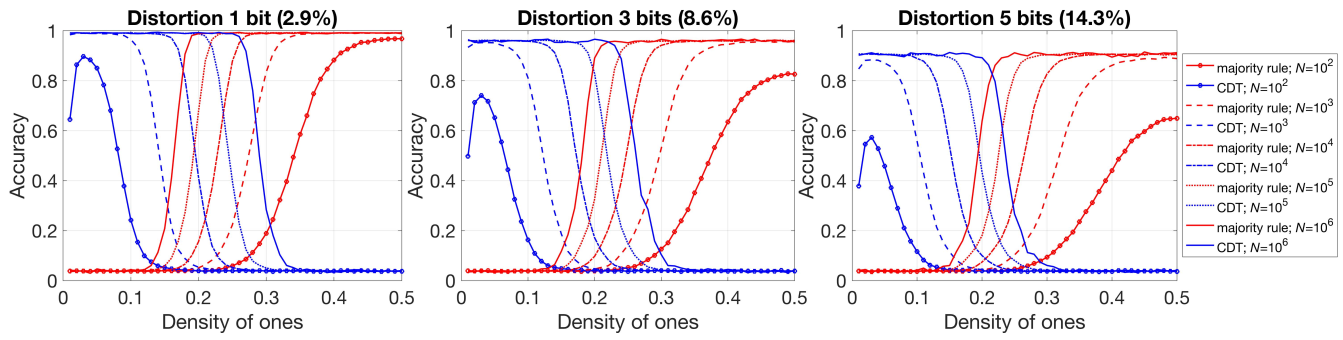


Fig. 4. Average recall accuracy of black and white images of letters using the majority rule and the CDT procedure against varying density of ones in HD vectors for three distortions levels: 1 bit (2.9%), 3 bits (8.6%), and 5 bits (14.3%). Dimensionality of HD vectors was set to 100, 1000, 10 000, 100 000, and 1 000 000 elements. The results were averaged across 1000 runs. Due to a large number of simulations (10^3) confidence intervals for 95% are narrow and indistinguishable from the mean values.

SDR with these parameters in Hierarchical Temporal Memory architecture [59]. As in [30] for the evaluation of dense HD vectors with dimensionality $N = 10\,000$ elements (bits) [1] were used.

1) Best Match Recall Under One-Shot Learning: In the learning phase, a set of *noise-free* images of letters (see [30]) was represented by HD vectors with different densities. All 26 letters were encoded and stored in the same way as exemplified in Section V-A. Thus, at the end of the letters' mapping procedure 26 HD vectors are created and stored in the list of memorized patterns. In the case of the original low-dimensional representations, twenty six 35-bits vectors corresponding to activation of pixels in images of letters were stored in a list.

In the recall phase images of the same letters distorted with three different levels of random distortions, 1 bit corresponding to a distortion of 2.9% of the pattern's size, 3 bits equivalent to 8.6% distortion, and 5 bits (14.3% distortion) were presented to the methods for the recall. An example of a noisy input was presented in [30, Fig. 7]. In the case of the dense HD vectors and the original low-dimensional representations, the pattern with the lowest normalized Hamming distance to the representation of the presented distorted pattern was returned as the output. In the case of the sparse HD vectors the pattern with the highest dot product to the representation of the presented distorted pattern was returned as the output.

Fig. 4 presents the average recall accuracy against varying density of ones in HD vectors. The density varied between 0.01 and 0.5 with step 0.01. Dimensionality of HD vectors was changed with factor of 10: 100, 1000, 10 000, 100 000, and 1 000 000. Each panel in the Fig. 4 corresponds to a particular level of distortion. Distributed representations of images were done via either majority rule or CDT procedure. All curves are characteristic in a sense that the CDT procedure shows the best performance for low values of density while the majority rule features the opposite behavior. At the same time, the best performances of both approaches match each other for each distortion level. When the dimensionality of HD vectors is low ($N = 100$) there is a clear peak in accuracy for the CDT procedure. In this case, the peak is due to the fact that at low densities (e.g., $p_1 = 0.01$) HD vectors with almost no ones can be generated. This phenomenon does not appear at high dimensionalities ($N = 10\,000$ and above). Another observation regarding the effect of dimensionality of HD vectors is that there is a large improvement in accuracy when going from $N = 100$ to $N = 1000$ while going from $N = 1000$ to $N = 10\,000$ and above there is only a small improvement for 5 bits distortion. Thus, the dimensionality is important, but only up to a certain point beyond which increased dimensionality does not affect the accuracy.

Fig. 5 presents the recall results for sparse and dense HD vectors and for the reference original low-dimensional representations. To obtain the results 1000 distorted images of each letter for every level of distortion were presented for recall. The charts show the percentage of the correct recall output. Note, that in certain characters the recall is rather inferior to other letters. For example, the recall for character "O" is persistently lower than for other letters. This is due to its similarity to several other characters: "C," "D," "G," and "Q."

The analysis shows that: 1) there is no degradation of accuracy when the test problem is transferred from the original low-dimensional representation into high-dimensional distributed representations and 2) the result holds even for relatively low dimensions of the sparse HD vectors ($N = 2048$ and $M = 40$). The gain from high-dimensional representation becomes evident when distortions are added to the representations. For the case of DDRs it was demonstrated in [30] that there is a threshold on the noise level for the original low-dimensional input patterns beyond which Hamming distance becomes totally unreliable metric for detection of similarity, while distributed representations are robust to substantially larger noise levels. The robustness comes from the usage of random HD vectors in high-dimensional spaces because in contrast to the original low-dimensional representation they require a substantial level of distortion for bringing two distributed representation close to each other.

Also the accuracy of the HD recall is higher for nonbinary input patterns, to which, for example, a third color is added as the background to the letters in our test case. For the sake of space saving the results of the corresponding experiments are omitted in this paper, and interested reader is referred to [30] for the details.

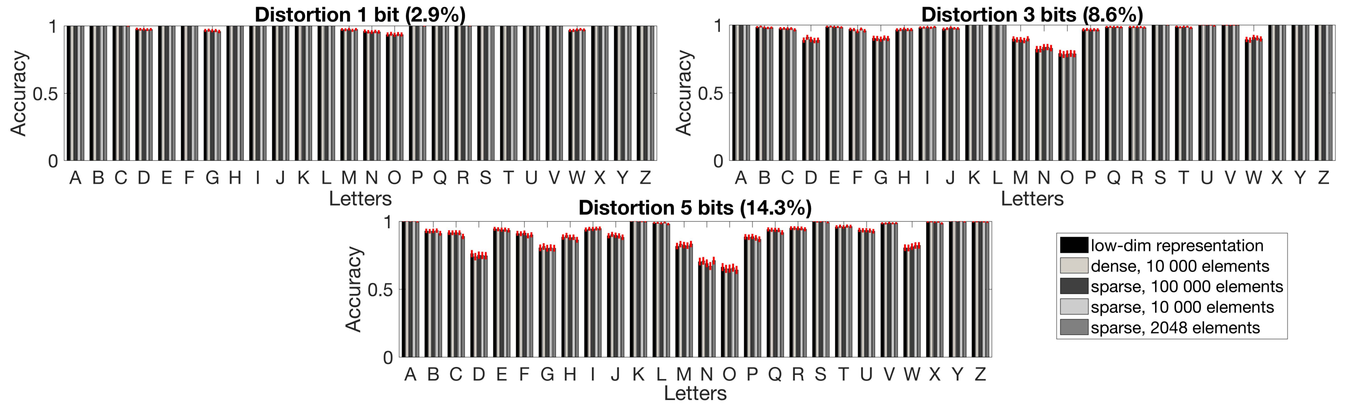


Fig. 5. Test results for black and white images of letters using distortions of 1 bit (2.9%), 3 bits (8.6%), and 5 bits (14.3%). Red bars depict 95% confidence intervals.

