Holographic Graph Neuron: A Bioinspired Architecture for Pattern Processing

Denis Kleyko, Evgeny Osipov, Alexander Senior, Asad I. Khan, and Yaşar Ahmet Şekercioğlu

***Abstract*— In this paper, we propose a new approach to implementing hierarchical graph neuron (HGN), an architecture for memorizing patterns of generic sensor stimuli, through the use of vector symbolic architectures. The adoption of a vector symbolic representation ensures a single-layer design while retaining the existing performance characteristics of HGN. This approach significantly improves the noise resistance of the HGN architecture, and enables a linear (with respect to the number of stored entries) time search for an arbitrary subpattern.**

***Index Terms*— Associative memory (AM), holographic graph neuron (HoloGN), hyperdimensional computing, pattern recognition, vector symbolic architectures (VSAs).**

I. Introduction

GRAPH neuron (GN) is an approach for memorizing patterns of generic sensor stimuli for later template matching [2], [3]. It is based on the hypothesis that a better associative memory (AM) resource can be created by changing the emphasis from high-speed sequential CPU processing to parallel network-centric processing. In contrast to contemporary machine-learning approaches, GN allows the introduction of new patterns in the learning set without the need for retraining. While doing so, it exhibits a high level of scalability, i.e., its performance and accuracy do not degrade, as the number of stored patterns increases over time.

Vector symbolic architectures (VSAs) [4] are a bioinspired method of representing concepts and their meanings for modeling cognitive reasoning. It exhibits a set of unique properties which make it suitable for the implementation of artificial general intelligence [5]–[7] and hence the creation of complex systems for sensing and pattern recognition without reliance on complex computation. In the biological world, extremely successful applications of such approaches can be found. One example is the ordinary house fly: it is capable of conducting very complex maneuvers, even though it possesses a very little computational capacity.[1] Another interesting biological example is the compound eyes of arthropods. These compound eyes consist of a large number of sensors with limited and localized processing capabilities for performing relatively complex sensing tasks [9].

This paper presents contributions in two domains: the organization of AM and the properties of connectionist distributed representation. In the first area, this paper introduces a novel bioinspired architecture, called holographic GN (HoloGN), for one-shot pattern learning, which is built upon GN's flexible input encoding abstraction and the strong reasoning capabilities of the VSA representation. In the second area, this paper extends the understanding of the performance properties of distributed representation, which opens the way for new applications.

This paper is structured as follows. Section II places HoloGN in the scope of AM research. Section III presents an overview of the work related to the matter presented in this paper. The background information on the theories, concepts, and approaches used in HoloGN is described in Section IV. Sections V–VII present the main contribution of this paper, the design of the HoloGN architecture, and its performance characteristics. Finally, the conclusion is drawn in Section VIII.

II. HoloGN in the Scope of Associative Memory Research

This section presents a discussion which places HoloGN in the scope of other AM approaches. AM is designed for applications requiring fast pattern matching. Compared with random access memory in modern computing architectures, where the goal is to retrieve the content of a certain place in the memory by supplying the address of this place, the goal of the AM is different. The cue to the memory is a pattern in a generic sense,[2] which is stored in memory, either entirely or in parts. The task is to search the entire memory for the best matching entry. On a very high level, the taxonomy of AM can be divided into localist and distributed approaches.

Localist-based AM models [10] rely on the so-called grandmother cell hypothesis, where each part of a pattern considered by the model is represented with a single neuron. In studies of

Manuscript received January 26, 2015; revised February 11, 2016; accepted February 21, 2016. This work was supported by the Swedish Foundation for International Cooperation in Research and Higher Education under Grant IG2011-2025.

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

A. Senior is with the Department of Electrical and Computer Systems Engineering, Monash University, Clayton, VIC 3800, Australia (e-mail: alexander.senior@monash.edu).

A. I. Khan is with the Clayton School of Information Technology, Monash University, Clayton, VIC 3800, Australia (e-mail: asad.khan@monash.edu).

Y. A. Şekercioğlu is with the Heudiasyc Laboratory, Compiègne University of Technology, Compiègne 60200, France (e-mail: asekerci@ieee.org).

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.2016.2535338

[1] In [8], house fly's properties are compared and contrasted with an advanced fighter plane as follows: "Whereas the F-35 Joint Strike Fighter, the most advanced fighter plane in the world, takes a few measurements—airspeed, rate of climb, rotations, and so on and then plugs them into complex equations, which it must solve in real time, the fly relies on many measurements from a variety of sensors but does relatively little computation."

[2] The term pattern is used here to represent a set of (heterogeneous) values or concepts that repeat over time to form an experience of an artificial system.

perception, such models, in spite of known criticisms, may be plausible in certain biological systems in which a wide range of neurons prefer specific stimuli; for example, in [11], it was shown that a population of a few tens of neurons in a monkey's brain was selectively responsive to different features of their body.

The second broad class of AM models, which are based on distributed representations, opposes the localist AMs by suggesting that for implementing association-based operations on structured information, it is implausible to have dedicated neurons for each element of the structure. In distributed representation theory, a specific stimulus is coded by a unique pattern of activity over a group of neurons.

The work presented in this paper has its roots and inspiration in two specific models of the distributed AM: sparse distributed memory (SDM) [12] and hierarchical GN (HGN) [2]. SDM is a mathematical model of human long-term memory, which is used for storing and retrieving large amounts (in the order of 2^{1000} bits) of information. Its two main principles are the aggregation of similar stimuli based on the statistical similarity of their encoded representations and reducing the overlap between the representations by storing them sparsely over a huge memory space. SDM has been extensively applied to pattern recognition [13] as the model for implementing AM in the context of cognitive computing architectures [14], as well as its demonstrated suitability for approximating Bayesian inference [15]. One of the unsolved challenges attributed to SDM is the so-called encoding problem, i.e., the problem of how to encode stimuli in the distributed representation [16].

HGN can be classified as a distributed AM in the sense that it models a stimulus as a graph of activities of multiple elementary neurons. Note that in HGN, the definition of a neuron is simplified: it is modeled as an array of possible values taken from a finite alphabet of discrete values. Without discussing the biological plausibility of this model, it is applicable in many practical real-world scenarios. For example, all current sensing devices are characterized by having a finite operating range, and quantization techniques (i.e., representing the level of the sampled continuous signal in a finite number of levels) suggest that this model is feasible. In practice, this model of a neuron and an interconnected network of them enables the lightweight implementation of AM-based sensor networks using low-end and power-constrained computing devices.

The work presented in this paper provides a step toward addressing the encoding problem of SDM. In particular, it demonstrates that a simple (in the computational sense) but usable model of a neuron from the HGN approach leads to the practical implementation of an AM using high-dimensional (HD) representations. By doing so, we eliminate the need to maintain a graph hierarchy by applying the HD representations and mathematical apparatus of SDM for modeling congregations of memories based on statistical similarities between the encoded concepts. As a result, the encoding of an acquired heterogeneous sensory stimulus not only preserves the properties of the original model but also paves the way to an entirely new class of applications, such as in [17] and [18].

III. Related Work

AM is a subdomain of artificial neural networks, which utilizes the benefits of content-addressable memory [19] in microcomputers. The AM concept was originally developed in an effort to utilize the power and speed of existing computer systems for solving large scale and computationally intensive problems by simulating biological neurosystems.

The HGN approach [2] is a type of AM which signifies the hierarchical structure in its implementation. Hierarchical structures in AM models are of interest, as these have been shown to improve the rate of recall in pattern recognition applications. The distributed HGN scheme also allows for better control of the network resources. This scheme compares well with contemporary approaches, such as self-organizing map and support vector machine in terms of speed and accuracy.

The VSAs were introduced in [5] as a class of connectionist models that use hyperdimensional vectors (i.e., vectors of several thousand elements) to encode structured information as a distributed or holographic representation. In this technique, the structured data are represented by performing basic arithmetic operations on field-value tuples. Distributed representations of data structures are an approach actively used in the area of cognitive computing for representing and reasoning upon semantically bound information [4], [20]. The cognitive capabilities achievable using VSAs have been demonstrated by creating systems capable of solving Raven's progressive matrices [21], [22] and via the imitation of concept learning in honey bees [23], [24].

In [25], a VSA-based knowledge-representation architecture is proposed for learning arbitrarily complex, hierarchical, and symbolic relationships (patterns) between sensors and actuators in robotics. Recently, the theory of hyperdimensional computing, and VSA in particular, has been adopted for implementing novel communication protocols and architectures for collective communications in machine-to-machine communication scenarios [26], [27]. The first work demonstrates the unique reliability and timing properties essential in the context of industrial machine-to-machine communications. The latter work shows the feasibility of implementing collective communications using the current radio technology. This paper presents an algorithmic ground for further design of the distributed HoloGN in addition to the architecture presented in [26].

