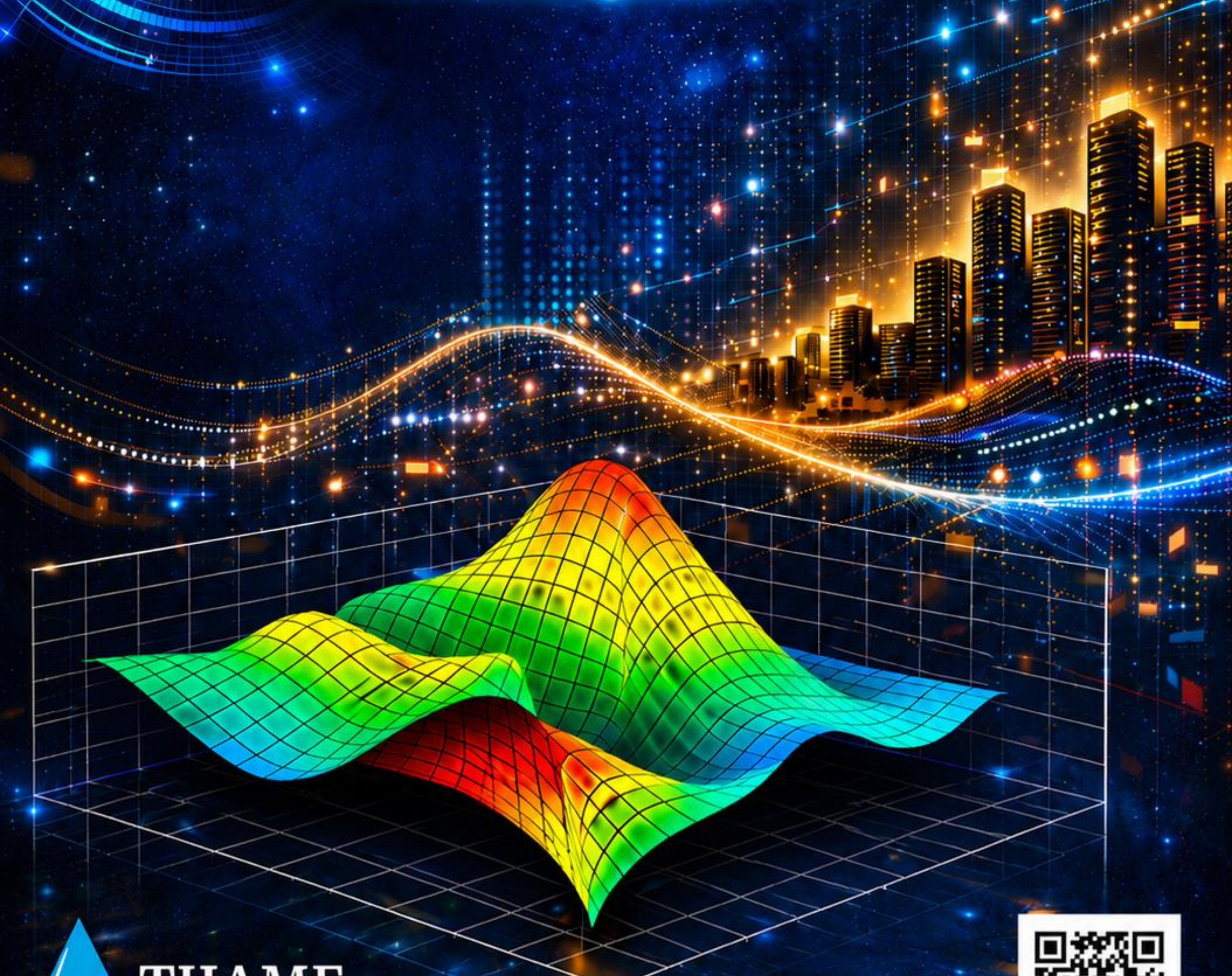
JMMME

JOURNAL OF MATHEMATICAL MODELS AND METHODS IN ENGINEERING



TIIAME
National Research University

JMMME

JOURNAL OF MATHEMATICAL MODELS AND METHODS IN ENGINEERING

JOURNAL INFORMATION

"Journal of Mathematical Models and Methods in Engineering" is an international, peer-reviewed, and open-access scientific journal established in the Republic of Uzbekistan by the National Research University "Tashkent Institute of Irrigation and Agricultural Mechanization Engineers" under Certificate No. 1648073 dated June 15, 2026.

