Informatics and Information Technologies

DOI: https://doi.org/10.15407/kvt208.02.005

RACHKOVSKIJ D.A.[1, 2], DSc (Engineering),
Chief Researcher, Dept. of Neural Information Processing Technologies,
Visiting Professor, Dept. of Computer Science, Electrical and Space Engineering,
https://orcid.org/0000-0002-3414-5334, e-mail: dar@infrm.kiev.ua
GRITSENKO V.I.[1], Corresponding Member of NAS of Ukraine,
Directorate Adviser, ORCID ID 0000-0002-6250-3987, e-mail: vig@irtc.org.ua
VOLKOV O.Ye.[1], PhD (Engineering), Director,
https://orcid.org/0000-0002-5418-6723, e-mail: alexvolk@ukr.net
GOLTSEV A.D.[1], PhD (Engineering), Senior Researcher,
Acting Head of the Dept. of Neural Information Processing Technologies,
https://orcid.org/0000-0002-2961-0908, e-mail: root@adg.kiev.ua
REVUNOVA E.G.[1], DSc (Engineering),
Senior Researcher, Dept. of Neural Information Processing Technologies,
https://orcid.org/0000-0002-3053-7090, e-mail: egrevunova@gmail.com
KLEYKO D.[3], PhD (Computer Science), Researcher,
https://orcid.org/0000-0002-6032-6155, e-mail: denis.kleyko@ri.se
LUKOVICH V.V.[1]
Researcher of the Dept. of Neural Information Processing Technologies
ORCID ID 0000-0002-3848-4712, e-mail: vvl97@ukr.net
OSIPOV E.[2], PhD (Computer Science), Professor,
https://orcid.org/0000-0003-0069-640X, e-mail: evgeny.osipov@ltu.se
[1] International Research and Training Center for Information
Technologies and Systems of the NAS of Ukraine and of the MES of Ukraine,
40, Acad. Glushkova av., Kyiv, 03680, Ukraine
[2] Department of Computer Science, Electrical and Space Engineering,
Lulea University of Technology, 971 87 Lulea, Sweden
[3] RISE Research Institutes of Sweden AB,
164 40 Kista, Sweden

NEURAL DISTRIBUTED REPRESENTATIONS FOR ARTIFICIAL INTELLIGENCE AND MODELING OF THINKING

Introduction. *Current progress in the field of specialized Artificial Intelligence is associated with the use of Deep Neural Networks. However, they have a number of disadvantages: the need for huge data sets for learning, the complexity of learning procedures, excessive specialization for the training set, instability to adversarial attacks, lack of integration with knowledge of the world, problems of operating with structures known as binding or composition problem. Over-*

coming these shortcomings is a necessary condition for advancing from specialized Artificial Intelligence to general one, which requires the development of alternative approaches.

***The purpose of the paper** is to present an overview of research in this direction, which has been carried out at the International Center for 25 years. The approach being developed stems from the ideas of N. M. Amosov and his scientific school. Connections to the Hyperdimensional Computing (HDC) and Vector Symbolic Architectures (VSA) field as well as to current brain research are also provided.*

***Results.** The concept of distributed data representation is outlined, including HDC/VSA that are capable of representing various data structures. The developed paradigm of Associative-Projective Neural Networks is considered: codevector representation of data, superposition and binding operations, general architecture, transformation of data of various types into codevectors, methods for solving problems and applications.*

***Conclusion.** An adequate representation of data is one of the key issues within the Artificial Intelligence. The main area of research reviewed in this article is the problem of representing heterogeneous data in Artificial Intelligence systems in a unified format based on modeling the neural organization of the brain and the mechanisms of thinking. The approach under development is based on the hypothesis of distributed representation of information in the brain and allows representing various types of data, from numeric values to graphs, as vectors of large but fixed dimensionality.*

The most important advantages of the developed approach are the possibility of natural integration and efficient processing of various types of data and knowledge, a high degree of parallel computing, reliability and resistance to noise, the possibility of hardware implementation with high performance and energy efficiency, data processing based on associative similarity search — similar to how human memory works. This allows one to unify the methods, algorithms, and software and hardware for Artificial Intelligence systems, increase their scalability in terms of speed and memory with an increase in data volume and complexity.

The research creates the basis for overcoming the shortcomings of current approaches to the specialized Artificial Intelligence based on Deep Neural Networks and paves the way for the creation of Artificial General Intelligence.

***Keywords:** distributed data representation, associative-projective neural networks, codevectors, hyperdimensional computing, vector symbolic architectures, artificial intelligence.*

INTRODUCTION

The current progress in the field of specialized Artificial Intelligence is associated with the use of Deep Neural Networks. However, they have a number of disadvantages: the need for huge data sets for learning, the complexity of learning procedures, excessive specialization for the training set, instability to adversarial attacks, lack of integration with knowledge of the world, problems of operating with structures known as binding or composition problem. Overcoming these shortcomings is a necessary condition for advancing from the specialized Artificial Intelligence to the general one, requiring the development of alternative approaches.

In 1960s, Nikolai M. Amosov formulated a hypothesis [1] about the mechanisms of information processing by the human brain that produce intelligent behavior. Those ideas were further developed in his subsequent works, including [2–5]. In fact, an approach was proposed to create Artificial Intelligence based on modeling the principles of human thinking and neural network organization of the brain. To develop and implement the approach, at the turn of the 1960s, the Department of Biological Cybernetics was founded at the Glushkov Institute of Cybernetics. Since 1997, the work of Amosov's school has continued at the department of Neural Information Processing Technologies of the International Research and Training Center for Information Technologies and Systems of the National Academy of Sciences and the Ministry of Education and Science of Ukraine (the International Center).

Fig. 1. The B-512 neurocomputer that had a 512-bit machine word length.

Initially, the developments of localist semantic M-networks [2–5] and assembly Hebb-like neural networks were carried out in parallel at the Amosov's department, resulting in the world's first autonomous robot controlled by neural networks in a natural environment [3]. In the late 1980s, Ernst M. Kussul proposed the foundations of the original paradigm of the Associative-Projective Neural Networks (APNNs) [5, 6]. The idea was to combine the hierarchical organization of Amosov's world model with the advantages of distributed representations, as well as with Hebb's cell assemblies. For the efficient implementation of APNNs, as a result of two projects in Japan jointly with Wacom, high-performance specialized neurocomputers were created using the Japanese element base, see Fig. 1. This development entered the history of Ukrainian informatics.

The article presents an overview of the research that have been carried out at the International Center for 25 years in the direction of developing the ideas of N.M. Amosov and his scientific school. Therefore, it is important to emphasize that this article is focused heavily on the results obtained from a single research group and, hence, it does not give the full credit to the related ideas and methods developed by other groups. We highlight some connections to Hyperdimensional Computing (HDC) and Vector Symbolic Architectures (VSA), as well as to brain research, however, for a comprehensive treatment of HDC/VSA and its connection to APNNs the readers are kindly referred to [7, 8].

DISTRIBUTED DATA REPRESENTATIONS

To represent data of various type, modality, and complexity, APNNs use distributed representations. They are based on modeling a “distributed” or “holographic” representation of information in the brain, as an alternative to “localist” representations [9]. In localist representations, each “object” (for example, a feature, a physical object, a relation, a complex structure, etc.) corresponds to a certain node (neuron), represented by a vector component, the dimension of which is equal to the number of neurons. In the distributed approach, an object is represented as “distributed” over a set of neurons. Distributed representation is a form of vector representation where each object is represented by a subset of vector components, and each vector component can belong to representations of

many objects. In distributed representation, the state of individual components of representation cannot be interpreted without knowing the states of other representation components. To be useful in applications, the distributed representations of similar objects must be similar (by some measure of similarity of the corresponding vectors — for example, by the value of the dot product or cosine of the angle between the vectors).

Distributed representations possess the following advantages:

– high information capacity. For example, if one object is represented by M binary components of a D-dimensional vector, then the number of representable objects is equal to the number of combinations M from D, in contrast to D/M in localist representations;

– direct access to the representation of the object. A distributed representation of a complex structure can be processed directly, without tracing pointers or connections required in symbolic or localist representations;

– an explicit representation of similarity. Similar objects have similar representations that can be directly compared using efficiently computable vector similarity measures (e.g., dot product, Minkowski distance, etc.);

– a rich semantic basis, which is provided through the direct use of representations based on features and the possibility of representing the similarity of the features themselves in their vector representations;

– for many types of distributed representations – the ability to recover the original representations of objects;

– the ability to work in conditions of noise, malfunction, and uncertainty, as well as neurobiological plausibility.

Since distributed representations of various objects are vectors, a rich arsenal of methods developed for vector data can be applied to their processing.

It was believed that the main disadvantage of distributed representations is the inability to represent the structure [10]. However, various researchers have developed a number of "structure-sensitive" distributed representations in various formats [10–13]. Such distributed representations are called hypervectors, and models based on them are called Hyperdimensional Computing (HDC) or Vector Symbolic Architectures (VSA) [7, 8]. The dimension of hypervectors is large, usually more than 1000, and reaches hundreds of thousands or more. The main operations on hypervectors are superposition, used to combine multiple hypervectors, and binding, used to associate them in representing structures. In various hypervector models, these operations are implemented in different ways, but the dimensionality of hypervectors at the input and output of these operations does not change.

THE ASSOCIATIVE-PROJECTIVE NEURAL NETWORKS

In APNNs, we use binary hypervectors with components from $\{0,1\}$. Moreover, those are sparse vectors, that is, the proportion of their non-zero components is small. This data representation format allows an efficient processing. Historically, and to distinguish from other hypervector types, we call such hypervectors as "codevectors", and will use the term throughout the text of this article.