IV. Overview of Essential Concepts and Theories

A. Hierarchical Graph Neuron

Fig. 1 shows the HGN approach. Consider only the bottom layer of the construction without the hierarchy of upper nodes; this bottom level is the original flat network of GNs [2]. Each GN is a model for a set of generic sensory values (e.g., the value of a pixel or a real value of sensory data). When seen as a network, graph neurons can be modeled by an array where columns are individual GNs and rows are possible symbols, which a neuron can recognize, e.g. an integer between 0 and 100. For example, if there are only two possible symbols, say X and Y in the alphabet of a pattern,

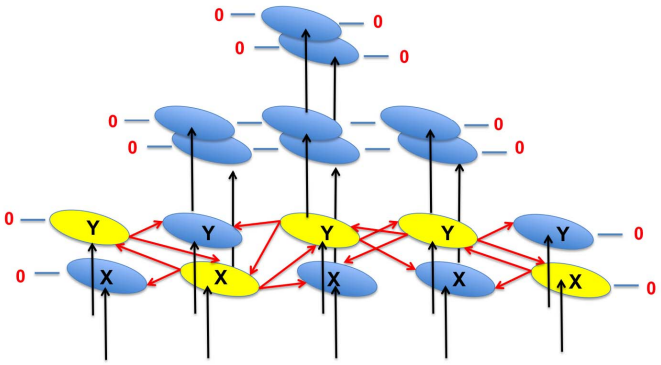


Fig. 1. HGNs of a five-element pattern of two symbols. Short arrows: how GNs communicate indices of the activated GN elements with their neighbors, creating logical connectivity. Null GN numbers are assigned at the start and at the end of the array to maintain a consistent reporting scheme, where every activated GN reports to both the adjacent GNs.

then only two rows are needed to represent those symbols. The number of columns[3] in the GN array determines the size of patterns which it can analyze.

An input pattern is defined as a stimulus produced within the network. In Fig. 1, each GN can analyze a symbol (X or Y) of a pattern consisting of five elements. In each GN (column), only the element with the matching value (a row ID) would respond. For example, if the pattern is YXYYX, then in the second column, the X element will be activated in response to this input stimulus. If a particular element in the GN column is activated, it sends a report to all adjacent GNs. The report contains the activated GN's element ID (the row index). Otherwise, it simply ignores the stimulus and returns to the idle state.

During the next phase, all GNs communicate the indices of the activated elements with the adjacent columns at their level, and in addition, communicate the stored bias information to the layer above. The procedure continues in an ascending manner until the collective bias information reaches the top of the hierarchy. The higher level GNs can thus provide a more authoritative assessment of the input pattern. The accuracy of HGN has been demonstrated to be comparable to the accuracy of neural network with back-propagation [2].

B. Example of HGN Operation

In order to give a better understanding of the encoding procedure of the original HGN, consider the task of memorizing primitive pixel patterns, as shown in Fig. 2. The patterns are 6×6 black and white pixel images. Suppose that the pixels are numbered from 1 to 36 starting from the upper left corner. To facilitate the presentation, each pattern is given the name of a still life figure in the Game of Life [28]. That is, the patterns in Fig. 2 show the ship, the aircraft carrier, and the beehive. The patterns will be subsequently presented to the HGN in the order ship, carrier, and beehive.

The bottom layer of the HGN consists of an array of 36 GNs $(\mathrm{GN}_1, \mathrm{GN}_2, \ldots, \mathrm{GN}_{36})$, where each GN_i $(i = 1, \ldots, 36)$ is

[3]In this paper, the words column and GN are used interchangeably and refer to a single GN. The term GN array refers to several GNs used to recognize a pattern of several elements, where one neuron is used to recognize one element of the pattern.

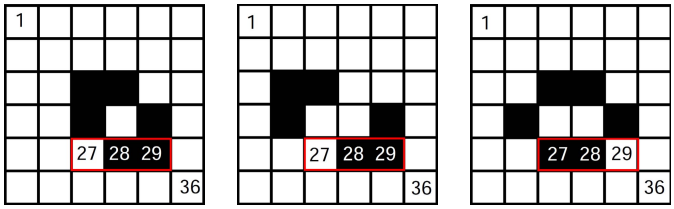


Fig. 2. Three examples of pixel images for HGN memorization. These patterns are named after still life patterns in the Game of Life [28]. The highlighted cells have ordered indices 27, 28, and 29 and are referred to in the text. (a) Ship still life. (b) Aircraft carrier still life. (c) Beehive still life.

dedicated to recognizing the state (ON/OFF) of the corresponding pixel. The subscript i corresponds to the number of the pixel to which this GN is assigned. GN_i is a column with two elements $\mathrm{GN}_i[0]$ = white and $\mathrm{GN}_i[1]$ = black. In what follows, the operation of the three GNs highlighted by the red rectangle in Fig. 2 is considered, i.e., GN_{27}, GN_{28}, and GN_{29}.

The first presented pattern (ship) results in activations $\mathrm{GN}_{27}[0]$, $\mathrm{GN}_{28}[1]$, and $\mathrm{GN}_{29}[1]$, i.e., white–black–black. The activated indices will be communicated by each GN to their immediate neighbors. Each GN then notes down which neighbors were activated in a table of information called the bias array. The bias array essentially links a certain activation of neighbors to an integer index of the record. In this example, the middle black element ($\mathrm{GN}_{28}[1]$) will associate the activation of its white neighbor on its left and its black neighbor on its right with an index of 0, for example. It will then communicate this index to the element directly above it in the hierarchy of layers (as shown in Fig. 1). This (upper) element will in turn broadcast its activation to its neighbors, receive similar messages, and create a new entry in its own bias array. This process will continue until it reaches the uppermost layer in the GN hierarchy, consisting of a single column of two elements.

Now, consider the aircraft carrier and beehive images; as the highlighted pixels in the aircraft carrier [Fig. 2(b)] are the same in the two images, when the three elements broadcast their activation and consult their bias arrays, they will find that they encountered the same activation before and hence will emit the same index to the higher layer. However, in the beehive image [Fig. 2(c)], the activation will now be black–black–white instead of white–black–black. This means that $\mathrm{GN}_{27}[1]$ and $\mathrm{GN}_{29}[0]$ will be activated for the first time. They will form entries in their (empty) bias arrays in a similar fashion as before and transmit the index (0) to the assigned node of the upper hierarchical level. Meanwhile, the black element in the middle ($\mathrm{GN}_{28}[1]$) will activate as before, but as its activated neighbors are different, it will form a new entry in its bias array with an index of 1, and transmit this information to the next upper layer. In this way, the differences and similarities between patterns are stored in a distributed manner throughout the hierarchy of GNs.

C. Motivation for Holographic Graph Neuron

An important issue in hierarchical models is the resource requirement overhead, in particular, with regard to the number

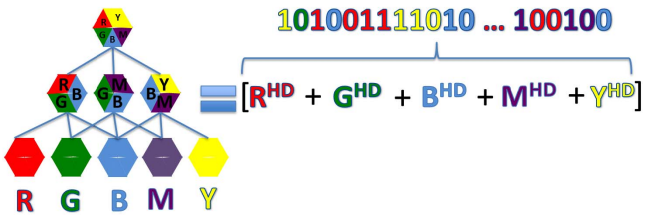


Fig. 3. High-level illustration of the proposed solution. Representation of the HGN using VSA. R: red. G: green. B: blue. M: magenta. Y: yellow.

of processing elements required. For example, if the HGN for recognition of patterns of five elements (as shown in Fig. 1) is to be implemented in a wireless sensor network, where a sensor node is a single GN, then nine sensor nodes are required, but only five are actually used to observe patterns (bottom layer). This paper proposes a holographic approach, which borrows the abstraction of the GN and enables a flat GN array to operate with higher level of accuracy and comparable recall time to that of HGN, without the need for a complex topology and additional nodes. The high-level logic of the proposed solution is shown in Fig. 3, which illustrates the concept underlying both the approaches. On the left, HGN creates the representation of a pattern through communication indices of the activated GN elements between neighbors. Thus, on the highest level, HGN forms a representation of the whole pattern (illustrated with mixed colors in Fig. 3). In contrast, on the right, HoloGN achieves a similar result by encoding each GN element and the combination of their particular activations into a distributed representation. The fundamentals of the algebra of distributed representations are presented in Section IV-D.

D. Fundamentals of Vector Symbolic Architecture and Binary Spatter Codes

As mentioned previously, VSA is an approach for encoding and operating on distributed representation of information, and has previously been used mainly in the area of cognitive computing for representing and reasoning upon semantically bound information [4], [29].

The fundamental difference between the distributed and localist representations of data is as follows: in traditional (localist) computing architectures, each bit and its position within a structure of bits are significant (for example, a field in a database has a predefined offset amongst other fields, and a symbolic value has a unique representation in ASCII codes), whereas in a distributed representation, all entities are represented by vectors of very high dimensions. For the remainder of this paper, the term HD vector is used when referring to such codes. In particular, the binary spatter code variety of HD vectors is utilized. High dimensionality refers to the fact that in HD vectors, several thousand positions (of binary numbers) are used for representing a single entity; Kanerva [4] proposes the use of vectors of 10 000 binary elements. Such entities have the following useful properties.

1) Randomness: Randomness means that the values at each position of an HD vector are independent of each other, and 0 and 1 components are equally probable. In very high dimensions, the distances from any arbitrary chosen HD vector to more than 99.99% of all other vectors in the representation space are concentrated around 0.5 normalized Hamming distance. Interested readers are referred to [4] and [12] for a comprehensive analysis of the probabilistic properties of the hyperdimensional representation space.