The journal specializes in publishing scientific research on mathematical modeling, numerical methods, computational algorithms, optimization, and analytical approaches across various fields of engineering — including civil engineering, mechanical engineering, energy, hydraulic engineering, materials science, robotics, and environmental protection. It serves as a scientific platform for scholars, engineers, educators, and doctoral researchers to exchange theoretical and applied knowledge.

All submitted manuscripts undergo a double-blind peer review process and are published in strict compliance with high academic quality standards and publication ethics.

The journal has been published since June 15, 2026, once per quarter, that is, 4 times per year. Manuscripts are accepted in English and must be prepared in accordance with international IMRAD standards.

TIIAME
National Research University

FOUNDER	TIIAME National Research University
ADDRESS	39 Qori-Niyoziy Street, Mirzo Ulugbek District, Tashkent, Republic of Uzbekistan
PHONE	+998 (93) 397-12-39
WEBSITE	jmmme.uz
CERTIFICATE	No. 1648073, June 15, 2026
PUBLISHED	2026-06-15
LANGUAGE	English
FREQUENCY	Quarterly / 4 times per year
REVIEW	Double-blind peer review
STANDARD	International IMRAD standards

INDEXED AND DISCOVERABLE THROUGH

ISSN INTERNATIONAL STANDARD SERIAL NUMBER · Google Scholar · zenodo · doi · OpenAIRE · OPEN ACCESS

JMMME

JOURNAL OF MATHEMATICAL MODELS AND METHODS IN ENGINEERING

TIIAME
National Research University

Journal of Mathematical Models and Methods in Engineering

journal homepage: https://jmmme.uz/index.php/journal

Neural networks with configurable affinity functions for binary object classification

S.Yu. Leonov[1*], O.V. Lipchanska[2]

[1,2]*Department of Computer Engineering and Programming, "Kharkiv Polytechnic Institute" National Technical University, Kharkiv, 61000, Ukraine*

ARTICLE INFO

Keywords:
computer systems for recognition;
computing systems;
data transmission;
Hamming neural network;
affinity functions;
diagnostics of systems for determining operability;
classification of discrete objects;
binary vectors

ABSTRACT

The paper addresses computer systems for the recognition of discrete objects encoded by binary and bipolar feature vectors. It is shown that the well-known Hamming neural network, which uses the scalar product of bipolar vectors and the Hamming distance for recognition, does not allow the application of more refined similarity functions when object features are binary-encoded. Modifications of the network architecture are proposed, implementing computing systems for evaluating object similarity using the Jaccard, Russell-Rao, Sokal-Michener, Kulczynski, and Yule functions. Neuron and neural block structures are developed that ensure data transmission between the network's layers and perform diagnostics of systems for determining the operability of recognition algorithms through the parallel computation of match/mismatch variables for the features of compared binary vectors. A numerical example is given, confirming the correctness of the proposed computer tools for classification. It is shown that the developed approach extends the scope of application of neural-network-based computer systems for recognition and makes it possible to synthesize networks that determine several equally valid solutions simultaneously.

1. Introduction

The scalar product of vectors is often used in solving a variety of problems involving the determination of affinity (similarity or relationship) between different binary objects. In particular, the Hamming neural network [1 – 3], which uses the *DB* scalar product of two bipolar *n*-bit vectors $D=(d_1,\dots,d_n)$, $B=(b_1,\dots,b_n)$

$$DB=\sum_{k=1}^{n} d_k b_k = a_{eq}-r = 2a_{eq}-n \qquad (1)$$

where a_{eq} is the number of identical bipolar components in vectors *D* and *B*; *r* is the Hamming distance between vectors *D* and *B* (the number of distinct components in the vectors under consideration) $a_{eq}+r=n$, used in black-and-white image recognition based on the number of matching bipolar components, taking the Hamming distance into account

* Corresponding author. **S.Yu. Leonov**
E-mail addresses: serleomail@gmail.com

$$a_{eq} = n/2 + \frac{1}{2}\sum_{k=1}^{n} d_k b_k \tag{2}$$

The number of identical bipolar components in the vectors can be regarded as a measure of similarity between these vectors. For identical vectors, this measure of similarity reaches a maximum value equal to the number of binary bits in the vectors being compared. The similarity measure takes its minimum value, equal to zero, when the vectors being compared do not share a single binary bit with identical components.