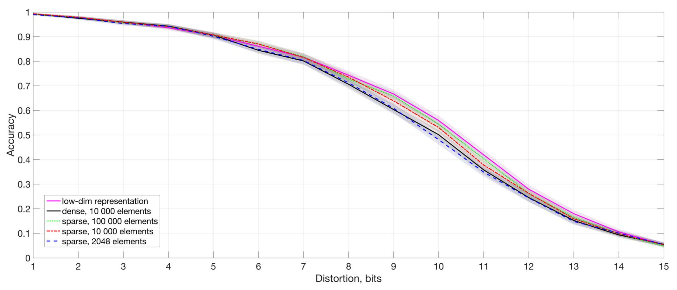


Fig. 6. Accuracy of recall under supervised learning as a function of the distortion level. Shaded areas depict 95% confidence intervals.

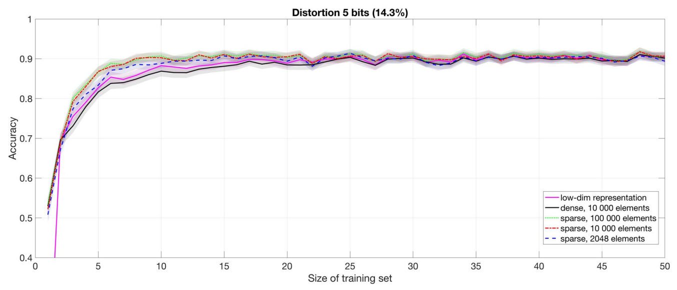


Fig. 7. Accuracy of recall under supervised learning as a function of the number of presented examples for a given level of distortion. Shaded areas depict 95% confidence intervals.

2) Recall Under Supervised Learning: This section presents the results of the comparison of the pattern recognition accuracy under supervised learning. In the context of this experiment, a supervised learning mechanism was implemented using prototype HD vectors. Note that the procedure described here as supervised learning is not exactly the traditional definition of the term which usually involves some kind of optimization of the representations with respect to an outcome measure. This is not the case here. The prototypes were formed by bundling distributed representations of distorted patterns of the same letter into the distributed representation of the letter during the training phase. In other words, in this scenario, each prototype stores a set of noisy letters generated from the same noise-free letter. Thus the prototype has information on the natural variability of the images, whereas the prototypes in the previous scenario with single shot learning do not. In the case of dense representations the majority rule was used while for sparse representations the thresholded summation was used for making the prototypes. After the thresholded summation the density of the prototype equalled the expected density of an HD vector representing a single image.

In the first experiment, we used series of randomly distorted images of letters with different level of distortion between 1 and 15 bits. In the experiments up to 50 original images for each letter and images at every level of distortion were presented for memorizing. For the particular level of distortion, all HD vectors of presented distorted images of the particular letter were combined together to form a single prototype HD vector of that letter. Thus, by the end of the training phase the AM contains 26 HD vectors, each jointly representing all (presented) distorted variants of the particular letter. In the recall phase for each distortion level 100 new distorted images of each letter were used as input. The accuracy was measured as the percentage of the correctly recognized letters averaged over the alphabet.

Fig. 6 illustrates the obtained results: 90% accurate recall was observed when training symbols were distorted by up to 5 bits (14.3%). While the accuracy predictably decreases rapidly with the increase of distortion in the presented images, a reasonable 80% recall accuracy was observed for learning sets with 7-bits distortion (20%). The accuracy curves for all four compared configurations show identical behavior.

Fig. 7 illustrates the convergence of the recall accuracy with the number of distorted original images presented to form the prototypes for the case of 5-bits distortion (14.3%). For larger training sets, the average accuracy in Fig. 7 is approaching the average value in Fig. 6 for 5-bits distortion. Interestingly learning curves for all three configurations of the sparse HD vectors are steeper than the curve for the dense HD vectors. For $N = 2048$ configuration, the curve starts from the lowest accuracy, but at the size of training set more than 5 images it performs as well as other sparse configurations of HD vectors. Thus, for the supervised learning based on the distorted images the sparse HD vectors require less images to achieve their optimal performance. For example, in Fig. 7 all configurations

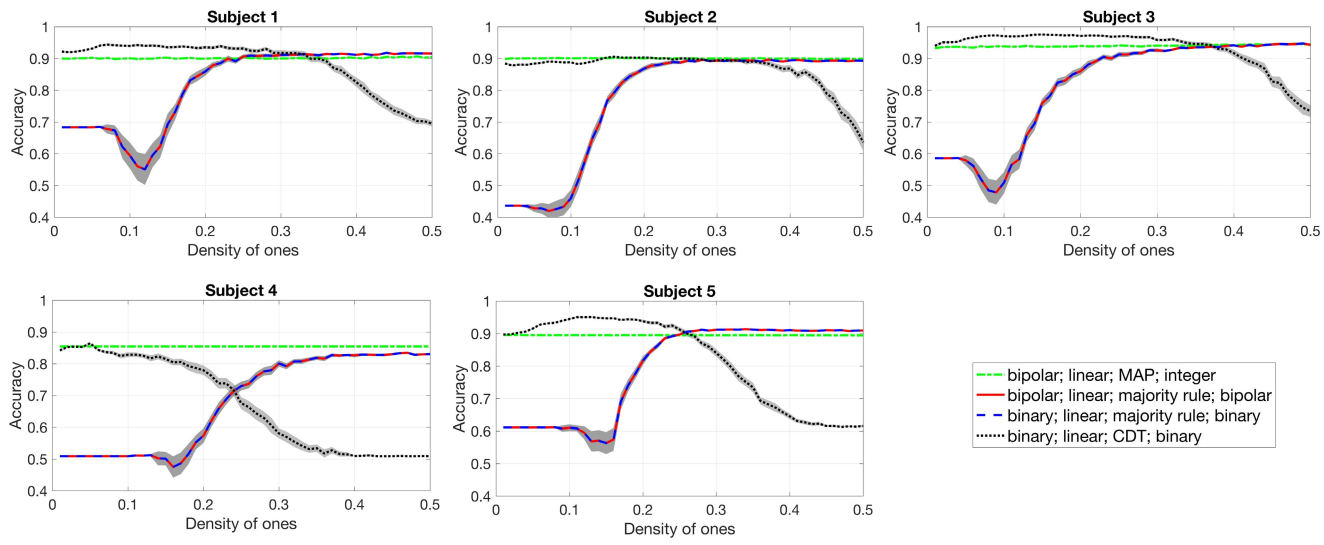


Fig. 8. Average recognition accuracy for five subjects using four different classifiers against varying density of ones in HD vectors. All classifiers use ideal linear mapping of integers to HD vectors. Shaded areas depict 95% confidence intervals. Dimensionality of HD vectors was set to 10 000 elements. The results were averaged across 50 runs.

of sparse HD vectors achieve the recall accuracy around 90% after 15 training images, while the dense HD vectors need 20 images to settle at the same level of the recall accuracy.

B. Effect of Different Mapping Approaches

This section considers the effect of different feature mapping approaches to dense and sparse HD representations. The studies were performed on the hand gesture recognition task.