Denote the density of a randomly generated HD vector (i.e., the number of 1s in an HD vector) as k. The probability of selecting a random vector of length d with density k, where the probability of 1s appearance equals p, is described by the binomial distribution

$$\Pr(k, d, p) = \binom{d}{k} p^k (1-p)^{d-k}. \tag{1}$$

When d is in the range of several thousand binary elements, the calculation of the binomial coefficient requires sophisticated computations. Therefore, approximations of binomial distribution are used for large values of d. It can be well approximated via the normal approximation, the de Moivre–Laplace theorem, or the Poisson distribution (for sparse vectors). The calculations in this paper use the de Moivre–Laplace theorem.

2) Similarity Metric: The similarity between two binary representation is characterized by normalized[4] Hamming distance

$$\Delta_H(A, B) = \frac{1}{d}\|A \otimes B\|_1 = \frac{1}{d}\sum_{i=0}^{d-1} a_i \otimes b_i \tag{2}$$

which (for two vectors) measures the number of positions in which they differ. Here, a_i and b_i are bits on positions i in vectors A and B of dimension d, and $\otimes$ denotes the bitwise XOR operation.

3) Generation of HD Vectors: Several random binary vectors with the above-mentioned properties can be generated from one such vector via the cyclic shift operation. Using this operation, a sequence of K vectors, which are pseudo-orthogonal to a given initial random HD vector A (i.e., the normalized Hamming distance between them equals approximately 0.5), is obtained by cyclically shifting A by i positions, where $1 \leq i \leq K < d$. Later in this paper, this operation is denoted as $\mathrm{Sh}(A, i)$. The cyclic shift operation has the following properties.

1) It is invertible, i.e., if $B = \mathrm{Sh}(A, i)$, then $A = \mathrm{Sh}(B, -i)$.
2) It is associative in the sense that $\mathrm{Sh}(B, i + j) = \mathrm{Sh}(\mathrm{Sh}(B, i), j) = \mathrm{Sh}(\mathrm{Sh}(B, j), i)$.
3) It preserves the Hamming weight of the result: $\|B\|_1 = \|\mathrm{Sh}(B, i)\|_1$.
4) The result is dissimilar to the vector being shifted: $(1/d)\|B \otimes \mathrm{Sh}(B, i)\|_1 \approx 0.5$.

Note that the cyclic shift is a special case of the permutation operation [4]. In the context of VSA, permutations were previously used to encode sequences of semantically bound elements.

4) Bundling of Vectors: Joining several entities into one structure is done with the bundling operation; it is implemented by a thresholded sum of the HD vectors representing

[4]That is, normalized to the dimension of the HD vectors.

the entities. A bitwise thresholded sum of n vectors results in 0 when $n/2$ or more arguments are 0, and 1 otherwise. In the case of an even number in sum, ties are broken at random, which is equivalent to adding an extra random HD vector [4]. Furthermore, the terms thresholded sum and majority sum are used interchangeably and denoted as $[A+B+C]$. The relevant properties of the majority sum are in the following.

1) For any number of operands, the result is a vector, with the number of 1 components being approximately equal to the number of 0 components.
2) The result is similar to all vectors included in the sum.
3) The more HD vectors that are involved in a majority operation, the closer the normalized Hamming distance between the resultant vector and any HD vector component is to 0.5.
4) If several copies of any vector are included in a majority sum, the resultant vector is closer to the dominating vector than to other components.

The algebra on VSA includes other operations, e.g., binding and permutation [4]. Since they are not used in this paper, their definitions and properties are omitted.

V. Holographic Graph Neuron

This section presents one of the main contributions of this paper: the adoption of the VSA data representation for the implementation of the HGN approach.

The commented MATLAB code for the implementation of HoloGN and simulation scenarios used in this paper to produce Figs. 8–12 are available online.[5]

A. Encoding

In the case of HoloGN, all its elements, i.e., symbols of individual neurons (e.g., possible values of image pixels), are indexed uniquely, and the index of a particular element is derived as a function of the GN's ID. Let IV_j be an initialization HD vector for GN j. The initialized vectors for different GNs are chosen to be mutually orthogonal. Then, the HD index of element i in GN j is computed as $E^{\mathrm{HD}}_{(j,i)} = \mathrm{Sh}(\mathrm{IV}_j, i)$, where Sh() is a cyclic shift operation resulting in the generation of a vector orthogonal to IV_j HD vector [30].[6]

B. Construction of VSA Representation of Activated GNs

Let n be the number of individual GNs. When a GN array (n GNs) observes a pattern, the activated elements communicate their HD-represented indices to all other GNs; the holographic representation of the activated elements is then

$$\mathrm{HGN} = \left[\sum_{j=1}^{n} \left(E^{\mathrm{HD}}_j\right)\right] \tag{3}$$

where E^{HD}_j is the HD index of the activated element in $\mathtt{GN}_j$, and the addition operation is the bundling operation, or thresholded sum, as described in Section IV-D.[7] As discussed previously, in the resultant HD vector, the normalized Hamming distances between each component and the composition vector are strictly less than 0.5. This property will be utilized later when constructing data structures for recall of patterns in HoloGN.

C. Data Structures for Storing and Retrieving Holographic Representations in HGN Elements

HoloGN will store the holographic representation of the entire pattern (3) observed across all GNs. A possible architecture is for all memorized patterns to be collected and stored centrally at a processing node, where the role of processing node can be assigned to one of GNs or to some other external device.

Depending on the particular application of HoloGN, the memorized patterns could be stored either in an unsorted list or in bundles. The first mode of storing HoloGN patterns corresponds to the case, where the structure of the observed patterns is unknown; the latter mode is used in the case of supervised learning. Section VI describes different HoloGN usages and recall strategies.

VI. HoloGN Recall Strategies

This section introduces and evaluates the performance of two major recall modes: the one-shot case and the case of supervised learning.

A. Time-Efficient ξ-Accurate Recall in an Unsorted HoloGN Storage

The common step in both recall modes is the procedure for the time-efficient search over an (unsorted) list of HoloGN records. Recall that all manipulations with VSA-encoded entities are done using simple bitwise arithmetic operations and calculations to obtain the Hamming distance between entities. However, it is assumed for this paper that there is no particular optimized implementation of VSA's bitwise operations; this is because such operations are tailored to the architectures of specific microprocessors, which operate with words of substantially lower dimensionalities (typically 32 or 64 bits). Therefore, the adoption of these methods for implementing the bitwise operations on words of thousands of bits would be cumbersome. Instead, an easily analyzable computational model is adopted in this paper, which could also be adapted to implementation in specialized computing architectures.

In what follows, each HoloGN pattern $\mathbf{h}_i$ is modeled as a row vector of d elements. The list of stored HoloGN patterns is, therefore, modeled as an $l \times d$ matrix, where l is the number of the learned (stored) HoloGN patterns. Denote this matrix as $\mathbf{H}$. The task of recalling a pattern with a target recall threshold of ξ ($\xi < 0.5$) is formulated as finding the rows $\mathbf{h}_i$ in $\mathbf{H}$ with normalized Hamming distances to the query pattern $\mathbf{h}_q$ less than or equal to ξ.

[5] A git user can obtain the source code using the command git clone https://github.com/eaoltu/hologn.git. Readers who are not familiar with the git version management system can download the code directly from `https://sites.google.com/site/evgenyosipov/professional/research-projects/hologn`. The Readme.txt file included in the bundle contains details on how to work with the code. To save space, the code snippets are not included in this paper. Instead, the readers are assisted with references to particular functions in the implementation.

[6] In the implementation, the encoding is done in the *hologn_encoder* function.

[7] In the HoloGN code, the majority sum is implemented in the *majority_sum* function. This function is then used to construct the HGN as in (3).

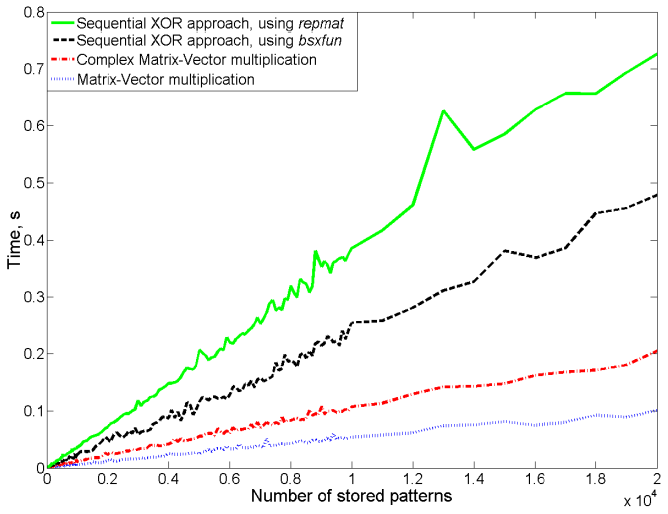


Fig. 4. Comparison of different approaches to recalling patterns, showing the time taken to calculate the Hamming distance against the number of previously presented patterns.

The conventional way of computing the normalized Hamming distance between vectors would be to perform the following sequence of computations for each row in $\mathbf{H}$.

1) Perform an elementwise XOR with the vector query.
2) Sum up all elements in the intermediate result.
3) Divide the result by the dimensionality of the vectors.

Although there is no fundamental significance in the way the Hamming distance is calculated in a specific computing environment, in practice, it is important to have efficient implementation. Therefore, the performance of two implementations of this method using MATLAB's `repmat` and `bsxfun` functions is shown by the top two curves in Fig. 4; `bsxfun` applies an element-by-element binary operation to two arrays with singleton expansion enabled. The curves demonstrate a linear but rapid increase in the recall time with the increase in the number of the stored patterns. The lowest curve in Fig. 4 shows the performance of matrix-vector multiplication of the same size, which is chosen as the reference case. The results were obtained on an Intel Core i7-3520M 2.9 GHz machine with Windows 7 operating system using one processor.