The concept of the Hamming distance and measures of similarity (2) is based on the number of mismatched or matching bits in the binary vectors being compared. However, in [4 – 10], it is noted that a more refined classification can be introduced for the compared binary vectors using the special Table 1 when binary encoding the vectors $D = (d_1, \ldots, d_n)$, and $B = (b_1, \ldots, b_n)$, which contain the encoded features of certain compared objects.

Table 1

Relationship between the features of a pair of objects (binary vectors)

D \ B	1	0
1	a	e
0	f	b

The variable *a* in Table 1 is introduced to calculate the number of features (the number of components of binary vectors that are 1) that are present in both objects *D* and *B* simultaneously. It is calculated using the following formula:

$$a = \sum_{v=1}^{n} d_v b_v \tag{3}$$

The scalar product (3) of the binary vectors *D* and *B* is used to compute the similarity parameter (a measure of similarity between binary vectors) in the ART-1 discrete neural network [1, 2]. However, using only the nonzero components to assess the similarity of two binary vectors significantly limits the ability to make a more nuanced comparison of these vectors. To do this, knowledge of the variables *b*, *f*, and *e* (Table 1) is required.

The variable *b* in Table 1 is used to count the number of features (the number of zero components in the vectors) that the objects under consideration do not simultaneously possess

$$b = \sum_{v=1}^{n} (1 - d_v)(1 - b_v) \tag{4}$$

The variable *f* is used to count the number of attributes that object (vector) *B* has but object *D* does not. The variable *e*, on the other hand, is used to count the attributes that object *B* does not have but object *D* does:

$$f = \sum_{v=1}^{n} (1 - d_v) b_v \tag{5}$$

$$e = \sum_{v=1}^{n} d_v (1 - b_v) \tag{6}$$

An analysis of relations (3) – (6) shows that the measure of similarity between objects D and B increases as the affinity function a increases. The same cannot be said unequivocally about function (4), since the mere absence of identical features among objects can indicate either similarity between the objects or their belonging to different classes (when this is the only property they share). With regard to functions (5) and (6), it can be said that the measure of similarity between objects D and B is symmetric with respect to these functions.

Various combinations of functions (3) through (6) yield a wide range of affinity functions for binary vectors (objects whose presence or absence of features is encoded by binary components) [4 – 10]. Here are some of them:

Russell–Rao affinity function:

$$S_1 = \frac{a}{a+b+f+e} = \frac{a}{n} \tag{7}$$

Jaccard affinity function:

$$S_2 = \frac{a}{a+f+e} = \frac{a}{n-b} \tag{8}$$

Sokal–Michener affinity function:

$$S_3 = \frac{a+b}{n} \tag{9}$$

Kulczynski 1 and 2 affinity functions:

$$S_4 = \frac{a}{f+e} \tag{10}$$

$$S_5 = \frac{a}{2(a+e)} + \frac{a}{2(a+f)} \tag{11}$$

Dice affinity function:

$$S_6 = \frac{a}{2a+e+f} \tag{12}$$

Yule affinity function:

$$S_7 = \frac{ab-ef}{ab+ef} \tag{13}$$

Here is the research gap. Despite the effectiveness of the Hamming neural network for bipolar-encoded objects, it cannot be applied directly when the components of the compared objects are encoded with a binary (0/1) alphabet, nor can it evaluate affinity using functions such as Jaccard, Sokal–Michener, Kulczynski, and others. The large number of such affinity functions indicates the absence of a single universal function, and there are no generally accepted recommendations for choosing among them. This motivates the need for a systematic method of synthesizing neural networks that can compute arbitrary affinity functions for binary-encoded objects and, optionally, return several solutions.

The aim of this work is to develop modifications of the Hamming neural network that make it possible to solve recognition problems for discrete objects whose components are described using binary components, and where affinity functions of the form (7) – (13) serve as the proximity measure.

2. Methods

2.1 Illustrative comparison of the affinity functions

To characterize the behaviour of the affinity functions, we first perform a numerical experiment.

Example 1. Using the affinity functions (3) – (13) and the Hamming distance r, we compare three pairs of vectors D and B:

D = (1 1 0 1 1 1 0 0 0 1);
B = (1 0 1 0 1 1 1 0 0 1);

D = B = (1 1 0 1 1 1 0 0 0 1) and vectors D and B_1 = (0 0 1 0 0 0 1 1 1 0). The results of the calculations are presented in Table 2.