As a superposition operation, component-wise disjunction of codevectors is used. When a set of codevectors is represented by a component-wise disjunction, the presence of some single component in the resulting codevector is determined by the

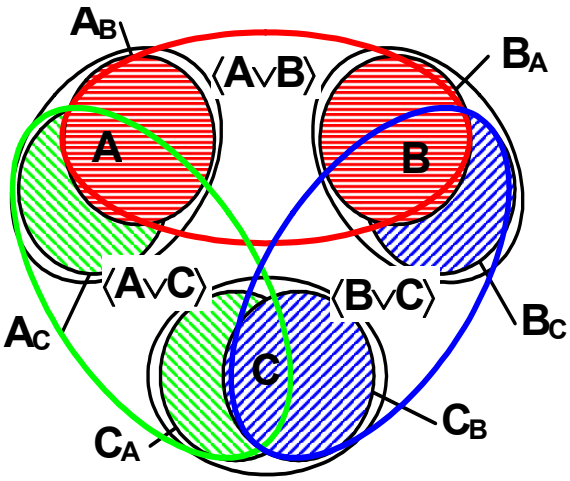

Fig. 2. Binding via the Context-Dependent Thinning. Smaller ovals represent the 1-components of the codevectors **A**, **B**, **C**. Larger ovals show the codevectors ⟨**A** ∨ **B**⟩, ⟨**B** ∨ **C**⟩, ⟨**A** ∨ **C**⟩ bound by CDT. Circles $\mathbf{X}_\mathbf{Y}$ denote the subset of 1s preserved in the codevector **X** when **Y** is also present. It can be seen that, e.g., $\mathbf{A}_\mathbf{B}$ and $\mathbf{A}_\mathbf{C}$ are different subsets of **A.** Note that actually the 1-components belong to randomly generated codevectors.

presence of such a component in at least one codevector of the set. Thus, the individual components do not contain information about the combination of codevectors in the set. Binding operations are used in HDC/VSA to preserve this information.

For binding codevectors, Context-Dependent Thinning procedures were proposed [10]. In one version of this procedure, the bound codevector ⟨**Z**⟩ is formed from the codevectors $\mathbf{X}_i$ to be bound as follows:

$$\mathbf{Z} = \vee_i \mathbf{X}_i;\ \langle \mathbf{Z} \rangle = \vee_{k=1,K} (\mathbf{Z} \wedge \tilde{\mathbf{Z}}(k)) = \mathbf{Z} \wedge \vee_{k=1,K} \tilde{\mathbf{Z}}(k).$$

Here $\tilde{\mathbf{Z}}(k)$ is **Z** with permuted components. For each k, a random independent permutation is used which is fixed for the specific k. It is also possible to use the same permutation multiple times.

The subset of 1-components of each codevector $\mathbf{X}_i$ that is preserved in ⟨**Z**⟩ depends on **Z**, and hence on each and all $\mathbf{X}_i$, as shown in Fig. 2. Thus, information is stored about a specific set of elements of the set, the codevectors of which participated in the formation of ⟨**Z**⟩, ensuring binding.

The number K of used permutations controls the number of 1-components in the final ⟨Z⟩. Thus, it is possible to normalize the number of 1s, i.e., degree of sparsity, in the resulting codevector. Note that the operation of component-wise conjunction also provides binding, but increases the degree of sparseness of the resulting codevector, that might be an undesirable feature of binding operation.

The general APNN architecture. The APNN architecture was proposed and developed in [6, 14–18]. It is generally recognized that for a reasonable common-sense behavior, an intelligent agent needs a model of the world that includes knowledge specific to the subject area, as well as information about the agent itself. Such a model allows the agent to understand the world and helps it in its interaction with the world, for example, by predicting the results

of actions. The goal of developing APNNs is to offer an approach to create a complex hierarchical model of the world that supposedly exists in the brain of humans and higher animals, as a step towards the Artificial General Intelligence. Two types of hierarchy are considered in APNNs: compositional (part-whole), as well as classification or generalization hierarchy (class-member or is-a). An example of a compositional hierarchy: letters → words → sentences → paragraphs → text. An example of a classification hierarchy: apple → big red apple → this big red apple in the hand.

The APNN model of the world is based on models of objects of various modalities, including sensory (visual, acoustic, tactile, motor, etc.) and more abstract modalities (linguistics, planning, reasoning, abstract thinking, etc.), which are hierarchically organized. Models should exist for objects of different nature, for example, events, objects, feelings, features, etc. Models (their representations) of different modalities can be combined, which leads to multimodal representations of objects and their associations with behavioral schemes (reactions to objects or situations), see [6, 14–18].

The APNN architecture is based on fixed-dimensional codevectors for objects of varying complexity and generality, which represent various heterogeneous data types, for example, numeric data, images, sequences, graphs (Sec. 4). Codevectors can be formed "on the fly" (without training). APNNs have a multi-module, multi-level and multi-modal structure, see Fig. 3 and [18] for more details. The modules are connected by the bundles of projective one-to-one connections that just copy codevectors between the modules. The architecture includes the modules of independent modalities and sub-modalities. For example, independent visual features of an object such as shape, size, texture, color are represented by codevectors in the modules M11...M14. A codevector of a visual model of an object is formed from these feature codevectors in the module M21. In the modules M22...M25, the codevectors of acoustical, tactile, olfactory, taste models are formed. The codevector of an integral multi-modality sensory model of an object is formed in the module M31, to which its name codevector could be added from the module M32. Probably, name could be a feature in the models of all modules.

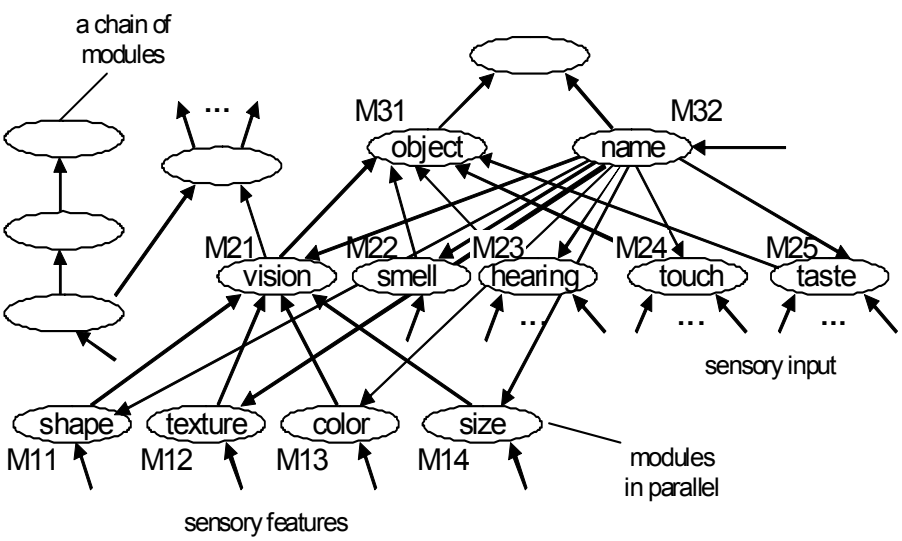

Fig. 3. An example of the APNN architecture.

The modules form, store and process a set of codevectors representing object models of a certain modality and a certain level of the compositional hierarchy. Module's codevectors are constructed from codevectors obtained from other modules, such as lower-level modules of the same modality, or from modules of other modalities. The lower level of the compositional hierarchy consists of modules that provide interface (of representations) with the external environment. A codevector is similar to the codevectors of its elements of the lower levels of the compositional hierarchy, as well as to the codevectors of higher levels, for which the codevector is an element. Thus, using similarity search in the memory of lower- and higher-level elements, it is possible to recover both the codevectors of the elements of lower levels and the compositional codevectors of higher levels. Similarity search in the memory of single module allows finding similar objects.

So, each module should have a long-term memory where it stores its codevectors. It was proposed to use distributed auto-associative memory of the Hopfield type as the module's memory [19, 20, 21, 22]. Let us consider it in some more details.

Associative memory and the generalization hierarchy. One of the key modes of processing codevectors is similarity search, i.e., search in the database (set) for a codevector most similar to the query codevector. This can be effectively done using a neural network distributed auto-associative memory with binary connections (an auto-associative version of the Willshaw memory). Each memory neuron (corresponding to the codevector component) is connected to all other neurons by a connection with a weight of 0 or 1. Each codevector is memorized according to the binary version of the Hebbian rule: the weight of the connection between memory neurons corresponding to the 1-components of the codevector is set to 1. If the weight of the connection is already 1, it does not change.

After the set of codevectors is memorized, a codevector-query is given as the input, which may not belong to the stored set of codevectors (for example, it may include only a part of the known components of the codevector and/or noise). It is multiplied by the matrix of connections with a subsequent binarization by comparison with the threshold. The result is again fed into the input. After several such iterations of evolving the memory, the output codevector is retrieved which is the codevector from the memorized set that is most similar to the input codevector.

We have developed a method for analyzing the characteristics of such a memory [20, 21, 19]. For a wide range of codevector dimensionality, the degree of sparsity, and the level of distortion of codevectors-queries, it was shown that the accuracy of the obtained estimates of information characteristics exceeds the accuracy of the Gibson-Robinson method, and the maximum information efficiency of this memory (per bit of connection implemented in computer memory) is higher than that of the Hopfield network.

Note that the studied learning rule does not allow using this auto-associative memory for generalization. According to Hebb's ideas about cell assemblies, active neurons often found together (corresponding to 1-components of codevectors), when learning by increasing the weight of connections between them, form the "cores" of assemblies, i.e., strongly connected subsets of neurons that fire together more easily. This may correspond, for example, to typical category features and prototype objects. And vice versa, rare combinations of active neurons form a "fringe" corresponding, for example, to features of specific objects, see [19]. To implement this, connections between neurons should be not binary, but gradual, as in the Hopfield network. An-

other option is to use not a deterministic, but a stochastic learning rule, in which the weight of the connection between 1-components changes with a certain probability, which is set by the "learning rate" parameter [14].