1) Binary Spatter Codes as Complex Numbers: In order to improve the efficiency of calculating Hamming distances over a vast number of HoloGN patterns, it is proposed to represent HoloGN patterns using complex numbers, where a binary 0 would be represented by $\sqrt{-1}$ (i.e., the imaginary number); and a binary 1 would remain 1. The idea behind this transformation is simple: the multiplication of bits in the same position should produce three outcomes: $-1 = j \times j$, $1 = 1 \times 1$, and $j = 1 \times j$. That is, the multiplication of two similar bits produces a real number, and the multiplication of two different bits produces an imaginary number. In this way, the sum of the imaginary parts over all positions in the resulting vector, divided by dimensionality d, will correspond to the normalized Hamming distance between the two vectors. Thus, the suggested method allows us to implement the calculation of Hamming distance using the standard method

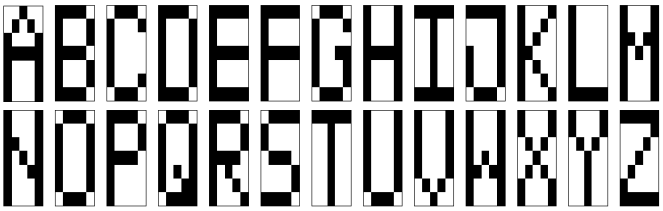


Fig. 5. List of letter images for comparison.

of matrix-vector multiplication, as shown in the following:

$$\mathbf{H} \times \mathbf{h}_q = \begin{pmatrix} \sqrt{-1} & 1 & \cdots & \sqrt{-1} \\ 1 & 1 & \cdots & \sqrt{-1} \\ \vdots & \vdots & \ddots & \vdots \\ \sqrt{-1} & \sqrt{-1} & \cdots & 1 \end{pmatrix} \times \begin{pmatrix} \sqrt{-1} \\ 1 \\ \vdots \\ \sqrt{-1} \end{pmatrix} = \begin{pmatrix} 254 + 1633j \\ 617 + 3824j \\ \vdots \\ 548 + 4952j \end{pmatrix}. \quad (4)$$

The performance of the proposed method is shown in Fig. 4 (dashed curve). It demonstrates that the calculation of the Hamming distance is only two times slower than the usual matrix-vector multiplication. In particular, to calculate the Hamming distance from the target vector to each of 20 000 stored patterns, it takes approximately 200 ms on the test machine. Further optimization of the matrix multiplication and execution on parallel architectures indicate the realistic bounds on the recall time over extremely large numbers of patterns.[8]

B. Case Study 1: Best Match Probing Under One-Shot Learning

1) Example (Encoding and Recall of Letters Using HoloGN): Consider the 5×7 black and white pixel images of Latin letters, as shown in Fig. 5. The encoding process is exemplified in Fig. 6 and includes the following steps.

1) Initialization of HoloGN is as follows.
   a) Set the dimensionality of the HD vectors.[9] In this paper, 10 000 bits are used for the simulations.
   b) Set the number of GNs.[10] The image of a letter consists of 35 pixels. Every pixel is assigned one GN, i.e., 35 GNs are initialized in the simulations.
   c) Generate the initialization of HD vector IV for each GN.[11]
2) Present a letter image to the initialized HoloGN.[12] Images for all 26 letters are stored in Letters.mat.
3) For each GN (pixel), shift cyclicly this GN's IV to the value of the pixel (1 for white and 0 for black).[13]

[8] In the HoloGN implementation, the transformation of the array of binary values into the array of complex values is done by the *bin2com* function.
[9] Line 28 in the *hologn_encoder* function.
[10] Line 31 in the *hologn_encoder* function.
[11] Line 36 in the *hologn_encoder* function.
[12] Line 45 in the *letters_encoding* function.
[13] Lines 43–45 in the *letters_encoding* function.

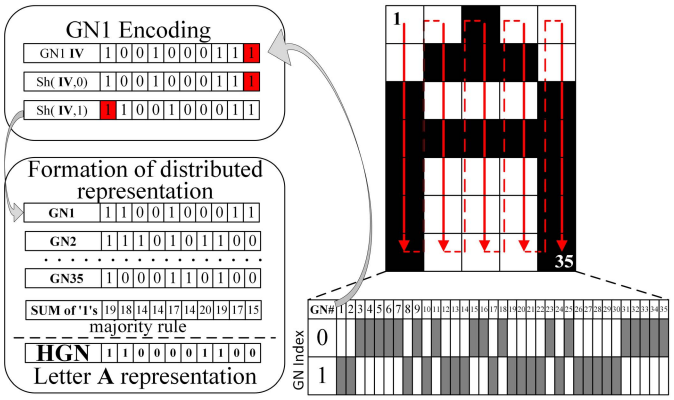


Fig. 6. Example of HoloGN encoding for letter A. For simplicity of presentation, the encoding operation is exemplified on 10-D vectors. In the simulations, HD vectors with 10 000 elements were used.

4) Form the distributed representation of the letter (see Section V-B) using shifted IV vectors.[14]
5) Store the letter's representation in the list of memorized patterns.[15]

The encoding process is repeated for all 26 letters. Thus, at the end of the letter encoding procedure, 26 HD vectors are created, which are stored in the list of memorized patterns.

The recall phase for a randomly distorted letter is done as follows.

1) The pixels of the chosen letter are randomly distorted according to the specified distortion level.[16]
2) The distributed representation of the distorted pattern is formed as described earlier using the *letters_encoding* function.
3) The processing unit calculates Hamming distances from the representation of the distorted letter to all of the 26 representations stored in the list of memorized patterns.[17]
4) The stored pattern with the minimal Hamming distance to the distorted one is recalled by the HoloGN as the desired letter.[18]

Note that the whole simulation scenario for case study 1 is available in the file *Scenario_recall_pattens_with_distortions*.

The first usage of HoloGN probes the existence of the target query pattern amongst the previously memorized patterns. The perfect match in this case would be indicated by a normalized Hamming distance of zero. The deviation from zero, therefore, reflects the degree of proximity of the query to one or several stored HoloGN patterns.[19] In the following example, the accuracy of the HoloGN recall is compared with the performance of the original HGN approach. For the sake of fair comparison, the 7×5 pixel letters of the Roman alphabet (as in [2]) are used.

In the memorization phase, a set of noise-free images of letters shown in Fig. 5 was presented to both architectures.

[14]Line 48 in the *letters_encoding* function.
[15]Line 45 in the *letters_encoding* function.
[16]Line 66 in the *Scenario_recall_pattens_with_distortions* scenario.
[17]Line 28 in the *item_memory_c* function.
[18]Line 31 in the *item_memory_c* function.
[19]The recall of the closest memorized pattern for a given representation of the target query is implemented in the *item_memory_c.* function.

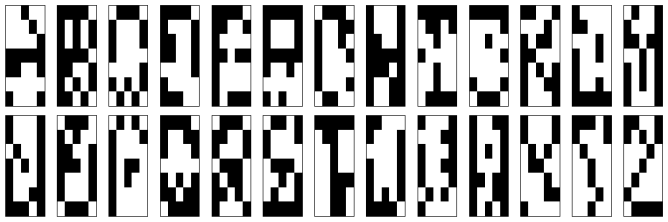


Fig. 7. Example of images distorted by 5 bits (14.3%) presented for recall.

In the recall phase, the images of the same letters distorted with different levels of random distortions (between 1 bit corresponding to a distortion of 2.9% of the pattern's size and 5 bits equivalent to 14.3% distortion) were presented to the architectures for the recall. An example of a noisy input is shown in Fig. 7. In the case of HoloGN, the pattern with the lowest normalized Hamming distance to the presented distorted pattern was returned as the output.

Fig. 8 shows the results of the accuracy comparison between the recall results for the HoloGN approach and the reference HGN architecture. 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. The analysis shows that the performance of the HoloGN-based AM at least matches that of the original approach. However, in certain characters, the recall is inferior to other letters. For example, the accuracy of character O recognition is persistently lower than other letters. This is due to its similarity to several other characters. In particular, when recalling O distorted by 5-bits HoloGN recall scoring is C—5.2%, D—11.4%, G—11.9%, and Q—5.1%. In the simulations, the average accuracy of HoloGN when recalling patterns is 51.7% higher than the accuracy of HGN.

Note that in the performance comparison, the average recall accuracy of HoloGN is equivalent to the average recall accuracy in the original low-dimensional space when an image is represented as 35-bit binary vector (0 corresponding to black and 1 corresponding to white) and the Hamming distance is used to measure the similarity between the images. The advantage of using HD vectors can be observed when distortions occur in the representational space of each approach, i.e., bit distortions to the original low-dimensional pattern (see Fig. 7) and the proportional distortion to the HD pattern formed by HoloGN. For example, with a 1-bit distortion, 2.9% of the original bit pattern is changed, and hence, 290 bits in the 10 000-bit HD vector would be changed to obtain the equivalent distortion. The results of the average accuracy of the recall for this experiment are shown in Fig. 9. Due to the properties of the distributed nature of HoloGN's representations, it shows superior robustness to distortions, while the recall accuracy in the low-dimensional representation significantly drops with the increased level of distortion.