An analysis of Table 2 shows that the affinity functions $S_1 - S_7$ behave similarly:

– they take on minimum values for vectors with opposite binary components;

– they take on maximum values for pairs of identical binary vectors;

– if the vectors being compared are neither identical nor have opposite components, then the affinity functions $S_1 - S_7$ take on values between their maximum and minimum values.

The Hamming distance behaves, in a sense, in the opposite way—it takes on its minimum value for identical binary vectors and its maximum value for vectors that do not share a single identical binary bit.

Affinity functions are not limited to the relations (3) – (13) [4 – 10]. The large number of these functions indicates that there is no single universal function suitable for comparing any binary vectors with binary-coded components. Nor are there any universally accepted recommendations for the application of known functions.

Table 2

Values of the affinity functions for various pairs of vectors

Case	a	b	e	f	S_1	S_2	S_3	S_4	S_5	S_6	S_7	r
Values of similarity functions for vectors D and B.	4	2	2	2	0,4	0,5	0,6	1	2/3	1/3	0,5	4
Similarity function values for identical vectors D = B = (1101110001).	6	4	0	0	0,6	0,75	1	∞	1	0,5	1	0
Similarity function values for vectors D and B_1 with opposite components.	0	0	6	4	0	0	0	0	0	0	–1	10

The choice of an affinity function is typically made by analyzing the results of computer simulations in feature space, encoded by binary components.

2.2 Reference architecture: the Hamming neural network

Hemming's neural network [1 – 3]. It has n input neurons $S_1, \ldots, S_n$ (Fig. 1), which receive information about the bipolar components of the input images $D^q = (d_1^q, \ldots, d_n^q)$, $q = \overline{1, L}$. The output signals of the S-elements replicate their input signals: $U_{outS_i} = U_{inputS_i} = d_i^q$, $i = \overline{1, n}$. Each element of the S-layer is connected to each neuron of the Z-layer (Fig. 1). The weights (

$w_{1p}^1, w_{2p}^1, \ldots, w_{np}^1$) of the connections of neuron Z_p ($p=\overline{1, m}$) contain information about the p-th reference image: $B^p=(b_1^p, \ldots, b_n^p)$: $w_{1p}^1=b_1^p/2, \ldots, w_{np}^1=b_n^p/2$, where the superscript in the connection weights indicates the layer of Z-neurons.

The activation functions of the elements of the Z-layer are given by:

$$g_z(U_{input}) = \begin{cases} 0, \text{ if } U_{input} \le 0, \\ k_1 U_{input}, \text{ if } 0 \le U_{input} \le U_n, \\ U_n, \text{ if } U_{input} > U_n, \end{cases} \tag{14}$$

where U_{input} is an input signal of a neuron of Z- layer; k_1, U_n are constants.