A gesture was characterized by a pattern of n latest measurements of each sensor, where n is a parameter, which can be tuned individually for each subject. Hence, a gesture is described by a tuple of $4n$ features (real numbers) in the range 0.0–20.0. These numbers were rounded to integers and were used as levels when mapping them to HD vectors. Each of $4n$ integers was assigned with a unique HD vector according to one of the four mapping schemes described in Section V-B. Two types of HD vectors were used: binary and bipolar. For the given pattern HD vectors representing values of features were bundled together using the chosen bundling operation (see Section II-D) to form the resultant HD vector representing the current gesture. During the training phase, for each gesture class prototype HD vector was formed by one of the operations (see Section IV-C). In the operating phase, similarity scores between the HD vector for the unknown gesture and the prototype HD vectors were used to assign the class with the highest similarity. The similarity is measured by cosine similarity for bipolar representations; by Hamming distance for binary dense representations; and by dot-product for binary sparse representations.

Ten different HD classifiers were used in the evaluation. Each classifier is described by three parameters: 1—Type of HD vectors for mapping features can be either binary or bipolar; 2—Type of feature mapping: orthogonal, ideal linear, approximate linear, and approximate nonlinear. Note that the first type represents each level by a random HD vector, while the last three types preserve similarity between the HD vectors representing similar levels. 3—Types of bundling operations used for constructing the HD vector of the pattern and the prototype HD vector of the class. For bipolar representations either elementwise summation (denoted by MAP) or majority rule can be used for both tasks. For binary dense representations majority rule is also used for both tasks. For binary sparse representations the CDT procedure is used to form pattern's HD vector while thresholded summation is used when constructing prototypes. In the experiments the average accuracy of the correct classification of a gesture for each subject was studied against density of ones in HD vectors representing features. In the case of bipolar representations "-1" values were dominating for the low densities of ones.

Fig. 8 illustrates the average recognition accuracy for each subject against varying density of ones in HD vectors for four classifiers in the case of the ideal linear mapping. Note that the classifier with bipolar representations, ideal linear mapping, and elementwise summation as the bundling operation corresponds to the classifier presented in [23] and, therefore, it is used as a benchmark for other classifiers. Its performance does not depend on the density of ones. It is due to the fact that the elementwise summation does not restrict the range of values of the resultant HD vector.

It is, however, opposite for other classifiers. If, instead of the elementwise summation rule, the majority rule with threshold 0 is used with bipolar representations ("bipolar; linear; majority rule; bipolar" in Fig. 8) then the performance of such a classifier varies with the density of ones. Moreover, this classifier behaves in the same way as the binary classifier with majority rule ("binary; linear; majority rule; binary" in Fig. 8) because they are essentially equivalent due to the fact that, in the case of bipolar representations, the majority rule with the threshold 0 produces the same result as the majority rule with the threshold $G/2$ for binary representations (given that ties are broken with the same random sequences as was the

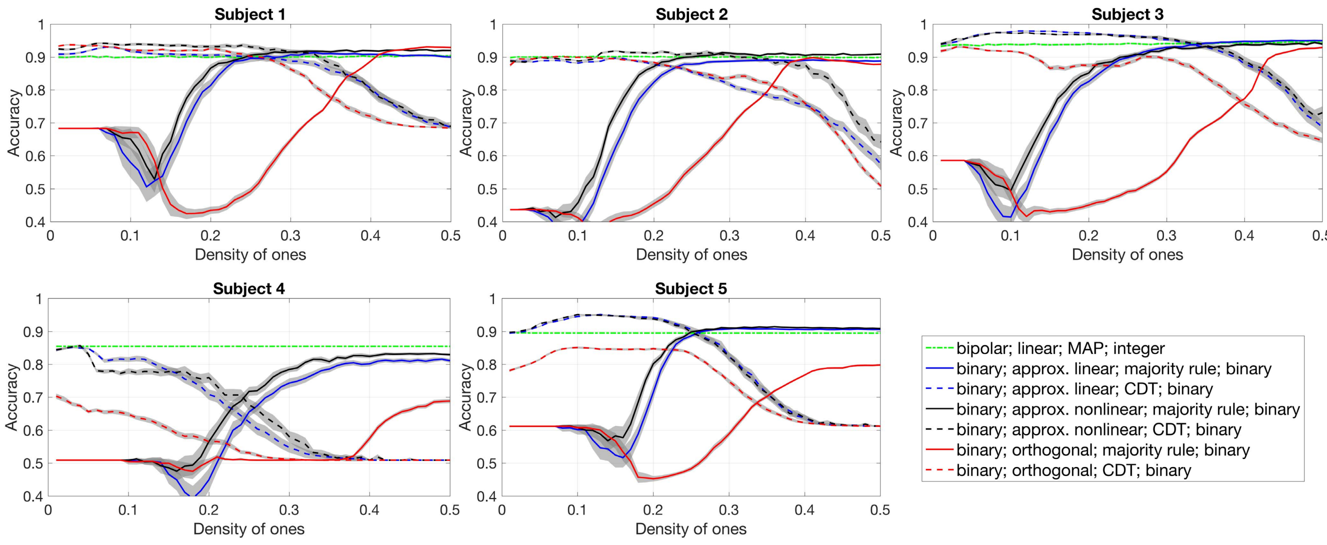


Fig. 9. Average recognition accuracy for five subjects using seven different classifiers against varying density of ones in HD vectors. Bipolar classifier with ideal linear mapping is used as a benchmark. Other six classifiers are using three different mapping schemes for dense and binary sparse representations. Shaded areas depict 95% confidence intervals. Dimensionality of HD vectors was set to 10 000 elements. The results were averaged across 50 runs.

case for the simulations in this section). The variation of the performance against density of ones in HD vectors is explained by the nature of bundling operations. If majority rule is used with binary sparse HD vectors then at some point there will be no elements in resultant HD vector which exceed threshold, hence the resultant HD vector after majority rule consist of all zeros.

Similarly, when the CDT procedure is used on dense HD vectors there is a high chance that the HD vector after OR will consist of all ones so that the thinning step of the procedure cannot decrease density of ones. In this case, the resultant HD vector after the CDT procedure is not more representative than all zeros HD vector. Thus, classifier with these bundling operations show satisfactory performance when operating in the designed regions, i.e., low densities of ones for the CDT procedure and high densities of ones for majority rule.

As a general guiding rule, a classifier with the bundling operation for sparse representations will demonstrate the best performance when the density of ones is small, while a classifier using the bundling operation for dense representations will perform better when the density of ones is closer to 0.5. Results in Fig. 8 confirm this observation: both binary classifiers achieve performance comparable with the benchmark bipolar classifier. The binary dense classifier shows the best performance when $p_1 = 0.5$. But for the binary sparse classifier for each subject there is a value of density of ones when it demonstrates its peak.

Fig. 9 demonstrates the effect of other mappings on the performance. The bipolar classifier in Fig. 9 is used as a benchmark. The overall behavior of classifiers is similar to Fig. 8: sparse classifiers perform better when density of ones is low; for dense classifiers $p_1 = 0.5$ is preferable. Approximately linear classifiers correspond to their ideal versions in Fig. 8. The performance does not suffer if the mapping is done approximately. It is an important observation as the approximate linear mapping has memory efficient implementation. The approximate nonlinear mapping demonstrates slightly better performance for subjects one, two, and four for the dense classifier. In the case of the sparse classifier the performance is higher for subjects one and two. It is concluded that the approximate nonlinear mapping could improve performance but the improvement is not guaranteed. Interesting results are observed for the orthogonal mapping. For first three subjects the accuracy of the classifiers with the orthogonal mapping does not differ much from the accuracies for other types of mapping. But the performance is significantly lower for subjects four and five. It can be explained by large variations of patterns produced by the last two subjects. Thus, in some cases the orthogonal mapping can perform as well as linear or nonlinear mappings, but there are situations when preserving similarity between neighboring levels leads to the increased accuracy. In the considered task, it is due to the fact that for some subjects, gestures can be recognized reasonably well in terms of absolute values of the measurements. Nevertheless, it is advised to use the orthogonal mapping when possible as it is the most efficient approach from the memory usage point of view.