Modeling the formation of cores and fringes in distributed auto-associative memory has so far been investigated only fragmentarily. The same applies to the use of such memory for codevector representations of sequences and structures (but see [16]). These topics are a promising direction for further research [14, 16, 23].

The part-whole hierarchy. In APNNs, the formation of the codevector of an object-whole from the codevectors of its objects-parts or objects-features is performed by superposition or Context-Dependent Thinning operations. A number of questions remains open:

- how to extract objects and their parts of different hierarchical levels?
- how to determine which hierarchical level an object belongs to?
- how to work with an object that can belong to different levels of the hierarchy and modules?

In this regard, it is of great interest to explore possible parallels with how this is done in the brain.

It was shown in [24], that the representation of composite objects is different from the representation of their parts, and there exists a representation in the brain that corresponds to a combination of parts. Moreover, in the process of learning new objects a representation of the object appears which is processed as easily as the representation of a separate feature (so-called unitization).

In [25], a memory model is confirmed in which both the features of the object and the object in the form of bound features are presented. Moreover, the features of an object can be bound not only through their common position, but directly with each other. In addition, it has been found that unbound features can be represented with greater resolution than when they are bound.

In [26], a hierarchy of episode representations is considered, in which the levels of objects, events, and narratives are distinguished. Moreover, the representations of both objects and events also have a hierarchical structure: there exists a representation of both the object-whole and its features [25], and events are represented both by their details as well as by coarser global information. Perhaps there are different mechanisms at work to memorize these different levels of hierarchy. This is manifested in the different nature of forgetting: events are forgotten as a whole, and objects can be forgotten partially, by separate features. In addition, more generalized information is memorized better than details.

Experiments in [27] made it possible to propose an episode recall model in which events can "scroll" forward until the beginning of the next event. In [28], it was experimentally shown that the representation of an object can be bound with both time and position.

The key problem for APNNs is the segmentation of objects (events) into parts and wholes. Progress in its solution can be facilitated by data on the solution of such tasks by the brain. It was shown in [29] that neural states at the upper levels of the hierarchy are activated longer than their parts from the lower levels. This is consistent with the APNN architecture.

When modeling the brain, the representation of spatial structures is usually considered in terms of either cognitive maps or cognitive graphs (see review [30]). In cognitive maps, locations are given by coordinates, and the relationships

between them can be viewed in terms of angles and distances, as on a map. In cognitive graphs, only some positions are given by nodes, and the edges between them are path segments, with no information about position or orientation relative to the global coordinate system. Only the topology can be specified (nodes are connected, but the path can be winding) or also labels (distances, directions, angles for existing edges). In [30], it is proposed that representations in the form of cognitive maps and graphs can exist simultaneously, complementing each other, and can also be used to represent not only spatial knowledge. In APNNs, representations of both these data types has been developed [31–33], that can be used both for brain modeling and in practical problems.

Some connected research directions. The representations and basic operations in HDC/VSA provide Turing-completeness. In [34–37], a paradigm different from HDC/VSA that operates with distributed representations is proposed, which is also Turing complete. It is formulated in terms of Hebb assemblies and focuses on the direct handling of assemblies in memory.

To create a "copy" of assembly in a target area (projection operation), a pattern of active neurons is formed in it by randomly projecting the activity pattern of the assembly from the original area and selecting the most active neurons. Note that we considered a similar transformation in [38–41], see also [42]. Then the resulting pattern of active neurons forms a new assembly using variants of the Hebbian rule. This process is complicated by the possibility of modifying the connection weights of a random projection and spreading activity along connections in the target area.

The possibility of back projection is also considered with modification of projection connections from the second area to the first one (bind operation), and merging of assemblies from the two areas to the third one. The last two operations can be used to form assemblies of structures, being analogous to binding and superposition in HDC/VSA. It is interesting to explore how the capabilities of this paradigm relate to the HDC/VSA models, both in terms of applications and biological relevance.

Sketches (see [43–46] and references therein) are compressed representations of data. In [47–48], it was proposed to combine sketches with Deep Neural Networks. Both the formation of sketches of hierarchical structures and the use of memory based on locality-sensitive hashing ([43–46] and references therein) for fast similarity search at each layer of Deep Neural Networks are considered. Random projection is used to form the sketch. The authors of [47–48] consider the problems of recovering input sparse vectors and random matrices themselves, using the projection results. The sparse recovery methods and dictionary learning are used. On the one hand, this aims to overcome the limitations of HDC/VSA related to the lack of learning. On the other hand, this complicates the scheme and introduces additional restrictions related to the formation and handling of sketches.

A theoretical analysis of binding operations other than CDT is considered in [49].

INPUT DATA TRANSFORMATIONS

The key problem in using codevectors to solve practical problems is to obtain them from the input data in such a way that similar codevectors correspond to similar input data (objects). We have developed such transformations for various types of data.

Sets (i.e., collections of elements without specifying an order or other structure) have the simplest codevector representation. Each element of the set is assigned a randomly generated codevector. The codevector of a specific set consisting of specific elements is obtained by superposition (component-wise disjunction). Sets containing the same elements will have similar codevectors. The greater the proportion of identical elements, the greater the similarity.

Numeric data. Numeric data — scalars and vectors — are perhaps the most common data type. We have developed and investigated three types of methods for transforming real vectors $a \in \Re^D$ into codevectors $A \in \{0,1\}^d$.

1. Receptive fields-based methods [45, 46].

The components of a codevector are formed by determining whether the input vector belongs to the receptive fields corresponding to the components of the codevector: $A_i = \psi_i(\mathbf{a})$, $i = 1,...,d$, where A_j are the binary components of the codevector $\mathbf{A}$, $\psi_i(\mathbf{a})$ is the indicator of the vector $\mathbf{a}$ location in some region of the input space, i.e., in the i-th receptive field.

The developed and investigated methods use hyperrectangular receptive fields with random boundaries in random subsets of the dimensions of the input space. This allows a computationally efficient determination of whether a vector belongs to a receptive field by comparison with the boundary values of the field in each of its dimensions. Significant overlap of codevectors of close input vectors is provided due to the presence of a large number of common receptive fields, i.e., due to the vectors falling into a large number of the same receptive fields.

2. Random projection-based methods [43, 44, 46].

The codevector $\mathbf{A}$ is formed by a random linear projection of the input vector $\mathbf{a}$ using a random matrix $\mathbf{R}(d \times D)$ and binarization of the resulting vectors by a component-wise non-linear threshold function T: $\mathbf{A} = T(\mathbf{Ra})$, see Fig. 4. Note that both $d >> D$ and $d << D$ could be the case. Here the matrix $\mathbf{R}$ is a random matrix with elements from a sub-Gaussian distribution. In particular, the Gaussian distribution, ternary one with elements from the set $\{-1, 0, 1\}$, and binary one with elements from $\{0, 1\}$ were studied. Such a transformation can be performed using a perceptron-like neural network with randomly selected connections. Threshold for non-linearity does not have to be zero, and it allows controlling the sparseness of the generated codevectors. Significant overlap of codevectors of close input vectors is provided by a large value of their dot product with the same random vectors — rows of a random matrix. Random projection properties allow estimating the cosine of the angle and the angle of the input vectors from such codevectors. Estimating the Euclidean distance and dot product requires knowledge of the Euclidean norms of the input vectors. A similar neural network architecture was found in the olfactory system of the Drosophila fly [42, 50, 51]. Related problems were also studied in [52, 53].

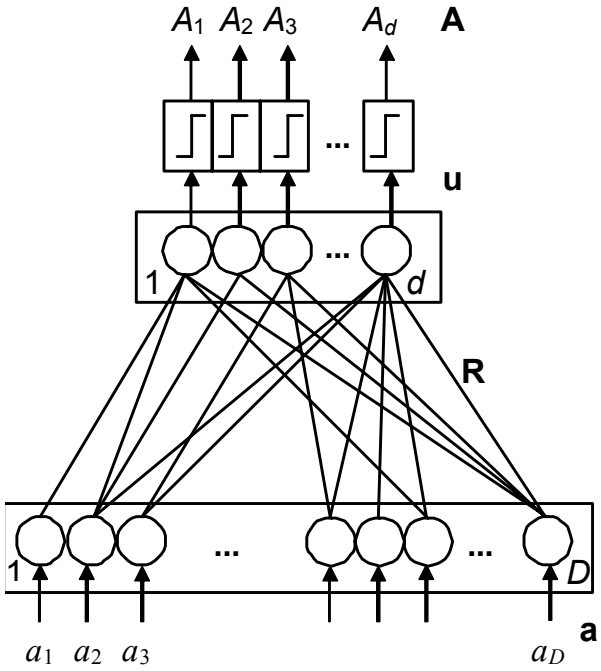

Fig. 4. A single layer perceptron for transformation of vector data by a random projection with a threshold.

3. Compositional methods [44, 45, 54].

For each value of each component-scalar $a(j)$, $j = 1,...,D$, of the input vector **a** with integer components, a codevector is formed, with similar codevectors corresponding to close values of the scalars, and dissimilar ones corresponding to distant values. The codevector **A** of the input **a** is formed from the codevectors $\mathbf{A}_{ja(j)}$ of the values of its components-scalars $a(j)$ как $\mathbf{A} = \langle \mathbf{A}_{1a(1)}, \mathbf{A}_{2a(2)},..., \mathbf{A}_{Da(D)} \rangle$, where $\langle\cdot\rangle$ is the CDT operation. A degenerate case of binding is the component-wise disjunction, in which case, in fact, binding does not occur. The dimension of codevectors for scalars and vectors is the same. Significant overlap of codevectors of close input vectors is ensured by constructing them from similar codevectors of the values of their components.

For some of these methods, the dependence of the probability of coincidence of the components of the codevectors on the value of the components of the input vectors was obtained, providing the similarity function that can be estimated from the dot product of the codevectors.