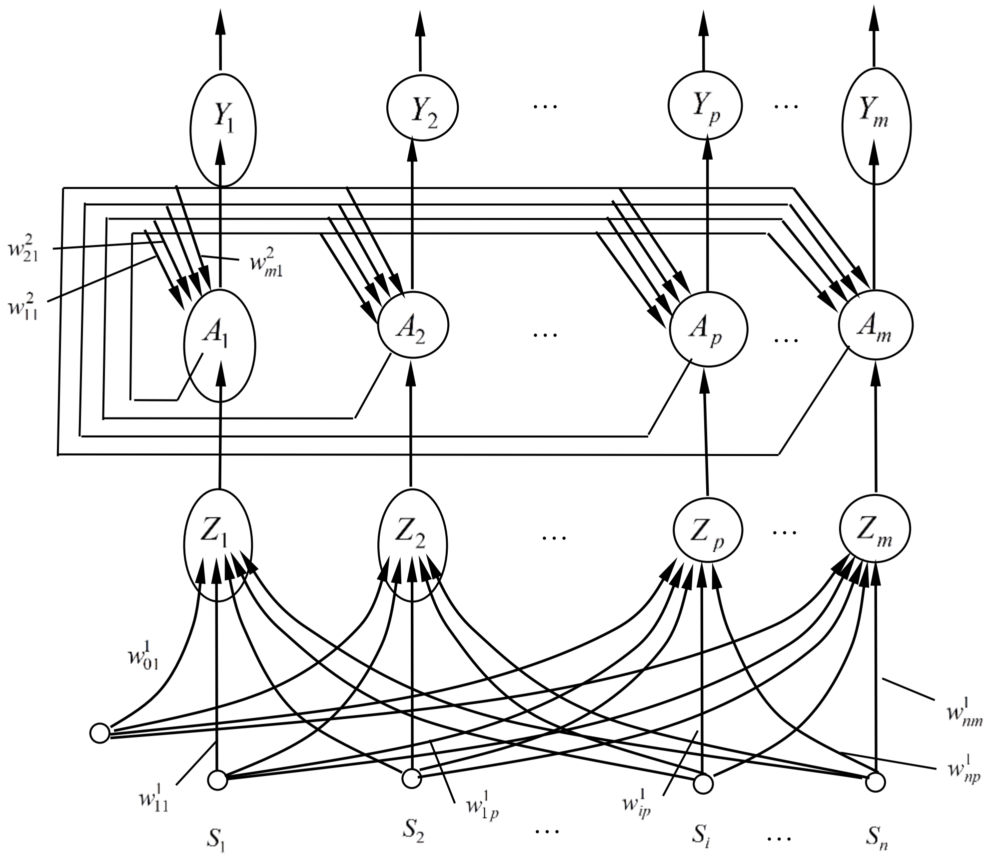


Fig. 1. The Hamming Neural Network

When an image $D^*=(d_1^*, \ldots, d_n^*)$ is fed into the neural network, each Z-element determines its input and output signals according to expressions of the form (2) and (14):

$$U_{inputZp} = n/2 + \frac{n}{2}\sum_{i=1}^{n} w_{ip} d_i^*,$$

$$U_{outZp} = g_z(U_{inputZp}) = a_p, \ p=\overline{1, m}.$$

The output signals from the neurons in the Z-layer are fed to the corresponding inputs of the elements in the A-layer (Fig. 1), which have activation functions described by the equation

$$U_{outAp} = g(U_{inputp}) = \begin{cases} U_{inputp}, \text{ if } U_{inputp} > 0, \\ 0, \text{ if } U_{inputp} \le 0, \ \ p = \overline{1,\ m}. \end{cases}$$

The weights of the connections between neurons in layer A are determined by the ratio

$$w_{ij}^2 = \begin{cases} 1, \ \text{ if } \ i=j, \\ -\varepsilon, \ \text{ if } \ i \ne j, \ \ i,j = \overline{1,\ m}, \end{cases}$$

where ε is a a constant that satisfies the expression $0 < \varepsilon \le 1/m$.

The layer of A-neurons operates cyclically, and the output signals of its elements are described by the relationship

$$U_{outAj}(t+1) = g(U_{outAj}(t) - \varepsilon \sum_{p=1,\ p \ne j}^{m} U_{outAp}(t)),\ \ j = \overline{1, m}$$

under initial conditions $U_{inputAj}(0) = a_j,\ \ j = \overline{1, m}$.

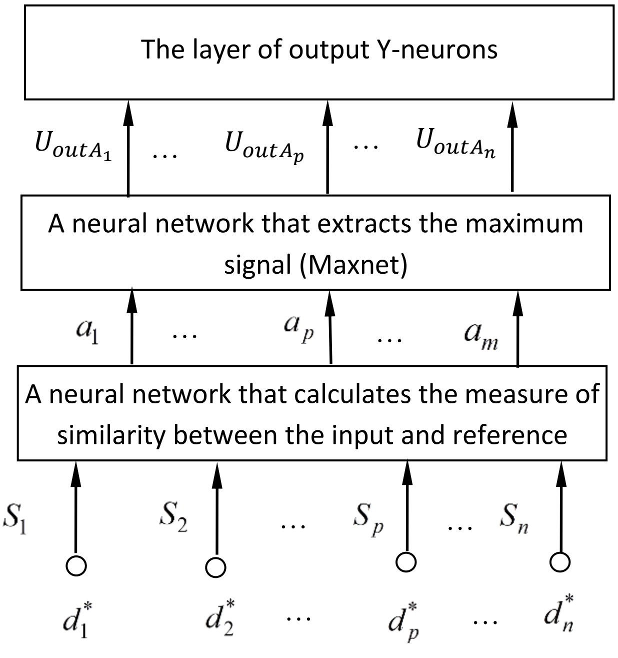
If there is a single largest input signal among the input signals a_p ($p = \overline{1, m}$), then as a result of the iterative process, only one neuron A_j will have an output signal greater than zero. Since the output signals of the elements in layer A are fed to the inputs of the neurons in layer Y, which have activation functions of the form

$$g_Y(U_{input}) = \begin{cases} 1, \text{ if } U_{input} > 0, \\ 0, \text{ if } U_{input} \le 0, \end{cases}$$

then the output of the Hemming network will consist of only one neuron Y_j with a unit output signal. This signal indicates that, in terms of image similarity (2), the input image $D^* = (d_1^*, \ldots, d_n^*)$ is closest to the reference image $B^j = (b_1^j, \ldots, b_n^j)$.