VII. Capacity of Distributed Representations for the Task of Correct Retrieval

This section discusses another important tradeoff of using binary HD representations—the dependence of their capacity on the density. The problem of the limited number of HD vectors recoverable from the bundle, i.e., the capacity of distributed representation, is common to both sparse and dense representations.

The capacity characteristic of binary distributed representations is important for a class of pattern recognition applications which requires an understanding of the details of the recall results. For example, noiseless memorization tasks [60] are often used for assessing reservoir computing architectures [24], [45], [61]. In the area of HD dimensional computing there is a similar task called trajectory association [21]. The task consists of two stages. The first stage considers storage of a pattern of tokens (e.g., a sequence of letters, phonemes, etc) HD vector memory. In the second stage, the pattern is retrieved from the memory HD vector.

A. On Capacity of Binary Sparse Distributed Representations

The capacity of the binary SDRs was not previously reported in the literature and therefore is one of the main contributions of this paper.

Denote the number of tokens existing in the system and represented by sparse HD vectors as D. Suppose that this number is finite. D is therefore also the size of the clean-up memory. Denote an N-dimensional sparse vector representing a token as $\mathbf{x}_\ell$.

Suppose a pattern of G tokens is stored in a single memory HD vector using disjunction operation and the CDT procedure.

Suppose without the loss of generality that tokens are bundled with preserved order.[5] The token's order in the bundle is implemented by the permutation operation, which is denoted as ρ. Note that tokens are randomly sampled from the dictionary and, thus, the length of the sequence of tokens can be arbitrarily long despite the finite size D of the dictionary. Thus, the bundle HD vector $\langle\Psi\rangle$ is formed as

$$\langle\Psi\rangle = \left\langle \vee_{m=1}^{G} \rho^{m}(\mathbf{x}_{\ell_m})\right\rangle. \tag{12}$$

The bundle HD vector $\langle\Psi\rangle$ is similar to its components due to the properties of the CDT procedure. Hence, each element of the pattern can be retrieved from the memory HD vector via the search through the clean-up memory. To retrieve a token from position m in the bundle, first, the permutation corresponding to this position is inverted from the memory HD vector as

$$\hat{\mathbf{x}}_{\ell_m} = \rho^{-m}(\langle\Psi\rangle). \tag{13}$$

Next, similarities between the unpermuted bundle HD vector $\hat{\mathbf{x}}_{\ell_m}$ and all HD vectors in the clean-up memory (i.e., D HD vectors representing the dictionary) are calculated. The similarity between two HD vectors is measured by dot product, d. Finally, the token on position m of the bundle is found as the most similar HD vector, i.e., the one with the highest value of the dot product.

Due to the stochastic nature of the distributed representations and the cross-interference, which is introduced into the bundle HD vector when storing many different tokens, the reconstructed pattern contains errors. The task of characterizing the capacity, therefore, converges to the probability of the correct retrieval of a token from a bundle HD vector.

The process of retrieving tokens from the bundle HD vector is also stochastic and can be described with the aid of two random variables. The first variable characterizes the dot product between the unpermuted bundle HD vector $\hat{\mathbf{x}}_{\ell_m}$ and

[5]This assumption allows inclusion of several copies of the same token because each position corresponds to a unique permutation. The permutation in turn preserves orthogonality between HD vectors representing the same token at different positions in the sequence. Note that this is just one possible way of representing ordering in a sequence.

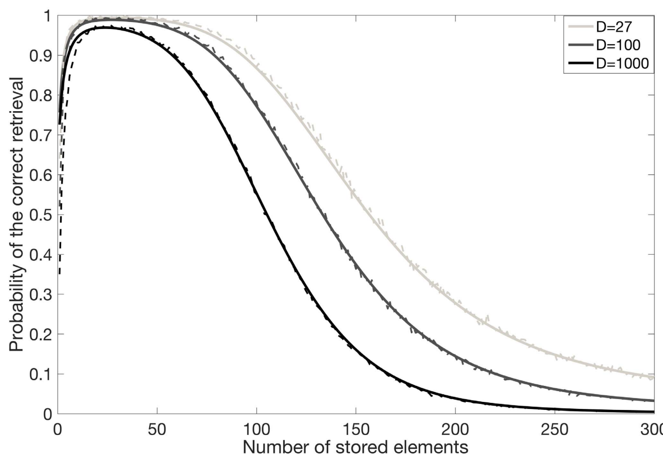


Fig. 10. Probability of the correct retrieval p_{corr} of a single token from the memory HD vector against the number of stored tokens for three different sizes of the dictionary $D = 27$; $D = 100$; and $D = 1000$. During the experiments, $N = 10\,000$, $M = 100$, and $T = 1$. The variances were estimated as 1.1 of the corresponding means. Solid lines are analytical values according to (14)–(18). Dashed lines are averaged experimental results.

the HD vector $\mathbf{x}_{\ell_m}$ in the clean-up memory, which corresponds to the correct token. The second variable characterizes the dot product between the unpermuted bundle HD vector and all $(D-1)$ *not correct* HD vectors in the clean-up memory.

As in [18] both variables are approximated by normal distributions. The former is referred as the hit distribution, while the latter as the reject distribution. Both distributions are defined via their means (μ_H and μ_R) and variances (σ_H^2 and σ_R^2). In the retrieval process, the correct token will be chosen if and only if its HD vector has the highest value of the dot product with the unpermuted bundle HD vector over the entire clean-up memory.

Formally, the probability of the correct retrieval of a token p_{corr} is calculated as follows (see [62] for the general capacity theory):

$$p_{\text{corr}} = \int_{-\infty}^{\infty} \frac{dh}{\sqrt{2\pi}\sigma_H} e^{-\frac{(h-(\mu_H-\mu_R))^2}{2\sigma_H^2}} \left[\Phi\left(\frac{h}{\sigma_R}\right)\right]^{D-1}. \tag{14}$$

Equation (14) depends on values of five variables. The size of the clean-up memory D is defined by the initial conditions of the trajectory association task. Values of other variables depend on the size of the pattern G, the dimensionality of HD vectors N, the expected number of ones in an HD vector M, and the number of iterations in the CDT procedure T.

Mean values of the hit and reject distributions can be calculated if the expected density of ones p_{T1} in the bundle HD vector is known. It can be calculated by (22), see Appendix VIII for details. Then, the mean of the reject distribution μ_R is the expected value of the dot product between two dissimilar HD vectors with different densities p_1 and p_{T1} and it is calculated as

$$\mu_R = N p_1 p_{T1}. \tag{15}$$

The mean of the hit distribution μ_H is proportional to the contribution of a single component HD vector into the

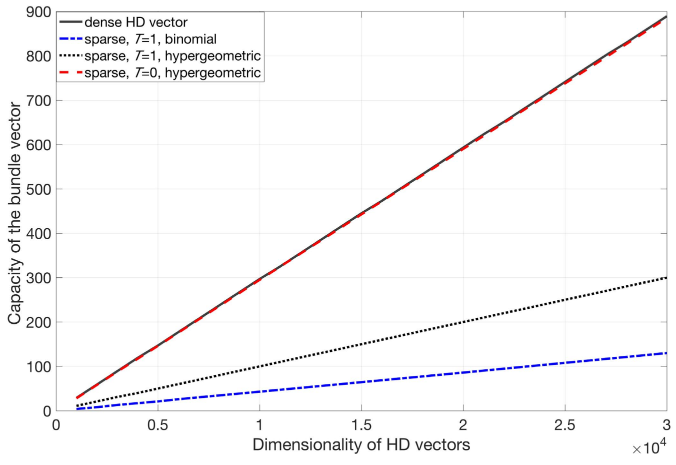


Fig. 11. Capacity of the resultant HD vector versus the dimensionality for dense and SDRs. The capacity shows the maximal number of components which can be retrieved with 99% accuracy. D was set to 27.