For all three types of methods, an analysis of the characteristics of codevectors was carried out, which makes it possible to choose their parameters in applications. Note that the non-linearity of the transformation of the input space into codevectors makes it possible to use linear models to solve nonlinear problems using codevectors.

The codevector representation of vector data is used not only in similarity search and classification problems (Sec. 5). We note the use of the method of receptive fields for continuous optimization [55, 56] and the method based on random projection in problems of decentralized flows [57].

Neural regularization approach. Numerical methods used to process vectors and matrices in statistics, machine learning, and inverse problem theory often turn out to be inefficient for large dimensions. This manifests itself both in an increase in computational costs and in the loss of stability of solutions. A productive approach to overcome these problems is the randomization approach. It allows not only to reduce computational costs when searching for solutions, but also, as it turned out, to give stability to numerical methods, see also [58–61].

To improve the stability and accuracy of solving discrete ill-posed inverse problems (DIP), we have developed new methods of neural regularization based on random projections, as well as on the basis of matrix expansions. The methods use an integer regularization parameter that determines the complexity of the linear model. Computable criteria have been developed for choosing the value of the regularization parameter that is optimal from the point of view of the accuracy of solving a discrete ill-posed problem, i.e., of the recovery of an unknown input signal. The application of the developed methods provides not only the stability and accuracy of the solution, but also reduces the computational complexity of the regularization. Our studies of the regularizing properties of random projections started in 2009 [62] and were further developed in [63–67].

An approach and methods for solving DIP based on random projection have been developed. To do this, it was proposed to left-multiply both parts of the approximate equation $\mathbf{Ax} = \mathbf{y}$ by a random matrix $\mathbf{R}_k \in \Re^{k \times N}$ with the number of rows k less than N. The vector of the recovered signal is obtained by multiplying the pseudo-inverse matrix $(\mathbf{R}_k\mathbf{A})^+$ by the right part $\mathbf{R}_k\mathbf{y}$ of the new equation. An experimental study of this basic method showed that averaging over random matrices leads to smoothing of the dependence of input and output recovery errors on k, as well as to a decrease in the number of local minima. As a result, it is easier to find the optimal value of k. This leads to a reduced computational complexity by restricting the search to $k_{\text{opt}+1}$ ($k_{\text{opt}} < N$) values of the criterion that allows getting optimal k (instead of calculating all N values of the criterion). In addition, the error of the recovery of the input vector is reduced relative to the case without averaging.

This gave us reason to believe that analytical averaging over random matrices can give the same useful result. The analytical averaging allowed us to propose a method of the "Deterministic Random Projection" for solving DIP, the error of which is always less than the error of the basic random projection method. Namely, the recovery of the input vector was proposed to be carried out as $\widetilde{\mathbf{x}} = \mathbf{A}^{\mathrm{T}}\mathbf{U}\mathbf{D}_k\mathbf{U}^{\mathrm{T}}\mathbf{y}$ [66], leading to error reduction by the value of variance that appears due to multiplication by a random matrix. Here $\mathbf{D}_k$ is a diagonal matrix of special kind corresponding to $\mathbf{R}_k$, see [66], $\mathbf{U}$ (and $\mathbf{V}$ below) is the matrix of the left (and right) singular vectors.

It was shown that the considered methods of solving DIP (the Tikhonov regularization, the Truncated Singular Value Decomposition, and the Deterministic Random Projection) weigh reciprocal singular values differently when obtaining the solution vector. The expression for estimating the input vector in the general case has the form $\mathbf{x}^* = \mathbf{V}diag\left(\frac{1}{s_i}w_i\right)\mathbf{U}^{\mathrm{T}}\mathbf{y}$. For the Tikhonov regularization, it is known that the weights decrease gradually:

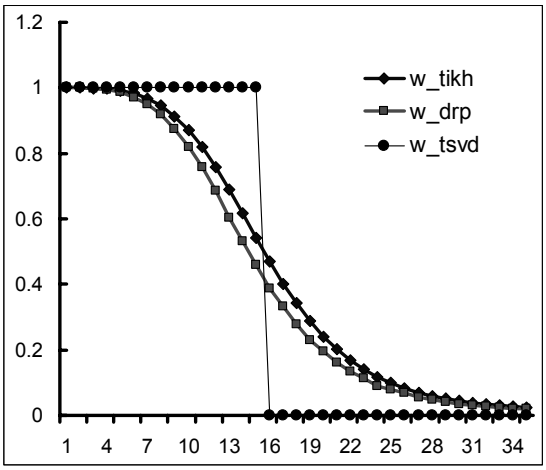

Fig. 5. Examples of the weights of the reciprocal singular values when estimating the input vector by the Tikhonov (w_tikh), the Deterministic Random Projection (w_drp), and the Truncated Singular Value Decomposition (w_tsvd) types of regularizations.

$$\mathbf{x}_{\text{Tikh}} = (\mathbf{A}^{\text{T}}\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^{\text{T}}\mathbf{y} = \mathbf{V}diag\left(\frac{s_i^2}{(s_i^2+\lambda)}\frac{1}{s_i}\right)\mathbf{U}^{\text{T}}\mathbf{y},\ w_{\text{Tikh}} = \frac{s_i^2}{(s_i^2+\lambda)}.$$

For the Truncated Singular Value Decomposition, the weights of $1/s_1 \ldots 1/s_k$ are equal to 1, whereas the weights for $1/s_{k+1} \ldots 1/s_N$ are equal to 0. For the Deterministic Random Projection,

$$\mathbf{x}_{\text{DRP}} = \mathbf{A}^{\text{T}}\mathbf{U}\mathbf{D}_k\mathbf{U}^{\text{T}}\mathbf{y} = \mathbf{V}\mathbf{S}\mathbf{D}_k\mathbf{U}^{\text{T}}\mathbf{y} = \mathbf{V}diag\left(s_i^2 d_{ki}\frac{1}{s_i}\right)\mathbf{U}^{\text{T}}\mathbf{y},\ w_{\text{DRP}} = s_i^2 d_{ki}.$$

Here d_{ki} are the diagonal elements of $\mathbf{D}_k$. So, the values of the weights gradually decrease with the increasing index of the singular value, see Fig. 5. This is similar to the behavior of weights for the Tikhonov regularization and provides a potentially higher accuracy compared to the Truncated Singular Value Decomposition, due to a better signal approximation.

Structured data. A number of methods for codevector representation of data with the sequence structure have been developed in [68, 69, 31]. They are based on the formation of codevectors which represent the elements of a sequence according to their positions. These codevectors are combined into a codevector of the whole sequence either by a superposition operation (componentwise disjunction) or by a binding operation. A widespread variety of sequences are strings, where the elements are symbols at sequential positions.

The following approaches have been developed to represent sequence elements taking into account their position:

1) multiplicative binding of codevectors of sequence elements with codevectors of their positions, as well as with codevectors of context elements [70, 16];

2) the use of permutations of codevectors of sequence elements to represent their order [71, 12];

3) representation by means of codevectors of n-grams (i.e., *n* consecutive elements) of a sequence [69, 72].

In the first approach, to represent identical or similar sequence elements at close positions with similar codevectors, correlated codevectors have been proposed as codevectors for close positions. In the second approach, to preserve the similarity of element codevectors at close positions, it was proposed in [73] to use partial (correlated) permutations. In the third approach, the representation of n-grams and whole strings by multiplicative binding of symbol codevectors permuted according to their position was proposed in [74]. In [69], it was considered how to approximate the edit distance of strings by vectors of frequencies of different n-grams, allowing a codevector implementation.

The recently proposed methods [31, 75] made it possible to form similar codevectors for sequences with similar elements at close positions. These methods, in contrast to the previous ones, allow obtaining codevectors of transformed (shifted) sequences not only by their formation from the transformed sequences, but also by transforming the codevector of the original sequence (the equivariance property).

In addition to the 1D case of sequences, several approaches have been developed for the codevector representation of objects with a two-dimensional structure, such as 2D images.

In the role-filler approach, codevectors of positions are formed in such a way that similar codevectors correspond to close positions. The features extracted from the images are also associated with codevectors. A feature at its position is represented as a codevector obtained by binding the codevectors of the feature and its position. The codevector of the entire image is obtained by superposition of the codevectors of all extracted features at their positions. Such codevector representations are similar for images that contain similar features at similar positions.

In the permutation-based approach, the representation of the feature codevector at its position is performed by permutations, and the resulting codevectors are superimposed to represent the entire image. The use of partial permutations makes it possible to ensure the similarity of codevectors of a feature at close positions [73].

Arbitrary binary features can be used as features. So, for the direct formation of codevectors from images, special binary LIRA features were proposed [76, 77]. For binary images, each LIRA feature is an indicator of the presence of 1/0 pixels at randomly selected positions of a randomly located local window. For gradual images, the brightness of pixels is compared against randomly selected thresholds. Each LIRA feature corresponds to a codevector component. The parameters are chosen so that the image usually contains a small fraction of the entire set of features, i.e., codevector is sparse.

The RLD features (Random Local Descriptors) can be considered as the development of LIRA features. In the RLD approach [73], the same set of features is extracted at each “interesting” point in the image. Each feature is assigned a random codevector. Features at their positions are represented by partial permutations of the corresponding codevectors. As a result, features at close positions produce similar codevectors. Due to this, RLD improves the results of LIRA in classification problems. Currently, methods for generating image codevectors are being developed that provide equivariance to some image transformations.

Complex structured data, such as hierarchically organized graphs of knowledge base episodes, are initially described by systems of hierarchically organized objects and relations. Based on the approaches from the previous sections, a number of methods for codevector representation of such data have been developed, their basic

blocks being codevector representations of relations. The codevectors of relations are recursively transformed into codevectors of complex hierarchical structures containing higher-order relations [10, 11, 18, 32, 33]. The codevectors of structures have the same dimension as the codevectors of their elements.