2.3 Modular representation and the proposed synthesis method

Hemming's neural network can also be represented in a more general form (Fig. 2).



(input image $D^* = (d_1^*, d_2^*, \ldots, d_{p,}^* \ \ldots, d_n^*)$

Fig. 2. Block diagram of the Hamming neural network

This representation of the Hamming neural network shows that the two main blocks of the network can be modified independently of one another. In particular, instead of a Maxnet, a neural network that identifies several identical maximum signals can be used. This makes it possible to find reference images that are all at the same minimum distance from the input [3, 11]. It also follows from the network architecture (Fig. 2) that the block calculating the measure of proximity between the input and reference images can compute various measures of proximity. The relative independence of the main blocks in the Hamming neural network makes it possible to propose neural networks both for working with binary images (vectors) and for using various affinity functions of the form (7) – (13) between the input and reference binary images.

In the Hamming neural network (Fig. 1), the scalar product of two binary vectors $D = (d_1, \dots, d_n)$, $B = (b_1, \dots, b_n)$ is computed as follows:

$$DB = \sum_{i=1}^{n} d_i b_i = (a+b) - (f+e) \tag{15}$$

where *a*, *b*, *f*, and *e* are variables defined by equations (3) – (6), with vector *D* being the input vector and vector *B* being the vector of neuron connection weights used to compute the scalar product. Since $(a+b)+(f+e)=n$, we can derive from (15) that

$$\sum_{i=1}^{n} d_i b_i = 2(a+b) - n$$

or

$$(a+b) = \frac{n}{2} + \frac{1}{2}\sum_{i=1}^{n} d_i b_i \; .$$

In a Hamming neural network, the value of the expression (*a* + *b*) is computed using a single neuron (Fig. 3). In the case of binary encoding of object features or binary vectors, it is not possible to use a Hamming network to compute the scalar product of two vectors. Most of the affinity functions (7) – (13) cannot be computed using a single neuron either. To compute these functions, one must first determine functions (3) – (6) using separate neurons and then use them to compute the affinity functions (7) – (13).

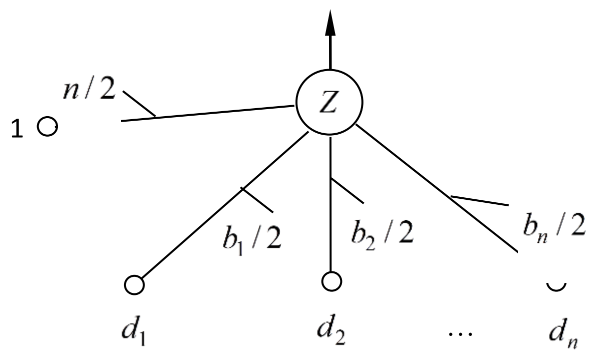


Fig. 3. A neuron for computing the scalar product of two bipolar vectors in a Hamming neural network;

$n/2$, $b_1/2$, $b_2/2$, …, $b_n/2$ are the neuron's connection weights; d_1, d_2, …, d_n are the components of the input vector.

Figure 4 shows a neuron for computing the variable *a*. The neuron's G_1 input signals are determined by the components of the binary vector *D*, and the connection weights are determined by the components of the binary vector $B = (b_1, \dots, b_n)$. Figure 5 shows a block for computing the variable *b* (Equation (4)). In this block, using the summing elements Σ_1, … , Σ_n the input signals $(1-d_1)$, $(1-d_2)$, …, $(1-d_n)$ of neuron G_2 are first computed; these are then multiplied by the corresponding weight coefficients $(1-b_1)$, $(1-b_2)$, …, $(1-b_n)$ and subsequently summed by the element G_2 . This leads to the computation of Equation (4). If, in Fig. 5, the input signals d_1 , d_2 , …, d_n are applied to the inputs of neuron G_2 instead of the signals $(1-d_1)$, $(1-d_2)$, …, $(1-d_n)$ computed by the summation elements, then neuron G_2 will compute the variable *e* (Equation (6)).