2) Encoding and Recall of Nonbinary Patterns: We extended our example into nonbinary patterns by changing the white background of the images into a third color (gray). In both HoloGN and using the low-dimensional representations, the original black and white images shown in Fig. 5 are memorized. To represent the images for recollection in

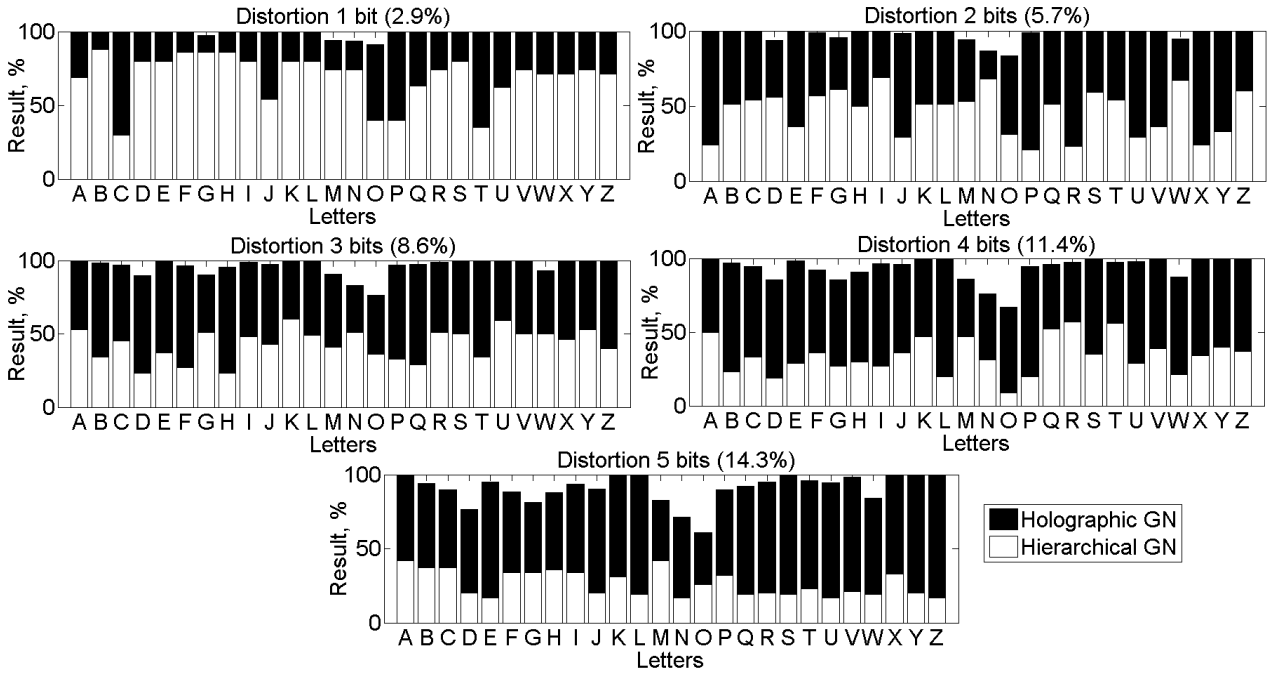


Fig. 8. Results from testing black and white images of letters using recall patterns with distortions ranging from 1 (2.9%) to 5 bits (14.3%).

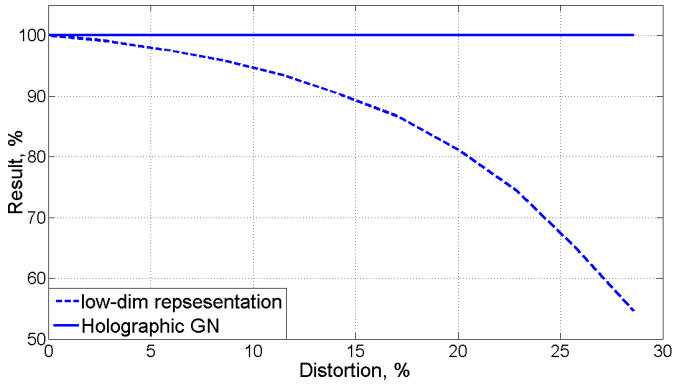


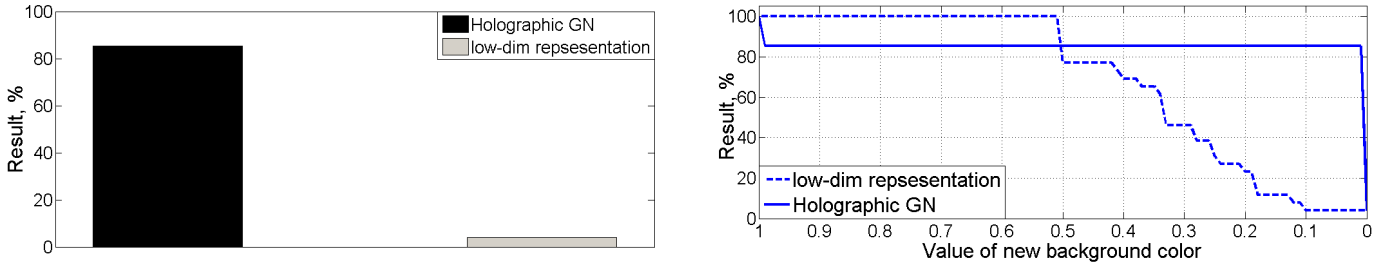
Fig. 9. Average accuracy of the recall for HoloGN and for the original low-dimensional (binary) representation against distortion level when distortions occur in the representational space of each approach, i.e., bit distortions to the original low-dimensional patterns and the proportional distortion to the HD representations of the patterns.

the low-dimensional space, each pixel is now represented by two bits, where 00 corresponds to black color, 10 to white color, and 01 to the new gray color; hence, the overall representation of a single image now requires 70 bits. For the HD representation in HoloGN, we encode the new color by shifting the initialization HD vector by two, as compared to no shift for black and one shift for white (see Fig. 6). This process is explained in Section VI-B.

The average accuracy for both approaches when queried with the nonbinary patterns is shown in Fig. 10. Initially, Hamming distance is used to measure similarity for both approaches, as shown in Fig. 10(a). The accuracy of HoloGN in this experiment is 85.5%. This is caused by the fact that some letters are subsets of others. For example, since the black pixels in the letter T are contained within the letter I, when recalling the letter T with the changed background, the similarity to the representation of the original letter is determined only by the black pixels in the same position as in the query. As both the original T and I have the same number of black pixels in common, the Hamming distances of their HD vectors to the HD vector of the query will be approximately the same, and hence, HoloGN will recall either T or I with roughly the same probability. Note that if the images of letters were constructed, such that none of the letters were a subset of another, the recall accuracy of HoloGN would be 100%.

The recall accuracy with the low-dimensional representations is very poor (3.9%). In fact, in this experiment, only the letter B was recalled correctly. The reason for this is that the letter B contains the least amount of white background, and hence, it is the most similar to all of the nonbinary patterns presented. Thus, with only one correct recall, the accuracy is $1/26$.

We also compared the two approaches when using Euclidean distance to measure similarity for the low-dimensional representation. As shown in Fig. 10(b), the value of the background color varies between 1 (i.e., white, so the altered images are exactly the same as in Fig. 5) and 0 (i.e., black, so the image is completely black). The Euclidean distance is calculated by the squared root of the sum of the squared differences between pixels. Note that similarity for HoloGN is calculated as previously, but white is represented by a cyclic shift of 255, black by 0, and the background color varies between 0 and 255. While the value of the new color is above 0.5 (i.e., more similar to white), the recall accuracy for the original representation is without error; however, it diminishes substantially as the value of the new color decreases to zero and the background becomes more similar to black. In contrast, due to the properties of the cyclic shift operation,



(a) (b)

Fig. 10. Average accuracy of the recall for HoloGN and for the original low-dimensional representation in a nonbinary case using a third color for the background in the test images. (a) Accuracy of HoloGN and the low-dimensional representation when Hamming distance is used as a metric for both. (b) Accuracy when Euclidean distance is used as a metric for the original representation (note that Hamming distance is still used for HoloGN). The value of the new color used in the Euclidean distance calculation was changed from 1 (white) to 0 (black), while in the HD vector, a cyclic shift from 255 (white) to 0 (black) was used.

HoloGN treats any shift (that is not zero or 255) representing the new color as completely dissimilar to both white and black, and hence HoloGN's performance is constant and the same as in the previous experiment. The comparison of the results in Fig. 10(b) for the two approaches shows that HoloGN's representations outperformed the low-dimensional representations approximately 50% of the time.

These results highlight the importance of similarity encoding by HoloGN and suggest that the effectiveness of encoding values by the cyclic shift operation depends on the specific application. For example, it is reasonable to encode symbols of letters as completely dissimilar HD vectors (see [30]), as their meanings are orthogonal; on the other hand, the encoding of continuous values would more likely require similar values with similar representations in HD spaces. Hence, future work on HoloGN will investigate new ways of representing similarity when mapping values of GN to HD vectors, using the approaches outlined in [31]–[34].