Consider an example of the Solar System episode representation (please see details in [33]) shown in a bracketed notation in Fig. 6 and as a graph sketch in Fig. 7. It includes objects Sun, Planet; attributes Mass, Temperature; relations Gravity, Attracts, Greater, Revolve-Around, And; higher-order relations Cause. Relations have arguments; the relation's name together with the particular arguments form the instance of a relation. For many types of relations, the order of arguments is essential. So, a relational instance can be described by roles specific to the relation and instances of arguments that fill them. To form the codevector of a relational instance, the role-filler approach uses a multiplicative binding of the role and filler codevectors to indicate which role the filler is assigned to in a relation (e.g., the second role in Greater is filled with a smaller object instance). The codevector of the whole episode is formed as shown in Fig. 8. In the predicate-arguments approach, random permutations of the codevectors of the arguments of a relation are used to represent their order, as in the representation of sequences.

```
(CAUSE
        (GRAVITY (MASS SUN) (MASS PLANET))
        (ATTRACTS SUN PLANET) )
(GREATER ( TEMPERATURE SUN)
        ( TEMPERATURE PLANET) )
(CAUSE
        (AND  (GREATER (MASS SUN)
                       (MASS PLANET) )
              (ATTRACTS SUN PLANET) )
        (REVOLVE-AROUND PLANET SUN) )
```

Fig. 6. Bracketed notation description of the Solar System episode

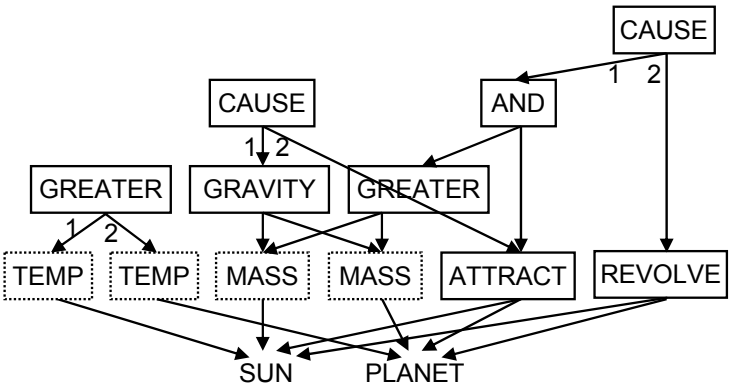

Fig. 7. Graph description of the Solar System episode.

SOLAR_SYSTEM =
⟨ **CAUSE_1** ∨ ⟨**GRAVITY_1** ∨ ⟨**MASS** ∨ **SUN**⟩ ⟩ ∨ ⟨**GRAVITY_2** ∨ ⟨**MASS** ∨ **PLANET**⟩⟩ ⟩
∨ ⟨**CAUSE_2** ∨ ⟨**ATTRACTS_1** ∨ **SUN**⟩ ∨ ⟨**ATTRACTS_2** ∨ **PLANET**⟩ ⟩
∨
⟨ **GREATER_1** ∨ ⟨**TEMPERATURE** ∨ **SUN** ⟩⟩
∨ ⟨ **GREATER_2** ∨ ⟨**TEMPERATURE** ∨ **PLANET**⟩ ⟩
∨
⟨ **CAUSE_1** ∨
⟨ **AND** ∨ ⟨**GREATER_1**∨⟨**MASS**∨**SUN**⟩⟩ ∨ ⟨**GREATER_2**∨⟨**MASS**∨**PLANET**⟩⟩ ⟩
∨ ⟨ **AND** ∨ ⟨**ATTRACTS_1** ∨ **SUN**⟩ ∨ ⟨**ATTRACTS_2** ∨ **PLANET**⟩⟩ ⟩
∨ ⟨ **CAUSE_2** ∨
⟨ **REVOLVE-AROUND_1** ∨ **PLANET**⟩ ∨ ⟨**REVOLVE-AROUND_2**∨**SUN**⟩ ⟩

Fig. 8. Codevector representation of the Solar System episode.

The similarity of codevectors of similar complex structured data is ensured by the similarity of the codevectors of their elements and the properties of binding operations. As a result, for structures with similar objects and relations, the created methods ensure the production of similar codevectors. Thus, by finding the similarity of the codevectors of relational structures using some vector similarity measure of vectors (for example, the dot product), we simultaneously evaluate the similarity of structures and the similarity of objects in these structures. This provides the basis for the creation of computationally efficient and qualitatively new methods for processing relational structures of data- and knowledge bases, which are based on similarity and simultaneously take into account both the structure and semantics of knowledge.

METHODS

Based on the proposed approaches, methods for solving various types of problems from the field of Machine Learning and Artificial Intelligence have been developed.

The approach of "example-based reasoning" [45, 46] is productively used by humans when solving problems of natural intelligence; it is also used for solving a wide range of problems in Artificial Intelligence systems. For inferences about a query (an input object or a situation), this approach uses a search for similar known objects or situations with which additional information is associated. In computer implementation, a base of objects-example is formed, i.e., a memory containing the past "experience" of the system. Examples found by similarity search in memory can be used directly, or as a source of additional information about the input object-query. One example of using this approach to solve classification problems is the nearest neighbor method, where the class label of the nearest (by some measure of similarity) example from the memory is transferred to the query. Another example is a linear classifier, where for each class, in the process of learning a vector representation of the generalized example is formed, and the classification is performed by choosing the class whose generalized example gives the maximum value of the dot product with the vector representation of the query.

The effectiveness of applying the example-based reasoning approach to solving problems depends on how the similarity between examples is defined. When using codevectors, this, in turn, depends on the methods of their formation and the applied similarity measures of codevectors. We also note the existence of efficient algorithms for fast similarity search [78, 79, 46].

The developed methods of codevector representation of data of various types make it possible to solve classification problems using linear vector classifiers. In particular, for vector data, often the class boundaries are not linearly separable. However, non-linear data transformation to codevectors overcomes this problem. Combining a specific transformation of input data into codevectors with a specific type of linear classifier yields a new version of the classifier [80–84]. The best-known linear classifiers are perceptrons and Support Vector Machines. We have proposed a perceptron with a large margin that approximates the results of Support Vector Machine, but is trained incrementally and fast [84]. As mentioned above, other types of classifiers can be used, such as the nearest neighbor classifier.

The architecture of a modular neural network with an assembly organization has been developed that can be considered as a generalization of the perceptron classifier [83]. Each module corresponds to a class. In the fully connected version, each neuron is connected by trainable connections with all other neurons of the module, and in the non-fully connected version, with a randomly selected subset of neurons. Learning is performed similarly to the perceptron learning rule, i.e., by increasing the weights of connections between active neurons in the module corresponding to the correct class, and decreasing the weights of connections in the module of the wrong class. In the recognition mode, neural activity is propagated along the connections in each module, and the activity of the module is calculated as the total sum activity of the module's neurons. The module with the highest activity determines the winning class.

Also, the architecture of a layered neural network with competitive layers for image processing has been developed. Each layer corresponds to a certain class of images and is a separate neural network. Neurons from different layers have a one-to-one correspondence with each other and with the 2D input retina. There is a competition between the corresponding neurons of all layers resulting in the activation of single most active neuron among all layers in each retina's position. Such networks have been applied to the problems of classifying handwritten digits, texture segmentation, and extracting image segments of different orientations [85, 86].

Concerning texture segmentation, a method has been developed for segmenting visual images into homogeneous regions of fine-grained texture [87, 88]. The peculiarities of the method are that it works without training, and no preliminary information about the analyzed image is required.

Work is underway to use the developed methods in tasks related to Unmanned Aerial Vehicles [89, 90].

APPLICATIONS

The effectiveness of the developed methods has been demonstrated by solving a wide range of applied problems. Upon the time of the original publications, a number of the results obtained were comparable to the state-of-the-art as indicated below.

To take into account the semantics of textual information (words, texts and their fragments) presented in the form of frequency vectors of the joint occurrence of words and their contexts, methods for the formation of "context codevectors" have been developed. In the task of searching for textual information, due to taking into account semantics, the accuracy of the search was increased up to 20% on the text datasets Time, Cranfield, Medlars [91]. Context codevectors have also been applied in tasks that require taking into account the semantic

similarity of words, including the search for synonyms and the choice of the proper target word in automatic translation.

Strong performance has been obtained in solving problems of classification and segmentation of textures [15, 86], recognition of handwritten characters and words [84, 92], acoustic signals [81, 93], etc. In text classification, on the Reuters-21578 dataset the accuracy was increased to 0.937 corresponding to the best contemporary results. In predicting the sensitivity of a cancerous tumor of glioma to chemotherapy, the prediction accuracy was improved to 88.5% compared to less than 81% for other methods. A classification accuracy of up to 99.5% has been achieved on the MNIST handwritten digits dataset [73, 76, 77, 84, 92]. It was shown that for a small feature set, a non-fully connected modular network improved the results of the perceptron classifier.

A high-precision system for recognizing a speaker by voice in noisy conditions has been created [93]. The individual features of the voices were fixed in the structure of the neural network in the process of training neural network classifiers. The network automatically generated individual portraits of voices as a collection of speech features. The system worked both in search mode in voice databases and in real time. The reliability of identification of microphone signals was within 94%–98% and reached 85%–94% for phone signals. The technology worked with the datasets that included an arbitrary number of voice samples from various people recorded via microphone and phone channels. The ability to search for defined voices in many hours of audio recordings was also implemented (audio data indexing).

Codevector representations of data with the structure of sequences have been tested in the tasks of detecting spam and intrusions in computer systems, classifying coding regions of genes, predicting the structure of proteins, searching for text duplicates, spellchecking, and modeling the visual similarity of words in humans [69, 31].