To compute the variable f (Equation (5)) in Fig. 5, it suffices to replace the weight coefficients $(1-b_1)$, $(1-b_2)$, …, $(1-b_n)$ with the corresponding weight coefficients $(b_1, \ldots, b_n)$.

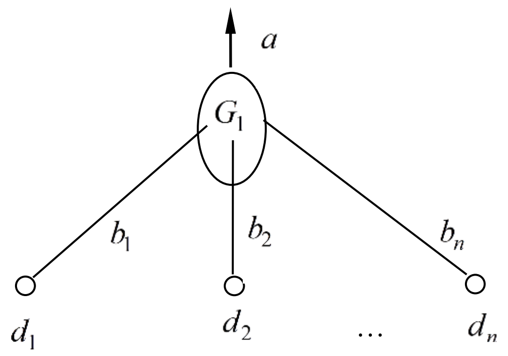


Fig. 4. A neuron for computing the variable a; b_1, b_2, …, b_n are the neuron's connection weights; d_1, d_2, …, d_n are the components of the input vector.

Given neurons or neural components for computing the variables *a, b, f,* and *e,* it is straightforward to derive neural networks for computing any of the affinity functions (7) – (13). Let us consider the most complex of these — the Kulchinsky 2 affine function. The neural network for calculating the proximity measure between the input and reference images is shown in Fig. 6.

Similarly, a neural network can be synthesized to compute any other affine function from the given list. In particular, Fig. 7 shows a neural network for computing the Yule affine function (equation (13)).

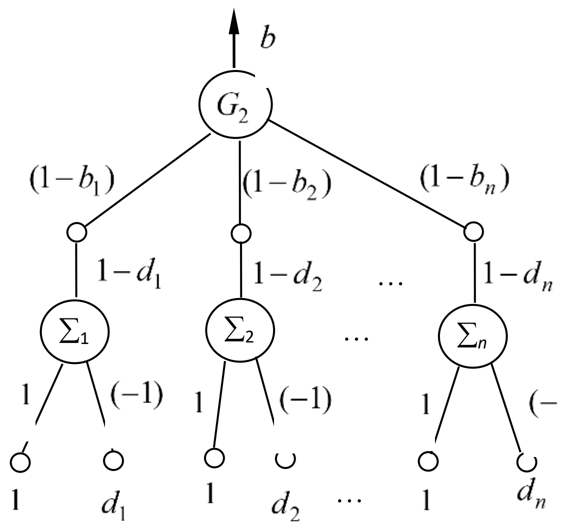


Fig. 5. A neuron for computing the variable b. The following notation is used in the figure: $(1-b_1)$, $(1-b_2)$, …, $(1-b_n)$ are the connection weights of the neuron ; G_2; Σ_1, …, Σ_n are summing elements for computing the G_2 neuron's input signals.

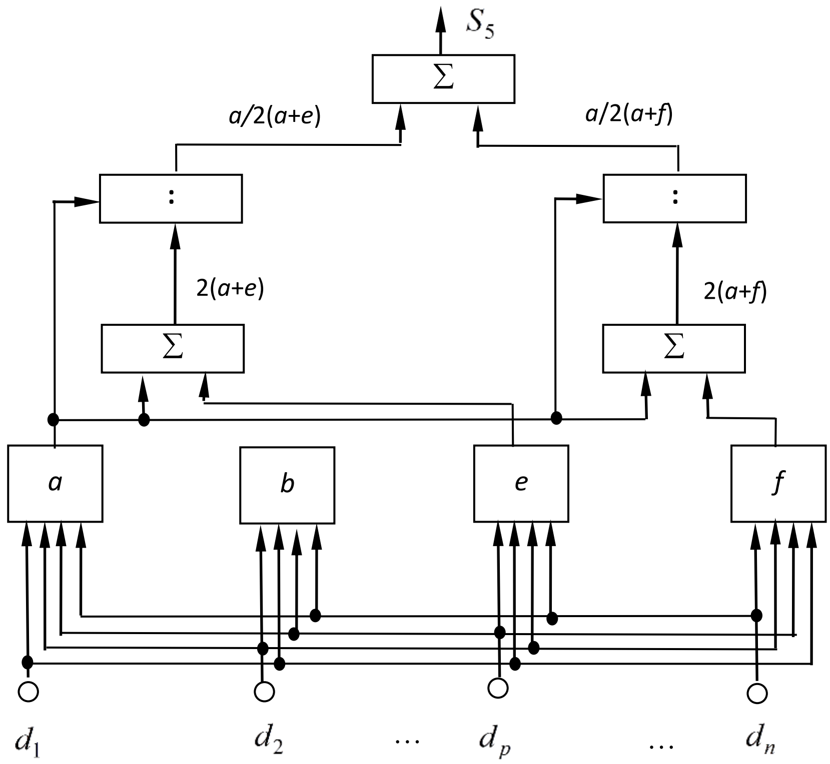


Fig. 6. Neural network computing the Kulchinsky function 2.