C. Case Study 2: HoloGN Recall Under Supervised Learning

The above-mentioned analysis is a very positive result for the proposed bioinspired AM-based pattern-processing architecture, since the accuracy of the original HGN approach was demonstrated to be as accurate as artificial neural networks with back-propagation [2]. While establishing formal relationships to the framework of artificial neural networks is outside the scope of this paper, this section presents the results of the pattern recognition accuracy of the HoloGN architecture under supervised learning.[20] In this case, the HoloGN is presented with a series of randomly distorted patterns for each letter with different levels of distortion (between 1 and 15 bits), as exemplified in Fig. 7. In the experiments, up to 500 patterns for each letter and every level of distortion were presented for memorization. For the particular level of distortion i, all y presented patterns of the particular letter L_i were bundled to a single HoloGN representation as

$$\mathbf{h}(L) = \left[\sum_{i=1}^{y}(\mathrm{HGN}(L_i))\right]. \quad (5)$$

[20]By supervised learning, we mean labeling distorted patterns by bundling them with the distributed representation of the correct character during the training phase.

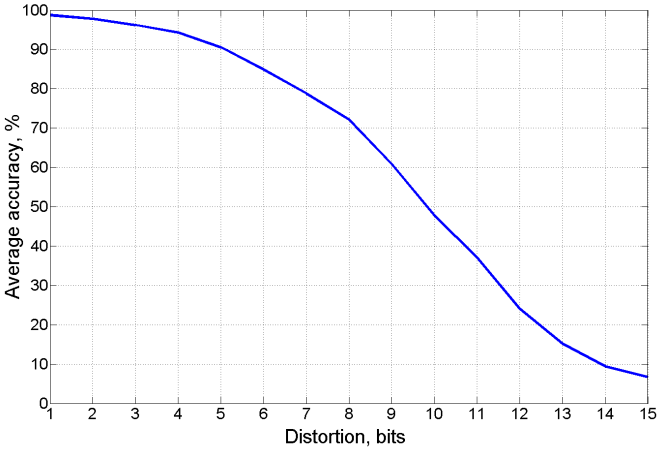


Fig. 11. Average accuracy of HoloGN recall under supervised learning memorization as a function of the distortion level.

Thus, by the end of the learning phase, the HoloGN list will contain 26 HD bundles, each jointly representing all (presented) distorted variants of the particular letter. In the recall phase, for each distortion level, HoloGN was presented with 500 new distorted patterns of each letter. The accuracy of the recall was measured as the percentage of the correctly recognized letters averaged over the alphabet.

Fig. 11 shows the obtained results: 90% accurate recall was observed when learning symbols distorted by up to 5 bits (14.3%). While the accuracy predictably decreases rapidly with the increase of distortion in the patterns presented, a reasonable 80% recall accuracy was observed for learning sets with 7-bit distortion (20%).

Fig. 12 shows the convergence of the HoloGN recall accuracy with the number of noisy samples presented for the case of 5-bit distortion (14.3%). For larger learning sets, the average accuracy in Fig. 12 is approaching the average value in Fig. 11 for 5-bit distortion. This is a positive result for the presented architecture, which illustrates the suitability of the HoloGN in applications requiring supervised learning.

VII. Pattern Decoding and Subpattern-Based Analysis

There is a class of pattern recognition applications, which requires an understanding of the details of the recall results.

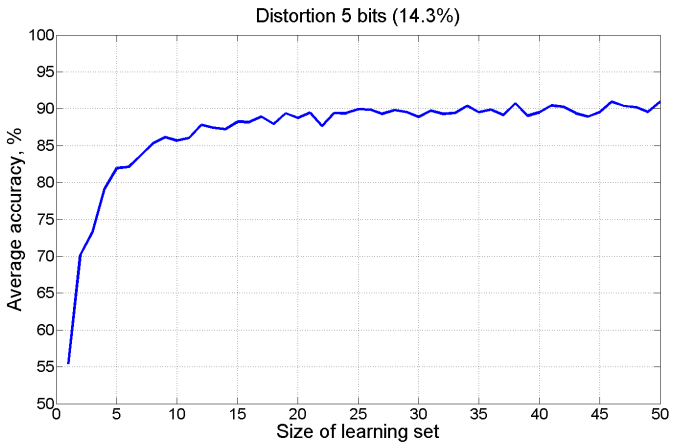


Fig. 12. Accuracy of HoloGN under supervised learning memorization as a function of the number of examples presented for a given level of distortion.

For example, when a recall returns several possible patterns of given recall accuracy, the task would be to understand the overlapping elements. This section considers two aspects of this task: a robust decoding of elementary components out of a distributed VSA representation and a quantitative metric of the similarity via direct comparison of distributed representations, without the need for decoding those representations.

The VSA approach of representing data structures by definition makes decoding of the individual components a tedious task, requiring a brute force test on the inclusion of all possible HD codewords for each GN. The majority sum—which is used for creating HoloGN representations of the observed patterns—imposes a limit on the number of HD codeword operands, above which a robust decoding of the individual operands is impossible.

A. Preliminaries

Denote the density of a randomly generated HD vector (i.e., the number of 1s in an HD vector) as k. The probability of picking a random vector of length d with density k, where the probability of 1s appearance, defined as p, is described by (1). The mean density of a random vector is equal to $d \cdot p$. Note that in reality, the density of randomly generated HD vectors will obviously deviate from the mean value. However, according to (1), the density k approaches the mean value with the increase of dimensionality d. In other words, the probability of generating HD vector with $k \gg d \cdot p$ or $k \ll d \cdot p$ decreases with the increase of dimensionality d.

Define thr as the threshold probability of generating a vector with a certain deviation of density being negligibly small. Let k^- and k^+ characterize the lower and the upper bounds of the interval of possible densities. This is shown in Fig. 13. The bounds for a given d, p, and thr (1) are calculated according to

$$k^-(d, p, \text{thr}) = \max_k(\Pr(k, d, p) \leq \text{thr} | k < (d \cdot p)) \quad (6)$$

and

$$k^+(d, p, \text{thr}) = \min_k(\Pr(k, d, p) \leq \text{thr} | k > (d \cdot p)). \quad (7)$$

The value of thr is chosen to be small (10^{-6}).

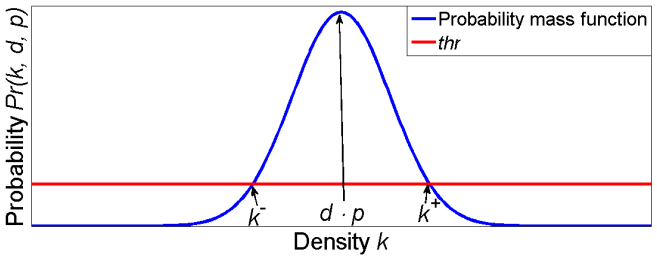


Fig. 13. Binomial distribution and its parameters describing HD vectors.

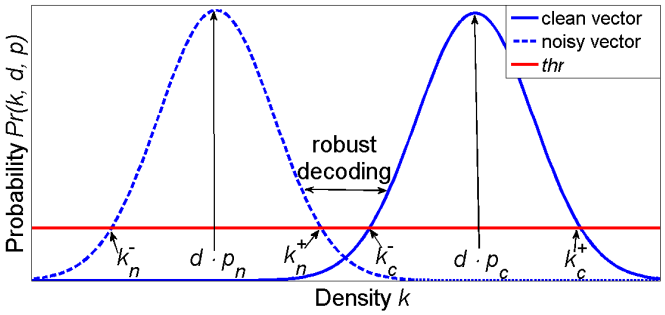


Fig. 14. Explanation of vector capacity. Solid line: random HD vector. Dashed line: noise introduced by majority sum.

B. Capacity of HoloGN Representations

Suppose there exists an item memory [4] containing HD vectors representing atomic concepts.[21] Recall that when several HD vectors are bundled by the majority sum the noise of flipped bits increases with the number of components. For a given dimensionality d, there is a limit on the number of bundled HD vectors, beyond which the resulting HD vector becomes orthogonal to every component vector; hence, the capacity of the resulting vector is defined as the maximal number of mutually orthogonal HD vectors, which can be robustly decoded from their majority sum composition by probing the item memory. Note, however, that component vectors in fact never become truly orthogonal to the resulting HD vector, although the component vectors can no longer be reliably extracted from the resulting HD vector.

In order to characterize the capacity of the composition vector for a given dimensionality, one needs to characterize the level of noise p_n introduced by the bundling operation. This is calculated as in [16] by

$$p_n(n) = \frac{1}{2^n}\left[\frac{1}{2} - \binom{n-1}{\frac{1}{2}(n-1)}\right] \quad (8)$$

where n is the number of atomic vectors in the resulting majority sum vector.

Consider an arbitrary HD vector $\mathbf{A}$ to be decoded from a majority sum composition. Let $\mathbf{N}$ be a vector of noise imposed by the majority sum operation. Since the components are mutually orthogonal, the density of 1s in the noise vector is also described by the binomial distribution $\Pr(k, d, p_n)$. As each new vector is added, the level of noise increases, and hence, the mean of the noise vector density will approach 0.5, as shown in Fig. 14. Due to the properties of HD space, vector

[21] In the case of HoloGN, an atomic concept is the code for the particular HoloGN element.