The developed methods for generating codevectors of complex structured data that include relations (Sec. 4.3) were used in reasoning by analogy to effectively search for analogs by similarity while simultaneously taking into account structure and semantics. The application of the proposed approach made it possible, on the ThinkNet knowledge base, to increase the search precision and recall up to 20% and 4 times, correspondingly [33]. In addition, methods for analogical mapping and inference were developed and tested.

In the problem of predicting the existence of chemical compounds on the INTAS00-397 dataset, the obtained results [94] exceeded the best known ones.

Due to the new methods of neural network regularization, systems with increased accuracy have been developed for gamma spectrometry at fixed and non-fixed measurement geometries [95], suppression of active interference [96], estimating the direction of signal arrival in antenna systems [97].

CONCLUSIONS AND PERSPECTIVES

The key issue in the problems of Artificial Intelligence is the adequate representation of data. The approach under development is based on the idea of distributed representation of information in the brain and allows representing various types of data, from numeric values to graphs, as vectors of large but fixed dimensionality. The similarity of the initial data is manifested in the similarity of the resulting vectors. This makes it possible to apply similarity search in solving a

number of problems based on case-based reasoning, presumably similar to how the human brain does, and also allows using an extensive arsenal of existing vector machine learning methods for processing and analysis.

We also develop theoretically substantiated feature extraction methods based on sparse multilayer neural network models and new approaches to regularization. The methods are applicable to data that allow both 1D and 2D representation, such as sequences (audio signals), 2D images, video sequence frames, and so on.

Recently, well-known figures in the field of Deep Neural Networks, such as G. Hinton [98], J. Bengio [99], Y. Schmidhuber [100] proposed that new ideas are required to overcome the shortcomings of Deep Neural Networks. The sense of the new direction of research is to provide the ability to form and operate structures consisting of internal representations of objects, without learning such representations from scratch. The methods being developed by us are also aimed to achieve such properties of "compositionality".

The main area of research reviewed in this paper is the problem of representing heterogeneous data in a unified format for the Artificial Intelligence systems based on modeling the neural network organization of the brain and the mechanisms of thinking hypothesized by Amosov. The most important advantages of the developed approach are the possibility of natural integration and efficient processing of various types of data and knowledge, a high degree of parallel computing, reliability and resistance to noise, the possibility of hardware implementation with high performance and energy efficiency, data processing based on associative search by similarity, similar to how human memory works. This allows one to unify methods, algorithms, software, and hardware for Artificial Intelligence systems, to increase their scalability in terms of speed and memory with an increase in data volume and complexity.

Currently, the topical direction of our developments is the creation of vector distributed representations of objects that would allow their modification inside the Artificial Intelligence system so that the result coincides with the representation of the external objects after some (corresponding) transformations. This applies to techniques for equivariantly representing sequences, as well as spatial objects such as 2D images and higher dimensional representations, including making such representations equivariant to translation, rotation, and scale. Such representations may be considered as analogous to imagery in human thinking, and operating with them may be seen as a form of creative thinking.

Another promising direction is the consideration of context. Many brain experiments show that the context has a great influence on the memorization of objects, events, scenes, etc. In APNNs, context is taken into account by the binding of codevectors that is performed by Context-Dependent Thinning, as well as by permutation. It is interesting to test the hypothesis that binding an object's codevector and a particular context's codevector yields strongly context-dependent representations of objects that do not allow the object to be recognized in another context. However, when an object is stored in many different contexts, a context-independent representation of the object is formed.

We believe that the reviewed studies create the basis for overcoming some of the shortcomings of modern approaches to specialized Artificial Intelligence based on Deep Neural Networks and will contribute to the development of Artificial General Intelligence.

REFERENCES

1. Amosov N.M. Modelling of thinking and the mind. New York: Spartan Books, 1967, 192 p.
2. Amosov N.M., Kasatkin A.M., Kasatkina L.M., Kussul E.M., Talaev S.A. Intelligent behaviour systems based on semantic networks. *Kybernetes*. 1973, Vol. 2, N 4, pp. 211–216.
3. Amosov N.M., Kussul E.M., Fomenko V.D. Transport robot with a neural network control system. *Advance papers of 4th Intern. Joint Conf. on Artif. Intelligence*. 1975, Vol 9, pp. 1–10.
4. Amosov N.M. Algorithms of the Mind. Kiev: Naukova Dumka, 1979, 223 p. (in Russian)
5. Amosov N.M., Baidyk T.N., Goltsev A.D., Kasatkin A.M., Kasatkina L.M., Rachkovskij D.A. Neurocomputers and intelligent robots. Kiev: Naukova Dumka. 1991, 269 p. (in Russian)
6. Kussul E.M. Associative neuron-like structures. Kiev: Naukova Dumka. 1992, 144 p. (in Russian)
7. Kleyko D., Rachkovskij D.A., Osipov E., Rahimi A. A survey on Hyperdimensional Computing aka Vector Symbolic Architectures, Part I: Models and data transformations. *ACM Computing Surveys*. 2022. https://doi.org/10.1145/3538531
8. Kleyko D., Rachkovskij D.A., Osipov E., Rahimi A. A survey on Hyperdimensional Computing aka Vector Symbolic Architectures, Part II: Applications, Cognitive Models, and Challenges. Accepted, *ACM Computing Surveys*. 2022. Available online: *arXiv:2112.15424*.
9. Thorpe S. Localized versus distributed representations The Handbook of Brain Theory and Neural Networks. Edited by M.A. Arbib.Cambridge, MA: The MIT Press. 2003, pp. 643–646.
10. Rachkovskij D.A., Kussul E.M. Binding and normalization of binary sparse distributed representations by context-dependent thinning. *Neural Computation*. 2001, Vol. 13, N 2, pp. 411–452.
11. Plate T. Holographic Reduced Representation: Distributed Representation for Cognitive Structures. Stanford: CSLI Publications, 2003, 300 p.
12. Kanerva P. Hyperdimensional computing: an introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation*. 2009, Vol. 1, N2, pp. 139–159.
13. Gayler R.W. Multiplicative binding, representation operators, and analogy. Advances in Analogy Research: Integration of Theory and Data from the Cognitive, Computational, and Neural Sciences. Sofia, Bulgaria: New Bulgarian University, 1998, p. 405.
14. Kussul E.M., Rachkovskij D.A., Baidyk T.N. Associative-Projective Neural Networks: Architecture, Implementation, Applications. *Proc. Neuro-Nimes'91*. 1991, pp. 463–476.
15. Kussul E.M., Rachkovskij D.A., Baidyk T.N. On image texture recognition by associative-projective neurocomputer. *Proc. ANNIE'91 Conference "Intelligent engineering systems through artificial neural networks"*. St. Louis, MO: ASME Press, 1991, pp. 453–458.
16. Kussul E.M., Rachkovskij D.A. Multilevel assembly neural architecture and processing of sequences. In Neurocomputers and Attention: Connectionism and Neurocomputers, vol. 2. Manchester and New York: Manchester University Press, 1991, pp. 577–590.
17. Gritsenko V.I., Rachkovskij D.A., Goltsev A.D., Lukovych V.V., Misuno I.S., Revunova E.G., Slipchenko S.V., Sokolov A.M., Talayev S.A. Neural distributed representation for intelligent information technologies and modeling of thinking. *Cybernetics and Computer Engineering*. 2013, Vol. 173, pp. 7–24. (in Russian).
18. Rachkovskij D.A., Kussul E.M., Baidyk T.N. Building a world model with structure-sensitive sparse binary distributed representations. *Biologically Inspired Cognitive Architectures*. 2013, Vol. 3, pp. 64–86.
19. Gritsenko V.I., Rachkovskij D.A., Frolov A.A., Gayler R., Kleyko D., Osipov E. Neural distributed autoassociative memories: A survey. *Cybernetics and Computer Engineering*. 2017, N 2 (188), pp. 5–35
20. Frolov A.A., Rachkovskij D.A., Husek D. On information characteristics of Willshaw-like auto-associative memory. *Neural Network World*. 2002. Vol. 12, No 2, pp. 141–158.
21. Frolov A.A., Husek D., Rachkovskij D.A. Time of searching for similar binary vectors in associative memory. *Cybernetics and Systems Analysis*. 2006, Vol. 42, N 5, pp. 615–623.
22. Fusi S. Memory capacity of neural network models. *arXiv:2108.07839*. 21 Dec 2021.
23. Steinberg J., Sompolinsky H. Associative memory of structured knowledge. 2022. https://doi.org/10.1101/2022.02.22.481380