superposition HD vector and depends on the expected density of ones in the memory HD vector; μ_H is calculated as

$$\mu_H = N p_{T1}/(p_{S1}/p_1) = N p_1 p_{P1}. \tag{16}$$

The estimation of variances for the distributions (σ_H^2 and σ_R^2) is a nontrivial task due to the complexity introduced by the CDT procedure. To complete the analysis these values were estimated empirically. According to the heuristic, which produces best estimation, the variances of both distributions are approximately equal to their means (in fact 1.1 of their means), i.e.,

$$\sigma_H^2 \approx 1.1\mu_H \tag{17}$$

$$\sigma_R^2 \approx 1.1\mu_R. \tag{18}$$

The probability of the correct retrieval of a token from the bundle can be estimated using (14)–(18). Fig. 10 illustrates the accuracy of the correct retrieval p_{corr} of a single token from the bundle HD vector for different number of stored tokens G for three different sizes of the dictionary. Analytical estimations with (14) match the experimental results very well. Note that all three curves demonstrate the accuracy lower than 100% when the number of stored tokens is small. It is caused by the fact that the density of ones is too low after the CDT procedure (see (22)). It is clear, that with the increase of D, the chance of an incorrect retrieval increases. At the same time, the probability is degrading gradually with the growth of pattern's size G. Eventually, for the large values of G the bundle HD vector becomes dissimilar to all of its components. In this case, the probability of the correct retrieval approaches that of a random guess, i.e., $\lim_{G\to\infty} p_{\text{corr}} = 1/D$.

B. On Capacity of Binary and Bipolar Dense Distributed Representations

The capacity of binary DDRs is extensively studied in [30]. The capacity of the bipolar, i.e., consisting of "+1" and "−1" elements, HD vectors is studied in [18]. Both analyses largely agree with each other. Fig. 11 compares the capacity for dense and sparse HD vectors versus the dimensionality. Dimensionality N varied between 1000 and 30 000. For a given dimensionality, Fig. 11 illustrates the maximal number of components which can be retrieved from the resultant HD vector with at least 99% accuracy. As defined above in (14), the capacity also depends on the size of the dictionary. In Fig. 11 it was fixed to $D = 27$. The number of "1" elements for sparse representations was set to $M = 85$ irrespective of dimensionality.

The solid line in Fig. 11 illustrates the capacity of binarized dense representations. Analogous capacity curve for sparse HD vectors after applying one iteration of the CDT procedure ($T = 1$) is depicted by the dash-dotted line. The capacity of SDRs is significantly lower than that of dense representations. But there are caveats to this. First, the variance of number of "1" elements in sparse HD vector affects the capacity as it is the source of noise. The variance is restricted by drawing sparse HD vector from the hypergeometric distribution rather than from the binomial distribution. In this case, number of "1" elements in HD vector is exactly M rather than approximately M in the case of the binomial distribution. The corresponding capacity curve is illustrated by the dotted line. The change of the distribution generating HD vectors increases the capacity more than twice.

Finally, it should be noted that the usage of the CDT procedure produces the thinned HD vector with lower density of ones but there is a compromise between the capacity and the sparsity of the thinned HD vector. The dashed line shows the capacity of the bundled HD vector without the CDT operation ($T = 0$). In this case, the capacity matches the capacity of dense representations (the upper line; solid and dashed lines coincide), but the bundled HD vector is also dense. In fact, its normalized number of "1" elements can be even higher than 0.5 as in dense representations. Hence, when using sparse representations there is a compromise between the sparsity of the bundled HD vector regulated by the CDT procedure and its capacity.

VIII. Conclusion

This paper discussed tradeoffs of selecting parameters of HD representations for classification and recall tasks. It is demonstrated that for the considered pattern recognition tasks both sparse and dense approaches behave nearly identical. At the same time implementation peculiarities may favor one approach over another (see Appendix B). One of such factors is the number of features required in the potential application. Sparse and dense representations use different mapping operations for the construction of distributed representations of patterns.

Dense representations use the majority rule operation when bundling HD vectors of individual features. This operation does not impose strict restrictions on the number of input HD vectors in order to maintain the density of the resultant HD vector. The resultant HD vector after the majority rule is always dense $p_1 \approx p_0 \approx 0.5$. The density of the resultant HD vector in the sparse representations depends on the number of superimposed inputs due to the CDT procedure.

Note that the CDT procedure adjusts the density of the resultant HD vector by iteratively applying permutation and conjunction operations. The conjunction operation is done with two independent HD vectors each with probability of non zero elements p_1, which in turn depends on the number of input HD vectors being superimposed by OR operation. After conjunction the resulting density is p_1^2. Therefore, the density of the resultant HD vector increases with increase in number of HD vectors in the OR-based superposition. This implies a limitation on the number of superimposed HD vectors for which the density of superposition can be controlled. In order to address this problem appropriate density keeping modifications to CDT operation must be made.

Sparse representation favors lower switching activity with simple and low-cost mapping operations compared to dense alternative. It also requires lower memory footprint. Assuming that the HD vector dimensionality for both approaches is the same, e.g., 8192 then each dense HD vector requires 8192 bits of memory. For sparse HD vectors, on the other hand, only indexes of non-zero elements can be stored. For example, to address each position in $N = 8192$ HD vector 13 bits are needed. Thus, an HD vector with density 0.012 ($M = 100$) can be stored using $13 \cdot 100 = 1300$ bits. Thus, the density of sparse HD vectors is an important adjustable parameter allowing for memory-efficient implementation of VSAs. Another important advantage of sparse representations is in higher capacity of some AM types [63]–[66].

Finally, the experiments in this paper considered the approach of forming class prototypes with HD vectors but there is another potential usage of binary sparse HD vectors. They can be used as the feature vectors for the recently proposed type of SVM [67], which requires that feature vectors are sparse, high-dimensional, and binary. All these conditions are fulfilled by the SDRs. Moreover, the characteristics of representations (e.g., the density of the resultant HD vector) can be easily adjusted as the mathematical mechanisms of the mapping process are well understood.

APPENDIX A
DENSITY OF HD VECTORS IN SPARSE REPRESENTATIONS

This section introduces equations describing the density of patterns' distributed representations formed by the sparse mapping with or without the CDT procedure. The density of the resultant HD vector depends on the chosen parameters of the mapping: the number of features in a pattern (G) and the number of CDT iterations (T) during the thinning process. Dimensionality of an HD vector is denoted as N while approximate number of ones in a component HD vector is M so that $p_1 = M/N$ is the density of ones in the component HD vector. The task is to characterize the density of the thinned HD vector after the CDT procedure (denoted by p_{T1}). The procedure is applied to G random component HD vectors. Assume also that all component HD vectors have the same expected density of ones, p_1.