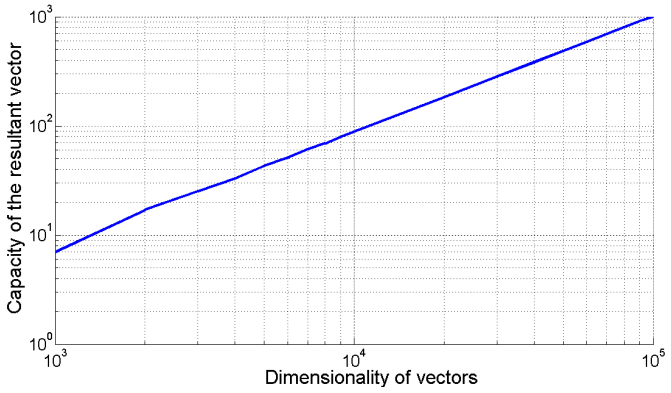


Fig. 15. Capacity of the component vector versus the dimensionality. The calculations use the threshold of thr $= 10^{-6}$.

A will be undecodable when the upper bound $k^+(d, p_n, \text{thr})$ of the density of noise vector **N** approaches the lower bound $k^-(d, 0.5, \text{thr})$. That is, the noisy version of **A** becomes orthogonal to its clean version. This logic is shown in Fig. 14, where the resulting majority sum vector is orthogonal to all components. Therefore, the capacity of the distributed representation with dimensionality d is computed by

$$\text{Capacity}(d, \text{thr}) = \max_n (k^+(d, p_n(n), \text{thr}) \leq k^-(d, 0.5, \text{thr})). \tag{9}$$

Fig. 15 shows the capacity of HD vector of different dimensionalities calculated for threshold probability thr $= 10^{-6}$. In particular, for $d = 10\,000$ bits, the capacity of the robustly decodable VSA is 89 vectors. A similar analysis of capacity of HD vectors consisting of 1 and -1 components was presented in [7]. The main difference between the two methods is that the calculations in (9) require the thr parameter, while the analysis in [7] requires the size of the item memory and the probability of successful decoding. Nevertheless, the two approaches for capacity estimation largely agree. For example, in [7], for the dimensionality of 90 000, the capacity is 1000, and the same capacity for (9) is achieved with the dimensionality of 100 000.

C. Calculation of the Number of Common Component Vectors in Two Resulting Vectors

Given the rather conservative limits on the number of robustly decodable elements in a distributed representation, it is important that the proposed HoloGN architecture can estimate the similarity between different patterns without decoding them. This section provides a method for quantitatively measuring the number of overlapping elements as a function of their relative normalized Hamming distance. Denote m and n ($m \leq n$) as lengths of two patterns, and denote c as the number of common elements in these patterns. Let **M** be a $c \times d$ matrix of common elements, where each row contains a random HD vector of dimension d encoding element c_i. Denote an arbitrary column of matrix **M** as **C**. Since rows in **M** are independent, the density of 1s in each column also follows the binomial distribution with $p = 0.5$ and length c. Denote the number of 1s in column **C** as $\|\mathbf{C}\|_1$.

In order to calculate the normalized Hamming distance between the distributed representations of two patterns with known m, n, and c, consider all possible cases when bits in the same position are different. The normalized Hamming distance between two patterns can be estimated as

$$p(c, m, n) = \sum_{\|\mathbf{C}\|_1=0}^{c} \frac{\binom{c}{\|\mathbf{C}\|_1}}{2^c} (p_1(m, c, \|\mathbf{C}\|_1) p_0(n, c, \|\mathbf{C}\|_1) + p_0(m, c, \|\mathbf{C}\|_1) p_1(n, c, \|\mathbf{C}\|_1)) \tag{10}$$

where $p_i(j, c, \|\mathbf{C}\|_1)$ stands for the probability of having i (0 or 1), when the representation consists of $j = m$ or $j = n$, atomic vectors and c of these vectors are overlapped.

Due to the symmetry in the calculation of probabilities, $p_i(j, c, \|\mathbf{C}\|_1)$ is presented only for $p_1(m, c, \|\mathbf{C}\|_1)$ case. There are three possibilities for the calculation of $p_1(m, c, \|\mathbf{C}\|_1)$.

1) If $\|\mathbf{C}\|_1$ is greater than $m/2$, then the result of the majority sum is 1, i.e., p_1 is equal to 1.
2) If the number of possible 1s is smaller than $m/2$, then the probability of p_1 is equal to 0.
3) Otherwise, the probability should consider all possible combinations and their probabilities.

Therefore, $p_1(m, c, \|\mathbf{C}\|_1)$ can be calculated as follows:

$$p_1(m, c, \|\mathbf{C}\|_1) = \begin{cases} 1 & \text{when } \|\mathbf{C}\|_1 > \dfrac{m}{2} \\ 0 & \text{when } (m - c) < s \\ \dfrac{\sum_{i=s}^{(m-c)} \binom{m-c}{i}}{2^{(m-c)}} & \text{otherwise.} \end{cases} \tag{11}$$

Here, $s = (m + 1/2) - \|\mathbf{C}\|_1$. Similarly, for $p_0(m, c, \|\mathbf{C}\|_1)$, we have

$$p_0(m, c, \|\mathbf{C}\|_1) = \begin{cases} 1 & \text{when } (c - \|\mathbf{C}\|_1) > \dfrac{m}{2} \\ 0 & \text{when } (m - c) < l \\ \dfrac{\sum_{i=l}^{(m-c)} \binom{m-c}{i}}{2^{(m-c)}} & \text{otherwise} \end{cases} \tag{12}$$

and $l = (m + 1/2) - (c - \|\mathbf{C}\|_1)$.

Fig. 16 shows the normalized Hamming distances between two resulting vectors for different numbers of overlapping vectors. The results show that the larger the number of common elements, the smaller the normalized Hamming distance between resulting vectors.

This method opens a way for the construction and analysis of patterns far beyond VSA's robustly decodable capacity. The problem with practical application of this method, however, comes with the rapid convergence of the normalized Hamming distance indicator to 0.5, making the difference between analyzable patterns indistinguishable, as shown in Fig. 16. For example, for HoloGN representations of patterns with 15 elements, subpatterns of three overlapped elements are robustly detected, while patterns with fewer overlapped elements are indistinguishable. Thus, the minimal number of overlapped elements in two patterns, which can

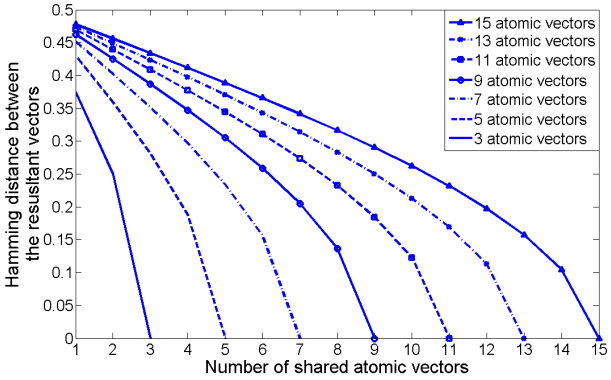


Fig. 16. Normalized Hamming distance between two resulting vectors against number of components in common. The number of atomic vectors is the same, $m = n$.

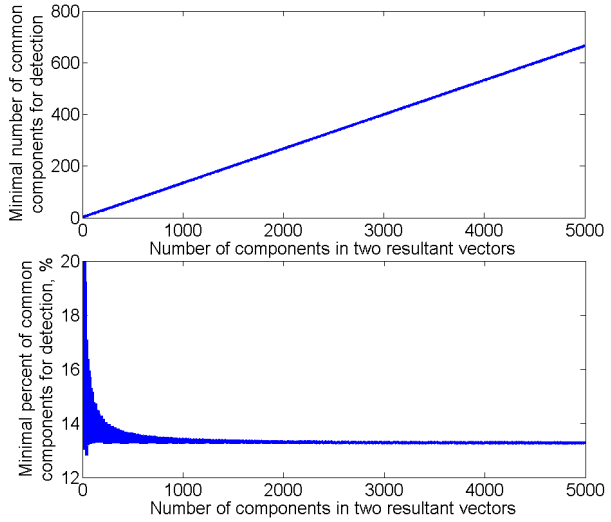


Fig. 17. Minimal number of common components, which is sensible between two patterns of the same size against the size of patterns, $d = 10\,000$ and $\text{thr} = 10^{-6}$.

be robustly detected using the normalized Hamming distance indicator, is called the bundle's sensitivity.

The analysis of the sensitivity is similar to the analysis of the capacity of VSA representation in Section VII-B. For two patterns of length m and n elements, and c overlapped components, the sensitivity is calculated by

$$\text{Sensitivity}(d, \text{thr}, m, n) = \min_{c}(k^{+}(d,\, p(c, n, m), \text{thr}) \leq k^{-}(d, 0.5, \text{thr})). \quad (13)$$

Fig. 17 shows the development of the sensitivity threshold with the number of elements in the compared patterns. The results show that the number of components for robust detection grows linearly with the size of the pattern. Patterns with more than 500 elements should contain at least 14% overlapped elements to be robustly detected by the proposed method.

VIII. Conclusion

This paper has presented HoloGN—a novel approach for memorizing patterns of generic sensor stimuli. HoloGN is built upon the previous GN algorithm and adopts a vector symbolic representation for encoding GN's states. The adoption of the vector symbolic representation ensures a single-layer design for the approach, which leads to much simpler computational operations. The approach presented in this paper possesses a number of unique properties. First, it enables a linear (with respect to the number of stored entries) time search for an arbitrary subpattern. Second, while maintaining the previously reported properties of the HGN, HoloGN improves the noise resistance of the architecture, leading to substantial improvement of pattern recall accuracy.