24. Liang J.C., Erez J., Zhang F., Cusack R., Barense M.D. Experience transforms conjunctive object representations: Neural evidence for unitization after visual expertise. *Cerebral Cortex*. 2020. Vol. 30, N 5, pp. 2721–2739.
25. Li A.Y., Fukuda K., Barense M.D. Independent features form integrated objects: Using a novel shape-color "conjunction task" to reconstruct memory resolution for multiple object features simultaneously. *Cognition*. 2022, Vol. 223, Article 105024.
26. Andermane N., Joensen B.H., Horner A.J. Forgetting across a hierarchy of episodic representations. *Current Opinion in Neurobiology*. 2021, Vol. 67, pp. 50–57.
27. Michelmann S., Hasson U., Norman K.A. Event boundaries are steppingstones for memory retrieval. 2021. Preprint DOI 10.31234/osf.io/k8j94.
28. Schneegans S., McMaster J.M.V., Bays P.M. Role of time in binding features in visual working memory. *Psychological Review.* 2022. https://doi.org/10.1037/rev0000331
29. Geerligs L., van Gerven M., Campbell K.L., Güçlü U. A nested cortical hierarchy of neural states underlies event segmentation in the human brain. *Neuroscience.* 2021. https://doi.org/10.1101/2021.02.05.429165
30. Peer M., Brunec I.K., Newcombe N.S., Epstein R.A. Structuring knowledge with cognitive maps and cognitive graphs. *Trends in Cognitive Sciences*. 2021, Vol. 25, N 1, pp. 37–54.
31. Rachkovskij D.A. Shift-equivariant similarity-preserving hypervector representations of sequences. *arXiv:2112.15475*. 2021.
32. Rachkovskij D.A. Representation and processing of structures with binary sparse distributed codes. *IEEE Trans. Knowledge Data Engineering*. 2001, Vol. 13, N 2, pp. 261–276.
33. Rachkovskij D.A., Slipchenko S.V. Similarity-based retrieval with structure-sensitive sparse binary distributed representations. *Comput. Intelligence*. 2012,Vol. 28, N. 1, pp. 106–129.
34. Papadimitriou C.H., Vempala S.S., Mitropolsky D., Collins M.J., Maass W. Brain computation by assemblies of neurons. *Proceedings of the National Academy of Sciences*. 2020, Vol. 117, N 25, pp. 14464–14472.
35. Müller M.G., Papadimitriou C.H., Maass W., Legenstein R. A model for structured information representation in neural networks of the brain. *eNeuro*. 2020, Vol. 7, N 3, pp. 1–17.
36. Mitropolsky D., Collins M.J., Papadimitriou C.H. A biologically plausible parser. *Transactions of the Association for Computational Linguistics*. 2021, Vol. 9, pp. 1374–1388.
37. Papadimitriou C.H., Friederici A.D. Bridging the gap between neurons and cognition through assemblies of neurons. *Neural Computation*. 2022, Vol. 34, N 2, pp. 291–306.
38. Rachkovskij D.A., Misuno I.S., Slipchenko S.V. Randomized projective methods for the construction of binary sparse vector representations. *Cybernetics and Systems Analysis*. 2012, Vol 48, N 1, pp. 146–156.
39. Rachkovskij D.A. Vector data transformation using random binary matrices. *Cybernetics and Systems Analysis*. 2014, Vol. 50, N 6, pp. 960–968.
40. Rachkovskij D.A. Formation of similarity-reflecting binary vectors with random binary projections. *Cybernetics and Systems Analysis*. 2015, Vol. 51, N 2, pp. 313–323.
41. Rachkovskij D.A. Estimation of vectors similarity by their randomized binary projections. *Cybernetics and Systems Analysis*. 2015, Vol. 51, N 5, pp. 808–818.
42. Dasgupta S., Stevens C.F., Navlakha S. A neural algorithm for a fundamental computing problem. *Science*. 2017, Vol. 358, pp. 793–796.
43. Rachkovskij D.A., Gritsenko V.I. Distributed Representation of Vector Data Based on Random Projections. Kyiv: Interservice, 2018. ISBN: 978-617-696-837-5. (in Ukrainian)
44. Rachkovskij D.A. Codevectors: Sparse Binary Distributed Representations of Numerical Data. Kiev: Interservice, 2019. ISBN: 978-617-696-987-7. (in Russian)
45. Gritsenko V.I., Rachkovskij D.A. Methods for Vector Representation of Objects for Fast Similarity Estimation. Kyiv: Naukova Dumka, 2019. ISBN: 978-966-00-1741-2. (In Russian)
46. Rachkovskij D. A. Introduction to fast similarity search. Kyiv: Interservice, 2019, 294 p. ISBN: 978-617-696-904-4. (in Russian)
47. Ghazi B., Panigrahy R., Wang J. Recursive sketches for modular deep learning. *Proceedings of the 36th International Conference on Machine Learning*. *PMLR*. 2019, Vol. 97, pp. 2211–2220.
48. Panigrahy R., Wang X., Zaheer M. Sketch based memory for neural networks. *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics* (*AISTATS'21*). 2021, San Diego, California, USA. *PMLR*. 2021, Vol. 130, pp. 3169–3177.

49. Hiratani N., Sompolinsky H. Optimal quadratic binding for relational reasoning in vector symbolic neural architectures. *arXiv:2204.07186*. 2022, pp. 1–32.
50. Chaitanya R., Hopfield J., Grinberg L., Krotov D. Bio-inspired hashing for unsupervised similarity search. *Proceedings of the 37th International Conference on Machine Learning*. *PMLR*. 2020, Vol. 119, pp. 8295–8306.
51. Liang Y., Ryali C., Hoover B., Grinberg L., Navlakha S., Zaki M., Krotov D. Can a fruit fly learn word embeddings? *Proc. ICLR'21*. 2021.
52. Li W. Modeling winner-take-all competition in sparse binary projections. In: Machine Learning and Knowledge Discovery in Databases. Edited by: F. Hutter, K. Kersting, J. Lijffijt, I. Valera. *Lecture Notes in Computer Science*. Vol. 12457. Cham: Springer, 2021, pp. 456–472. https://doi.org/10.1007/978-3-030-67658-2_26
53. Li W.Y., Zhang S.Z. Binary random projections with controllable sparsity patterns. *Journal of the Operations Research Society of China*. 2022. https://doi.org/10.1007/s40305-021-00387-0
54. Rachkovskij D.A., Slipchenko S.V., Misuno I.S., Kussul E.M., Baidyk T.N. Sparse binary distributed encoding of numeric vectors. *Journal of Automation and Information Sciences*. 2005, Vol. 37, N 11, pp. 47–61.
55. Izawa S., Kitai K., Tanaka S., Tamura R., Tsuda K. Continuous black-box optimization with quantum annealing and random subspace coding. *Proc. Adiabatic Quantum Computing* (AQC'21), June 22-25, 2021.
56. Izawa S., Kitai K., Tanaka S., Tamura R., Tsuda K. Continuous black-box optimization with an Ising machine and random subspace coding. *Phys. Rev. Research*. 2022, Vol. 4, N 2. Article 023062.
57. Barclay I. et al. Trustable service discovery for highly dynamic decentralized workflows. *Future Generation Computer Systems*. 2022. DOI: 10.1016/j.future.2022.03.035
58. Wei Y., Xie P., Zhang L. Tikhonov regularization and randomized GSVD. *SIAM J. Matrix Analysis Appl.* 2016, Vol. 37, pp. 649–675.
59. Zhang L., Wei Y. Randomized core reduction for discrete ill-posed problem. *J. Comput. Appl. Math*. 2020, Vol. 375, Article 112797.
60. Wei W., Zhang H., Yang X., Chen X. Randomized generalized singular value decomposition. *Commun. Appl. Math. Comput.* 2021, Vol 3, pp. 137–156.
61. Zuo Q., Wei Y., Xiang H. Quantum-inspired algorithm for truncated total least squares solution. *arXiv:2205.00455*. 2022
62. Revunova E.G., Rachkovskij D.A. Using randomized algorithms for solving discrete ill-posed problems. *J. Information Theories and Applications*. 2009, Vol. 16, N 2, pp. 176–192.
63. Rachkovskij D.A., Revunova E.G. Randomized method for solving discrete ill-posed problems. *Cybernetics and Systems Analysis*. 2012, Vol. 48, N 4, pp. 621–635.
64. Revunova E.G. Analytical study of the error components for the solution of discrete ill-posed problems using random projections. *Cybernetics and Systems Analysis*. 2015, Vol. 51, N 6, pp. 978–991.
65. Revunova E.G. Model selection criteria for a linear model to solve discrete ill-posed problems on the basis of singular decomposition and random projection. *Cybernetics and Systems Analysis*. 2016, Vol. 52, N 4, pp. 647–664.
66. Revunova E.G. Increasing the accuracy of solving discrete ill-posed problems by the random projection method. *Cybernetics and Systems Analysis*. 2018, Vol. 54, N 5, pp. 842–852.
67. Revunova O.G., Tyshchuk O.V., Desiateryk O.O. On the generalization of the random projection method for problems of the recovery of object signal described by models of convolution type. *Control Systems and Computers*. 2021, N 5–6, pp. 25–34.
68. Sokolov A., Rachkovskij D. Approaches to sequence similarity representation. *Int. J. Inf. Theor. Appl*. 2006, Vol. 13, N 3, pp. 272–278.
69. Sokolov A. Vector representations for efficient comparison and search for similar strings. *Cybernetics and Systems Analysis*. 2007, Vol. 43, N 4, pp. 484–498.
70. Rachkovskij D.A. Development and investigation of multilevel assembly neural networks. PhD thesis. Kiev, Ukrainian SSR: V.M. Glushkov Institute of Cybernetics, 1990. (in Russian)
71. Kussul E.M., Baidyk T.N. On information encoding in Associative-Projective Neural Networks. Technical Report 93-3. V.M. Glushkov Institute of Cybernetics, 1993. (in Russian)