The first step of the CDT procedure is the construction of the superposition HD vector that is created as a result of elementwise OR operation on all components. The density of

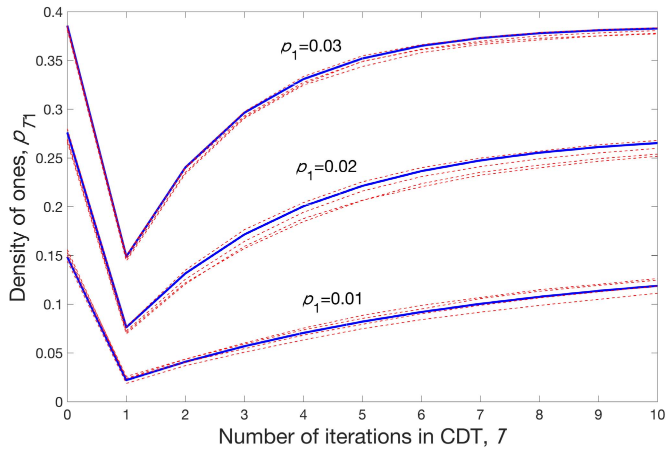


Fig. 12. Comparison between experimental and analytical densities of ones against the number of iterations in the CDT procedure for three different values of p_1. During the experiments, N was fixed to $N = 10\,000$ and $G = 16$. Number of iterations T varied between 0 and 10. Solid lines are analytical calculations according to (22). Dashed lines are experimental results, where each curve is the result of a single experiment using different randomly generated vectors.

the superposition HD vector is defined as

$$p_{S1} = 1 - (1 - p_1)^G. \tag{19}$$

The thinning is performed on the superposition HD vector using its independent permutations. During each iteration of the CDT procedure, the elementwise AND operation is performed on the superposition HD vector and its permuted version. Due to the permutation, the vectors are independent and the density of the result of elementwise AND is defined as

$$p_{\mathrm{SP}1} = p_{S1}^2. \tag{20}$$

Initially, the thinned HD vector is empty. During each iteration, it is updated using the elementwise OR operation with the result of elementwise AND operation between the superposition HD vector and its current permuted version. CDT procedure can be also done in the different order as, first, forming a thinning HD vector using the elementwise OR with T different permutations of the superposition HD vector. Permutations of an HD vector are dissimilar to each other, therefore, the density of the thinning HD vector after T iterations is

$$p_{P1} = 1 - (1 - p_{S1})^T = 1 - (1 - p_1)^{\mathrm{GT}}. \tag{21}$$

Second, the thinned HD vector is the result of the elementwise AND operation between thinning and superposition HD vectors. Note that these HD vectors are independent, then the expected density of ones in the thinned HD vector is calculated as

$$p_{T1} = p_{S1} p_{P1} = (1 - (1 - p_1)^G)(1 - (1 - p_1)^{\mathrm{GT}}). \tag{22}$$

Thus, when there is only one iteration in the CDT procedure (i.e., $T = 1$), (22) simplifies to $p_{\mathrm{SP}1}$ as in (20). Fig. 12 shows the comparison between experimental and analytical densities of ones against the number of iterations in the CDT procedure. The graph was obtained using HD vectors with $N = 10\,000$. Three different densities of ones

TABLE I
SUMMARY OF BINARY HD COMPUTING FRAMEWORKS

VSAs	Ref.	Symbol Set	Similarity metric	p_1	Preserving p_1
BSC	[1]	binary vectors	Hamming	0.5	Yes
MAP	[19]	bipolar vectors	cosine	0.5	No
BSDC	[22]	binary vectors	overlap of vectors	varies	No

in component HD vectors p_1 were considered: 0.01 ($M = 100$), 0.02 ($M = 200$), and 0.03 ($M = 300$). Number of iterations in the CDT procedure varied between 0 and 10, where 0 means that the CDT procedure was not used and the density of ones p_{T1} is calculated according to (19). Solid lines in Fig. 12 correspond to analytical results while dashed lines show experimental densities for individual experiments. It is clear from Fig. 12 that the averaged experimental results (the averaged curves are not shown as they precisely follow the analytical curves) follow the analytical ones for all considered values of p_1.

Note that the minimum values of p_{T1} are achieved when $T = 1$. However, in Fig. 12 these densities are higher than the corresponding p_1 values due to the large number of components G involved in the superposition. Thus, when there is a need for the CDT procedure to control the density when G is large, additional CDT iterations are needed.

APPENDIX B
SUMMARY OF BINARY HD COMPUTING FRAMEWORKS

Table I presents a summary of binary HD computing frameworks including the recommended reference for each approach. The last column indicates the simplicity of preserving the density of ones in HD vectors after operations.

REFERENCES

[1] P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cognit. Comput.*, vol. 1, no. 2, pp. 139–159, Oct. 2009.

[2] R. W. Gayler, "Vector symbolic architectures answer Jackendoff's challenges for cognitive neuroscience," in *Proc. Joint Int. Conf. Cognit. Sci. ICCS/ASCS*, 2003, pp. 133–138.

[3] H. Li *et al.*, "Hyperdimensional computing with 3D VRRAM in-memory kernels: Device-architecture co-design for energy-efficient, error-resilient language recognition," in *IEDM Tech. Dig.*, Dec. 2016, pp. 16.1.1–16.1.4.

[4] A. Rahimi, P. Kanerva, and J. M. Rabaey, "A robust and energy-efficient classifier using brain-inspired hyperdimensional computing," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, 2016, pp. 64–69.

[5] A. Rahimi *et al.*, "High-dimensional computing as a nanoscalable paradigm," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 9, pp. 2508–2521, Sep. 2017.

[6] B. Emruli, R. W. Gayler, and F. Sandin, "Analogical mapping and inference with binary spatter codes and sparse distributed memory," in *Proc. IJCNN*, Aug. 2013, pp. 1–8.

[7] D. Rasmussen and E. Eliasmith, "A neural model of rule generation in inductive reasoning," *Topics Cognit. Sci.*, vol. 3, no. 1, pp. 140–153, 2011.

[8] S. D. Levy, C. Lowney, W. Meroney, and R. W. Gayler, "Bracketing the beetle: How wittgenstein's understanding of language can guide our practice in AGI and cognitive science," in *Artificial General Intelligence* (Lecture Notes in Computer Science), vol. 8598. Cham, Switzerland: Springer, 2014, pp. 73–84.

[9] D. A. Rachkovskij, "Some approaches to analogical mapping with structure-sensitive distributed representations," *J. Experim. Theor. Artif. Intell.*, vol. 16, no. 3, pp. 125–145, 2004.

[10] S. V. Slipchenko and D. A. Rachkovskij, "Analogical mapping using similarity of binary distributed representations," *Inf. Theories Appl.*, vol. 16, no. 3, pp. 269–290, 2009.

[11] D. A. Rachkovskij and S. V. Slipchenko, "Similarity-based retrieval with structure-sensitive sparse binary distributed representations," *Comput. Intell.*, vol. 28, no. 1, pp. 106–129, 2012.

[12] S. D. Levy and R. Gayler, "Vector symbolic architectures: A new building material for artificial general intelligence," in *Proc. 1st Conf. Artif. Gen. Intell. (AGI)*, 2008, pp. 414–418.

[13] D. A. Rachkovskij, E. M. Kussul, and T. N. Baidyk, "Building a world model with structure-sensitive sparse binary distributed representations," *Biol. Inspired Cognit. Archit.*, vol. 3, pp. 64–86, Jan. 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2212683X12000552.

[14] F. R. Najafabadi, A. Rahimi, P. Kanerva, and J. M. Rabaey, "Hyperdimensional computing for text classification," in *Proc. Design, Autom. Test Eur. Conf. Exhibit. (DATE), Univ. Booth*, 2016, p. 1.

[15] D. Kleyko, E. Osipov, N. Papakonstantinou, V. Vyatkin, and A. Mousavi, "Fault detection in the hyperspace: Towards intelligent automation systems," in *Proc. IEEE 13th Int. Conf. Ind. Inf. (INDIN)*, Jul. 2015, pp. 1219–1224.