References

[1] E. Osipov, A. I. Khan, and A. Amin, "Holographic graph neuron," in *Proc. Int. Conf. Comput. Inf. Sci. (ICCOINS)*, 2014, pp. 1–6.

[2] B. B. Nasution and A. I. Khan, "A hierarchical graph neuron scheme for real-time pattern recognition," *IEEE Trans. Neural Netw.*, vol. 19, no. 2, pp. 212–229, Feb. 2008.

[3] A. I. Khan and A. H. M. Amin, "One shot associative memory method for distorted pattern recognition," in *Proc. 20th Austral. Joint Conf. Artif. Intell.*, vol. 4830. Gold Coast, QLD, Australia, 2007, pp. 705–709.

[4] 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.

[5] S. D. Levy and R. Gayler, "Vector symbolic architectures: A new building material for artificial general intelligence," in *Proc. Conf. Artif. General Intell., 1st AGI Conf.*, 2008, pp. 414–418. [Online]. Available: http://dl.acm.org/citation.cfm?id=1566174.1566215.

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

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

[8] R. Zbikowski, "Fly like a fly [micro-air vehicle]," *IEEE Spectr.*, vol. 42, no. 11, pp. 46–51, Nov. 2005.

[9] Y. M. Song *et al.*, "Digital cameras with designs inspired by the arthropod eye," *Nature*, vol. 497, no. 7447, pp. 95–99, May 2013.

[10] K. Pagiamtzis and A. Sheikholeslami, "Content-addressable memory (CAM) circuits and architectures: A tutorial and survey," *IEEE J. Solid-State Circuits*, vol. 41, no. 3, pp. 712–727, Mar. 2006.

[11] D. I. Perrett, E. T. Rolls, and W. Caan, "Visual neurones responsive to faces in the monkey temporal cortex," *Experim. Brain Res.*, vol. 47, no. 3, pp. 329–342, 1982. [Online]. Available: http://dx.doi.org/10.1007/BF00239352.

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

[13] 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.

[14] 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.

[15] J. T. Abbott, J. B. Hamrick, and T. L. Griffiths, "Approximating bayesian inference with a sparse distributed memory system," in *Proc. 34th Annu. Conf. Cognitive Sci. Society*, 2013, pp. 1–6.

[16] P. Kanerva, "Fully distributed representation," in *Proc. Real World Comput. Symp. (RWC)*, 1997, pp. 358–365.

[17] 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. Inform. (INDIN)*, Jul. 2015, pp. 1219–1224.

[18] D. Kleyko and E. Osipov, "Brain-like classifier of temporal patterns," in *Proc. Int. Conf. Comput. Inf. Sci. (ICCOINS)*, 2014, pp. 1–6.

[19] K. J. Schultz, F. Shafai, and G. F. R. Gibson, "Content addressable memory system with cascaded memories and self timed signals," U.S. Patent 6 230 236, May 8, 2001.

[20] P. Kanerva, "A family of binary spatter codes," in *Proc. Int. Conf. Artif. Neural Netw. (ICANN)*, 1995, pp. 517–522.

[21] 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.

[22] B. Emruli, R. W. Gayler, and F. Sandin, "Analogical mapping and inference with binary spatter codes and sparse distributed memory," in *Proc. IJCNN*, 2013, pp. 1–8.
[23] 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. Archit.*, vol. 14, pp. 57–72, Oct. 2015.
[24] D. Kleyko, E. Osipov, M. Björk, H. Toresson, and A. Öberg, "Fly-the-Bee: A game imitating concept learning in bees," *Procedia Comput. Sci.*, vol. 71, pp. 25–30, Dec. 2015.
[25] S. D. Levy, S. Bajracharya, and R. W. Gayler, "Learning behavior hierarchies via high-dimensional sensor projection," in *Proc. Workshops 27th AAAI Conf. Artif. Intell.*, 2013, pp. 1–3. [Online]. Available: http://www.aaai.org/ocs/index.php/WS/AAAIW13/paper/view/7075.
[26] D. Kleyko, N. Lyamin, E. Osipov, and L. Riliskis, "Dependable MAC layer architecture based on holographic data representation using hyperdimensional binary spatter codes," in *Multiple Access Communications*, Berlin, Heidelberg: Springer, 2012, pp. 134–145.
[27] P. Jakimovski, H. R. Schmidtke, S. Sigg, L. W. F. Chaves, and M. Beigl, "Collective communication for dense sensing environments," *J. Ambient Intell. Smart Environ.*, vol. 4, no. 2, pp. 123–134, Mar. 2012.
[28] M. Gardner, "Mathematical games," *Sci. Amer.*, vol. 223, no. 4, pp. 120–123, Oct. 1970.
[29] T. A. Plate, "Distributed representations and nested compositional structure," Ph.D. dissertation, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, 1994.
[30] D. Kleyko and E. Osipov, "On bidirectional transitions between localist and distributed representations: The case of common substrings search using vector symbolic architecture," *Procedia Comput. Sci.*, vol. 41, pp. 104–113, Dec. 2014.
[31] 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.
[32] D. A. Rachkovskij, "Formation of similarity-reflecting binary vectors with random binary projections," *Cybern. Syst. Anal.*, vol. 51, no. 2, pp. 313–323, 2015.
[33] O. Räsänen, "Generating hyperdimensional distributed representations from continuous valued multivariate sensory input," in *Proc. 37th Annu. Meeting Cognit. Sci. Soc.*, 2015, pp. 1943–1948.
[34] 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.

Denis Kleyko received the bachelor's (Hons.) degree in telecommunication systems and the master'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 pursuing the Ph.D. degree 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 high-dimensional computing, bioinspired cognitive architectures, and machine learning.

Evgeny Osipov completed a pre-doctoral programme at the EPFL Swiss Federal Institute of Technology, Lausanne, Switzerland, in 1999, and received the Licentiate of Engineering degree from the KTH Royal Institute of Technology, Stockholm, Sweden, in 2003, and the Ph.D. degree in computer science from the University of Basel, Basel, Switzerland, in 2005.

He is currently a Full Professor with the Dependable Communication and Computation Systems Group, Luleå University of Technology, Luleå, Sweden. He has authored or co-authored over 60 publications in computer science and engineering venues. His current research interests include cognitive computing and novel communication architectures applied to various scenarios of future cyber-physical systems and Internet-of-Things.

Prof. Osipov is a co-recipient of large research grants from the Swedish Research Council, the European Commission, the Swedish Research and Innovation Agency, and private foundations.

Alexander Senior received his B.Sc. and B.Eng. (Hons) degrees from Monash University, Clayton, VIC, Australia, where he is currently pursuing a Ph.D. degree.

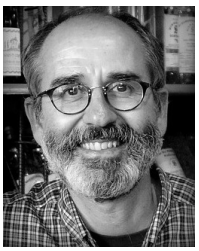
His current research interests include wireless sensor networks and distributed systems.

Mr. Senior has been privileged to participate in international research thanks to the STINT Grant by the Swedish Foundation for International Cooperation in Research and Higher Education.

Asad I. Khan received the bachelor's degree from the University of Engineering and Technology, Lahore, Pakistan, and the master's with distinction degree and the Ph.D. degree in engineering from Heriot-Watt University, Edinburgh, U.K.

He was appointed as a Lecturer with Heriot-Watt University, and held a senior IT management position with Monash University, Clayton, VIC, Australia, where he is currently a Tenured-Track Faculty Member with the Faculty of Information Technology. He has authored over 100 refereed publications, including two research monographs and several book chapters with three in Wiley's in 2010 The PROSE Award Honorable Mention Winning Title.

Dr. Khan is a co-recipient of large research grants from the Australian Research Council and the Department of Education Science and Training. He is a main foreign recipient of an Inter-Institutional STINT Grant by the Swedish Foundation for International Cooperation in Research and Higher Education. His work on parallel and bioinspired computing methods has led to several large research grants and the National HPC Award from the British Science and Research Council and leading industrial bodies in the U.K. and Australia.



Y. Ahmet Şekercioğlu received the B.Sc. and M.Sc. degrees from Middle East Technical University, Ankara, Turkey, and the Ph.D. degree from the Swinburne University of Technology, Melbourne, VIC, Australia, all in electrical and electronics engineering.

He held numerous positions as a Research Engineer with the private industry, prior to his academic career. He was a Lecturer with the Swinburne University of Technology for eight years. He was the Leader of the Applications Program of the Australian Telecommunications Cooperative Research Centre until the completion of the center's research activities in 2007. Until 2015, he was a member of the Academic Staff with the Department of Electrical and Computer Systems Engineering, Monash University, Clayton, VIC, Australia. He has established the Monash's Wireless Sensor and Robot Networks Laboratory. He is currently a Professorial Fellow with the Heudiasyc (Heuristics and Diagnostics for Complex Systems) Laboratory, Compiègne University of Technology, Compiègne, France. He leads a number of research projects on distributed algorithms for self-organization in mobile visual sensor and ad hoc networks, and networked robotics. He has authored over 130 papers in various forums, and graduated 15 Ph.D. and master's students (by Research) so far.