72. Kleyko D., Osipov E., De Silva D., Wiklund U., Vyatkin V., Alahakoon D. Distributed representation of n-gram statistics for boosting self-organizing maps with hyperdimensional computing. *Proc. PSI'19*. 2019, pp. 64–79.
73. Kussul E.M., Baidyk T.N., Wunsch D.C., Makeyev O., Martin A. Permutation coding technique for image recognition system," *IEEE Trans. Neural Networks*. 2006, Vol. 17, N 6, pp. 1566–1579.
74. Rachkovskij D.A. Application of stochastic assembly neural networks in the problem of interesting text selection. *Neural network systems for information processing*. 1996, pp. 52–64. (in Russian)
75. Rachkovskij D.A., Kleyko D. Recursive binding for similarity-preserving hypervector representations of sequences. *Proc. IJCNN'22*. 2022.
76. Kussul E., Baidyk T. Improved method of handwritten digit recognition tested on MNIST database. *Image and Vision Computing*. 2004, Vol. 22, pp. 971–981.
77. Makeyev O., Sazonov E., Baidyk T., Martin A. Limited receptive area neural classifier for texture recognition of mechanically treated metal surfaces. *Neurocomputing*. 2008, Vol. 71, N 7-9, pp. 1413–1421.
78. Rachkovskij D.A. Distance-based index structures for fast similarity search. *Cybernetics and Systems Analysis*. 2017, Vol. 53, N 4, pp. 636–658.
79. Rachkovskij D.A. Index structures for fast similarity search for binary vectors. *Cybernetics and Systems Analysis*. 2017, Vol. 53, N 5, pp. 799–820.
80. Kussul E.M., Baidyk T.N., Lukovich V.V., Rachkovskij D.A. Adaptive neural network classifier with multifloat input coding. *Proc. NeuroNimes'93*, Nimes, France, Oct. 25-29, 1993, pp. 209–216.
81. Lukovich V.V., Goltsev A.D., Rachkovskij D.A. Neural network classifiers for micromechanical equipment diagnostics and micromechanical product quality inspection. *Proc. EUFIT'97*, 1997, pp. 534–536.
82. Rachkovskij D.A., Kussul E.M. DataGen: a generator of datasets for evaluation of classification algorithms. *Pattern Recognition Letters*. 1998, Vol.19, N 7, pp. 537–544.
83. Goltsev A.D. Neural networks with assembly organization. Kyiv: Naukova Dumka. 2005, 200 p. (in Russian)
84. Rachkovskij D. Linear classifiers based on binary distributed representations. *J. Inf. Theories Appl.* 2007, Vol. 14, N 3, pp. 270–274.
85. Goltsev A., Rachkovskij D. A recurrent neural network for partitioning of hand drawn characters into strokes of different orientations. *International Journal of Neural Systems*. 2001, Vol. 11, pp. 463–475.
86. Goltsev A., Gritsenko V., Húsek D. Segmentation of visual images by sequential extracting homogeneous texture areas. *Journal of Signal and Information Processing*. 2020, Vol. 11, N 4, pp. 75–102.
87. Goltsev A. D., Gritsenko V.I. Algorithm of sequential finding the textural features characterizing homogeneous texture segments for the image segmentation task. *Cybernetics and Computer Engineering*. 2013, N 173, pp. 25–34.
88. Goltsev A., Gritsenko V., Kussul E., Baidyk T. Finding the texture features characterizing the most homogeneous texture segment in the image. *Proc. IWANN'15*. 2015, pp. 287–300.
89. Volkov O., Komar M., Volosheniuk D. Devising an image processing method for transport infrastructure monitoring systems. *Eastern-European Journal of Enterprise Technologies*. 2021, Vol. 4, N 2 (112), pp. 18–25. https://doi.org/10.15587/1729-4061.2021.239084
90. Gritsenko V., Volkov O., Komar M., Volosheniuk D. Integral adaptive autopilot for an unmanned aerial vehicle. *Aviation*. 2018, Vol. 22, N 4, pp. 129–135. http://dx.doi.org/10.3846/aviation.2018.6413
91. Misuno I. S., Rachkovskij D. A., Slipchenko S. V., Sokolov A. M. Searching for text information with the help of vector representations. *Probl. Programming*. 2005, N 4, pp. 50–59.
92. Goltsev A., Rachkovskij D. Combination of the assembly neural network with a perceptron for recognition of handwritten digits arranged in numeral strings. *Pattern Recognition*. 2005, Vol. 38, N 3, pp. 315–322.
93. Kasatkina L.M., Lukovich V.V., Pilipenko V.V. Recognition of the person's voice by the classifier LIRA. *Control Systems and Computers*. 2006, N. 3, pp. 67–73.

94. Slipchenko S. V. Distributed representations for the processing of hierarchically structured numerical and symbolic information. *System Technologies*. 2005, N 6, pp. 134–141. (In Russian)
95. Rachkovskij D.A., Revunova E.G. Intelligent gamma-ray data processing for environmental monitoring. In: Intelligent Data Processing in Global Monitoring for Environment and Security. Kiev-Sofia: ITHEA, 2011, pp. 136–157.
96. Revunova E.G. Rachkovskij D.A. Training a linear neural network with a stable LSP solution for jamming cancellation. *Intern. Journal Information Theories and Applications*. 2005, Vol.12, N 3, pp. 224–230.
97. Revunova E.G., Rachkovskij D.A. Random projection and truncated SVD for estimating direction of arrival in antenna array. *Cybernetics and Computer Engineering*. 2018, N 3(193), pp. 5–26.
98. Hinton G. How to represent part-whole hierarchies in a neural network. *arXiv:2102.12627*. 2021, pp. 1–44.
99. Goyal A., Bengio Y. Inductive biases for deep learning of higher-level cognition. *arXiv:2011.15091*. 2020, pp. 1–42.
100. Greff K., van Steenkiste S., Schmidhuber J. On the binding problem in artificial neural networks. *arXiv:2012.05208*. 2020, pp. 1–75.

Received 17.03.2022

Рачковський Д.А.[1,2], д-р техн. наук, проф.,
головн. наук. співроб. відд. нейромережевих технологій
оброблення інформації, запрошений професор,
відділення компютерних наук, електротехніки та космічної техніки,
https://orcid.org/0000-0002-3414-5334, e-mail: dar@infrm.kiev.ua
Гриценко В.І.[1], член-кореспондент НАН України,
радник дирекції, https://orcid.org/0000-0002-6250-3987, e-mail: vig@irtc.org.ua
Волков О.Є.[1], канд. техн. наук, старший дослідник,
директор, https://orcid.org/ 0000-0002-5418-6723, e-mail: alexvolk@ukr.net
Гольцев О.Д.[1], канд. техн. наук,
в.о. зав. відд. нейромережевих технологій оброблення інформації,
https://orcid.org/0000-0002-2961-0908, e-mail: root@adg.kiev.ua
Ревунова О.Г.[1], д-р. техн. наук,
старш. наук. співр. відд. нейромережевих технологій оброблення інформації,
https://orcid.org/0000-0002-3053-7090, e-mail: egrevunova@gmail.com
Клейко Д.[3], доктор філософії (комп'ютерні науки),
наук. спів., https://orcid.org/0000-0002-6032-6155, e-mail: denis.kleyko@ri.se
Лукович В.В.[1],
наук. спів. відд. нейромережевих технологій оброблення інформації,
https://orcid.org/0000-0002-3848-4712, e-mail: vvl97@ukr.net
Осипов Є.[2], доктор філософії (комп'ютерні науки), професор,
відділення комп'ютерних наук, електротехніки та космічної техніки,
https://orcid.org/0000-0003-0069-640X, e-mail: evgeny.osipov@ltu.se
[1] Міжнародний науково-навчальний центр інформаційних
технологій та систем НАН України та МОН України,
40, пр. Академіка Глушкова, м. Київ, 03680, Україна
[2] Технологічний університет Лулео, 971 87 Лулео, Швеція
[3] Дослідні інститути Швеції RISE, 164 40 Кіста, Швеція

НЕЙРОМЕРЕЖЕВІ РОЗПОДІЛЕНІ ПОДАННЯ ДАНИХ ДЛЯ ШТУЧНОГО ІНТЕЛЕКТУ ТА МОДЕЛЮВАННЯ МИСЛЕННЯ

Вступ. Сучасний прогрес у галузі спеціалізованого штучного інтелекту пов'язано з використанням глибоких нейронних мереж. Однак вони мають ряд недоліків: потреба у величезних наборах даних для навчання, складність навчальних процедур, надмірна спеціалізація навчального набору, нестійкість до змагальних атак, відсутність інтеграції зі знаннями про світ, проблеми роботи зі структурами, відомі як проблема зв'язування або композиції. Подолання

цих недоліків є необхідною умовою для просування від спеціалізованого штучного інтелекту до загального, що потребує розроблення альтернативних підходів.

Метою статті є огляд досліджень цього напряму, проведених у Міжнародному центрі протягом 25 років. Підхід до штучного інтелекту, що розробляється, випливає з ідей М.М. Амосова та його наукової школи. Також розглянуто зв’язки з напрямами гіпервекторних обчислень (HDC) та векторних символічних архітектур (VSA), а також з дослідженнями мозку.

Результати. Викладено концепцію розподіленого подання даних, включаючи HDC/VSA, які здатні подавати різні структури даних. Розглянуто розроблену парадигму асоціативно-проективних нейронних мереж: кодвекторне подання даних, операції суперпозиції та зв'язування, загальну архітектуру, перетворення даних різних типів у кодвектори, методи розв'язування задач та їхні застосування.

Висновок. Адекватне подання даних є одним з ключових питань штучного інтелекту. Основним напрямом дослідження, розглянутим у цій статті, є проблема подання різнорідних даних у системах штучного інтелекту в уніфікованому форматі на основі моделювання нейронної організації мозку та механізмів мислення. Розроблюваний підхід базується на гіпотезі розподіленого подання інформації в мозку та дає змогу подавати різні типи даних, від числових значень до графів, у вигляді векторів великої, але фіксованої розмірності.

Найважливішими перевагами розробленого підходу є можливість інтеграції та ефективного оброблення різних типів даних і знань, високий ступінь паралельності обчислень, надійність та стійкість до шумів, можливість апаратної реалізації з високою продуктивністю та енергоефективністю, оброблення даних на основі асоціативного пошуку за схожістю — подібно до того, як працює людська пам’ять. Це дає змогу уніфікувати методи, алгоритми та програмно-апаратні засоби для систем штучного інтелекту, підвищити їхню масштабованість за швидкістю та пам’яттю зі збільшенням обсягу та складності даних.

Дослідження створює основу для подолання недоліків сучасних підходів до створення спеціалізованого штучного інтелекту на основі глибоких нейронних мереж і відкриває шлях до створення загального штучного інтелекту.

Ключові слова: *розподілене подання даних, асоціативно-проективні нейронні мережі, кодвектори, гіпервекторні обчислення, векторно-символьні архітектури, штучний інтелект.*