[16] D. Kleyko, E. Osipov, R. W. Gayler, A. I. Khan, and A. G. Dyer, "Imitation of honey bees' concept learning processes using vector symbolic architectures," *Biol. Inspired Cognit. Architectures*, vol. 14, pp. 57–72, Oct. 2015.

[17] T. A. Plate, "Holographic reduced representations," *IEEE Trans. Neural Netw.*, vol. 6, no. 3, pp. 623–641, May 1995.

[18] S. I. Gallant and T. W. Okaywe, "Representing objects, relations, and sequences," *Neural Comput.*, vol. 25, no. 8, pp. 2038–2078, 2013.

[19] R. W. Gayler, "Multiplicative binding, representation operators & analogy," in *Advances in Analogy Research: Integration of Theory and Data From the Cognitive, Computational and Neural Sciences*, K. J. Holyoak, D. Gentner, and B. N. Kokinov, Eds. Sofia, Bulgaria: New Bulgarian Univ., 1998, pp. 1–4.

[20] S. I. Gallant and P. Culliton, "Positional binding with distributed representations," in *Proc. IEEE Int. Conf. Image, Vis. Comput. (ICIVC)*, Aug. 2016, pp. 108–113.

[21] T. A. Plate, *Holographic Reduced Representation: Distributed Representation for Cognitive Structures*. Stanford, CA, USA: Center for the Study of Language and Information, 2003.

[22] D. A. Rachkovskij, "Representation and processing of structures with binary sparse distributed codes," *IEEE Trans. Knowl. Data Eng.*, vol. 13, no. 2, pp. 261–276, Mar./Apr. 2001.

[23] A. Rahimi, S. Benatti, P. Kanerva, L. Benini, and J. M. Rabaey, "Hyperdimensional biosignal processing: A case study for EMG-based hand gesture recognition," in *Proc. IEEE Int. Conf. Rebooting Comput. (ICRC)*, Oct. 2016, pp. 1–8.

[24] O. Yilmaz, "Machine learning using cellular automata based feature expansion and reservoir computing," *J. Cellular Automata*, vol. 10, nos. 5–6, pp. 435–472, 2015. [Online]. Available: http://www.oldcitypublishing.com/journals/jca-home/jca-issue-contents/jca-volume-10-number-5-6-2015/

[25] C. Eliasmith, *How to Build a Brain*. Oxford, U.K.: Oxford Univ. Press, 2013.

[26] S. Purdy. (2016). "Encoding data for HTM systems." pp. 1–11. [Online]. Available: https://arxiv.org/abs/1602.05925

[27] G. Recchia, M. Sahlgren, P. Kanerva, and M. N. Jones, "Encoding sequential information in semantic space models: Comparing holographic reduced representation and random permutation," *Comput. Intell. Neurosci.*, vol. 2015, Jan. 2015, Art. no. 58. [Online]. Available: https://www.hindawi.com/journals/cin/2015/986574/

[28] D. A. Rachkovskij and E. M. Kussul, "Binding and normalization of binary sparse distributed representations by context-dependent thinning" *Neural Comput.*, vol. 13, no. 2, pp. 411–452, Feb. 2001.

[29] P. Kanerva, *Sparse Distributed Memory*. Cambridge, MA, USA: MIT Press, 1988.

[30] D. Kleyko, E. Osipov, A. Senior, A. I. Khan, and Y. A. Şekercioğlu, "Holographic graph neuron: A bioinspired architecture for pattern processing," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 6, pp. 1250–1262, Jun. 2017.

[31] J. A. Anderson, "Cognitive and psychological computation with neural models," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, no. 5, pp. 799–815, Sep./Oct. 1983.

[32] V. I. Gritsenko, D. A. Rachkovskij, A. A. Frolov, R. Gayler, D. Kleyko, and E. Osipov, "Neural distributed autoassociative memories: A survey," *Cybern. Comput. Eng.*, vol. 2, no. 188, pp. 5–35, 2017.

[33] H. Meng *et al.*, "A modified sparse distributed memory model for extracting clean patterns from noisy inputs," in *Proc. Int. Joint Conf. Neural Netw., (IJCNN)*, Jun. 2009, pp. 2084–2089.

[34] J. T. Abbott, J. B. Hamrick, and T. L. Griffiths, "Approximating Bayesian inference with a sparse distributed memory system," in *Proc. 34th Annu. Meet. Cognit. Sci. Soc.*, 2013, pp. 1–6.

[35] D. Aerts, M. Czachor, and B. De Moor, "Geometric analogue of holographic reduced representation," *J. Math. Psychol.*, vol. 53, no. 5, pp. 389–398, 2009.

[36] P. Kanerva, "Fully distributed representation," in *Proc. Real World comput. Symp.*, 1997, pp. 358–365.

[37] D. Kleyko, E. Osipov, and D. A. Rachkovskij, "Modification of holographic graph neuron using sparse distributed representations," *Procedia Comput. Sci.*, vol. 88, pp. 39–45, Oct. 2016.

[38] O. Räsänen, "Generating hyperdimensional distributed representations from continuous valued multivariate sensory input," in *Proc. 37th Meeting Cognit. Sci. Soc.*, 2015, pp. 1943–1948.

[39] D. A. Rachkovskij, "Real-valued embeddings and sketches for fast distance and similarity estimation," *Cybern. Syst. Anal.*, vol. 52, no. 6, pp. 967–988, 2016.

[40] D. A. Rachkovskij, I. S. Misuno, and S. V. Slipchenko, "Randomized projective methods for the construction of binary sparse vector representations," *Cybern. Syst. Anal.*, vol. 48, no. 1, pp. 146–156, 2012.

[41] R. Donaldson, A. Gupta, Y. Plan, and T. Reimer. (2015). "Random mappings designed for commercial search engines." pp. 1–21. [Online]. Available: https://arxiv.org/abs/1507.05929

[42] S. Ferdowsi, S. Voloshynovskiy, D. Kostadinov, and T. Holotyak, "Fast content identification in high-dimensional feature spaces using Sparse Ternary Codes," in *Proc. IEEE Int. Workshop Inf. Forensics Secur. (WIFS)*, Dec. 2016, pp. 1–6.

[43] D. A. Rachkovskij, "Binary vectors for fast distance and similarity estimation," *Cybern. Syst. Anal.*, vol. 53, no. 1, pp. 138–156, 2017.

[44] O. Rasanen and S. Kakouros, "Modeling dependencies in multiple parallel data streams with hyperdimensional computing," *IEEE Signal Process. Lett.*, vol. 21, no. 7, pp. 899–903, Jul. 2014.

[45] O. Yilmaz, "Symbolic computation using cellular automata-based hyperdimensional computing," *Neural Comput.*, vol. 27, no. 12, pp. 2661–2692, 2015.

[46] S. Benatti *et al.*, "A versatile embedded platform for EMG acquisition and gesture recognition," *IEEE Trans. Biomed. Circuits Syst.*, vol. 9, no. 5, pp. 620–630, Oct. 2015.

[47] O. J. Räsänen and J. P. Saarinen, "Sequence prediction with sparse distributed hyperdimensional coding applied to the analysis of mobile phone use patterns," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 9, pp. 1878–1889, Sep. 2016.

[48] A. G. Anderson and C. P. Berg, "The high-dimensional geometry of binary neural networks," in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–15.

[49] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 1–9.

[50] D. Kleyko, E. P. Frady, and E. Osipov. (2017). "Integer echo state networks: Hyperdimensional reservoir computing." pp. 1–10. [Online]. Available: https://arxiv.org/abs/1706.00280

[51] H. Jaeger, "Adaptive nonlinear system identification with Echo state networks," in *Proc. 15th Adv. Neural Inf. Process. Syst.*, 2003, pp. 609–616.

[52] D. Rachkovskij, "Linear classifiers based on binary distributed representations," *J. Inf. Theories Appl.*, vol. 14, no. 3, pp. 270–274, 2007.

[53] P. Gärdenfors, *The Geometry of Meaning: Semantics Based on Conceptual Spaces*. Cambridge, MA, USA: MIT Press, 2014.

[54] A. Iscen, T. Furon, V. Gripon, M. Rabbat, and H. Jégou. (2014). "Memory vectors for similarity search in high-dimensional spaces," pp. 1–13. [Online]. Available: https://arxiv.org/abs/1412.3328

[55] M. Laiho, J. H. Poikonen, P. Kanerva, and E. Lehtonen, "High-dimensional computing with sparse vectors," in *Proc. IEEE Biomed. Circuits Syst. Conf. (BioCAS)*, Oct. 2015, pp. 1–4.

[56] M. Sahlgren, A. Holst, and P. Kanerva, "Permutations as a means to encode order in word space," in *Proc. 30th Annu. Meet. Cognit. Sci. Soc.*, 2008, pp. 1300–1305.

[57] D. Widdows and T. Cohen, "Reasoning with vectors: A continuous model for fast robust inference," *Logic J. IGPL*, vol. 23, no. 2, pp. 141–173, 2015.

[58] D. A. Rachkovskij, S. V. Slipchenko, E. M. Kussul, and T. N. Baidyk, "Sparse binary distributed encoding of scalars," *J. Autom. Inf. Sci.*, vol. 37, no. 6, pp. 12–23, 2005.

[59] S. Ahmad and J. Hawkins. (2015). "Properties of sparse distributed representations and their application to hierarchical temporal memory." [Online]. Available: https://arxiv.org/abs/1503.07469

[60] H. Jaeger, "Long short-term memory in echo state networks: Details of a simulation study," Jacobs Univ., Bremen, Germany, Tech. Rep. 27, 2012.

[61] S. Nichele and A. Molund. (2017). "Deep reservoir computing using cellular automata." pp. 1–9. [Online]. Available: https://arxiv.org/abs/1703.02806

[62] E. P. Frady, D. Kleyko, and F. T. Sommer. (2018). "A theory of sequence indexing and working memory in recurrent neural networks," pp. 1–62. [Online]. Available: https://arxiv.org/abs/1803.00412

[63] A. Kartashov, A. Frolov, A. Goltsev, and R. Folk, "Quality and efficiency of retrieval for Willshaw-like autoassociative networks: III. Willshaw–Potts model," *Netw., Comput. Neural Syst.*, vol. 8, no. 1, pp. 71–86, 1997.

[64] A. Frolov, D. A. Rachkovskij, and D. Husek, "On informational characteristics of willshaw-like auto-associative memory," *Neural Netw. World*, vol. 12, no. 2, pp. 141–158, 2002.

[65] A. A. Frolov, D. Husek, and D. A. Rachkovskij, "Time of searching for similar binary vectors in associative memory," *Cybern. Syst. Anal.*, vol. 42, no. 5, pp. 615–623, 2006.

[66] A. A. Frolov, D. Húsek, and P. Y. Polyakov, "Recurrent-neural-network-based Boolean factor analysis and its application to word clustering," *IEEE Trans. Neural Netw.*, vol. 20, no. 7, pp. 1073–1086, Jul. 2009.

[67] K. Eshghi and M. Kafai, "Support vector machines with sparse binary high-dimensional feature vectors," Hewlett Packard Labs, Palo Alto, CA, USA, Tech. Rep. HPE-2016-30, 2016, pp. 1–10.

Denis Kleyko received the B.S. (Hons.) degree in telecommunication systems and the M.S. (Hons.) degree in information systems from the Siberian State University of Telecommunications and Information Sciences, Novosibirsk, Russia, in 2011 and 2013, respectively.

He is currently a Ph.D. Student with the Dependable Communication and Computation Systems Group, Department of Computer Science, Electrical and Space Engineering, Luleå University of Technology, Luleå, Sweden. His current research interests include vector symbolic architectures, hyperdimensional computing, bio-inspired cognitive architectures for processing biomedical signals, and machine learning.

Abbas Rahimi received the B.S. degree in computer engineering from the University of Tehran, Tehran, Iran, in 2010 and the M.S. and Ph.D. degrees in computer science and engineering from the University of California San Diego, San Diego, CA, USA, in 2015.

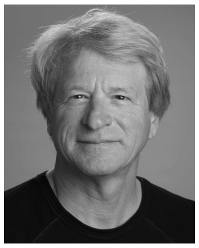
He was a Post-Doctoral Researcher with the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA, USA. He joined the Department of Information Technology and Electrical Engineering, ETHZ, in 2017. He is also affiliated with the Berkeley Wireless Research Center, University of California at Berkeley. His current research interests include embedded systems and software, brain-inspired computing, approximate computing, and massively parallel integrated architectures with an emphasis on improving energy efficiency and robustness.

Dr. Rahimi was a recipient of the ETH Zurich Postdoctoral Fellowship in 2017, the Best Paper at BICT 2017, and the Best Paper Candidate DAC 2013. His doctoral dissertation has received the 2015 Outstanding Dissertation Award in the area of "New Directions in Embedded System Design and Embedded Software" from the European Design and Automation Association.

Dmitri A. Rachkovskij received the D.Sc. degree in artificial intelligence from the International Research and Training Center for Information Technologies and Systems (IRTC ITS), National Academy of Sciences of Ukraine and Ministry of Education and Science of Ukraine, Kiev, Ukraine, in 2008, the M.S. degree in radiophysics from Rostov State University, Rostov, Russia, in 1983, and the Ph.D. degree from V. M. Glushkov Institute of Cybernetics, Kiev, in 1990.

In 1987, he joined the Cybernetics Center, IRTC ITS, where he is currently a Leading Research Scientist. He led over 20 projects and has authored or co-authored more than 80 refereed papers, including those in high-impact journals. His research is connected with the domain of artificial intelligence and neural networks. He also involved extensively in the areas of pattern recognition, software and hardware neurocomputers, and micromechanics. His current research interests include distributed representations, similarity search, analogical reasoning, and distributed autoassociative memories.

Evgeny Osipov received the Ph.D. degree in computer science from the University of Basel, Basel, Switzerland, in 2005. He is currently a Full Professor of dependable communication and computation system with the Department of Computer Science and Electrical Engineering, Luleå University of Technology, Luleå, Sweden. His current research interests include the application of cognitive computing and communication architectures to low-power embedded systems in the context of future cyber-physical systems, Internet-of-Things, and intelligent industries.



Jan M. Rabaey (F'95) holds the Donald O. Pederson Distinguished Professorship at the University of California at Berkeley, Berkeley, CA, USA, where he is currently a Founding Director of the Berkeley Wireless Research Center and the Berkeley Ubiquitous SwarmLab, and also the Electrical Engineering Division Chair. He has made high-impact contributions to a number of fields, including advanced wireless systems, low-power integrated circuits, sensor networks, and ubiquitous computing. His current interests include the conception of the next-generation integrated wireless systems over a broad range of applications and exploring the interaction between the cyber and the biological world.

Dr. Rabaey was a recipient of major awards, amongst which the IEEE Mac Van Valkenburg Award, the European Design Automation Association Lifetime Achievement Award, and the Semiconductor Industry Association University Researcher Award. He is a member of the Royal Flemish Academy of Sciences and Arts of Belgium. He has been involved in a broad variety of start-up ventures.