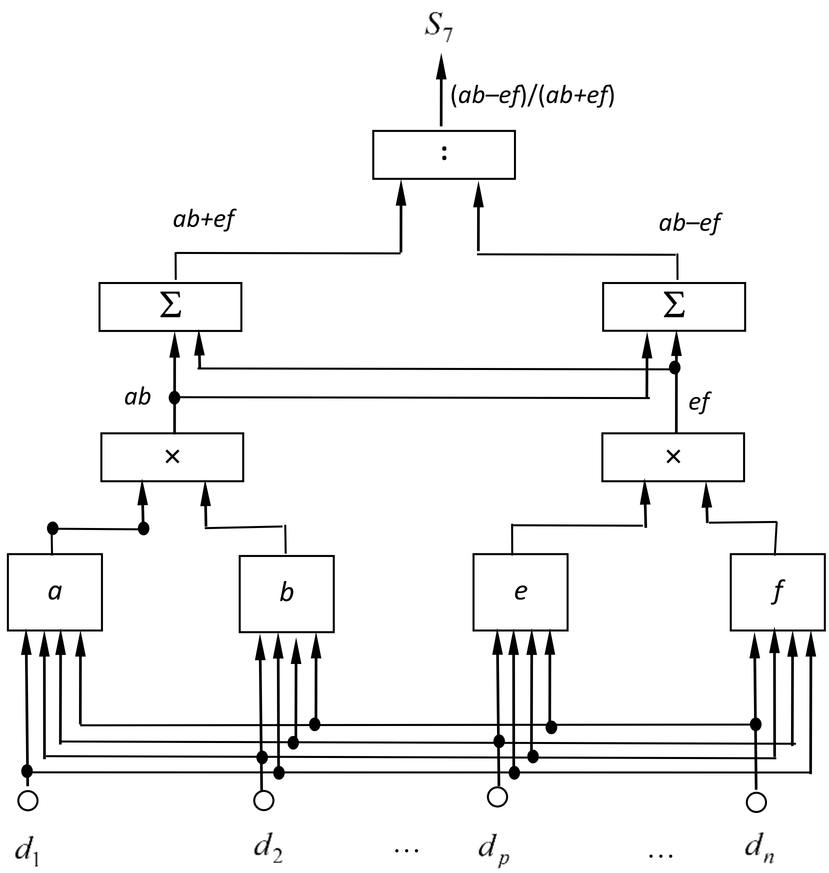


Fig. 7. Neural network that computes the Eul's affinity function.

3. Results

The numerical experiment (Example 1, Table 2) yields the following primary findings on the behavior of the affinity functions S1 – S7:

- they attain their minimum values for vectors with opposite binary components.
- they attain their maximum values for a pair of identical binary vectors.
- for vectors that are neither identical nor opposite, S1 – S7 take intermediate values between their maxima and minima.

In contrast, the Hamming distance behaves oppositely: minimal for identical vectors and maximal for vectors sharing no identical digit.

The affinity functions are not exhausted by (3) – (13) [4 – 10]; their large number indicates the absence of a single universal function, and there are no generally accepted recommendations for their application. In practice, the affinity function is chosen through computer simulation in the binary-encoded feature space.

The methodological results form a complete constructive synthesis procedure:

- elementary blocks compute a (Fig. 4), b, e, f (Fig. 5) directly from binary inputs;
- composite networks realize complex functions: Kulczynski 2 (Fig. 6, S5) and Yule (Fig. 7, S7);
- the modular architecture (Fig. 2) permits independent substitution of both the similarity-measure block (any of S1 – S7) and the Maxnet block (several solutions).

Thus, networks were obtained that operate directly on binary vectors and evaluate a broad family of affinity measures, whereas the original Hamming network is restricted to bipolar encoding and the similarity of Eq. (2).

4. Discussion

The results confirm and extend prior work [4 – 7] on the diversity of affinity measures for binary objects. Whereas the standard Hamming network [1 – 3] relies solely on the bipolar scalar product and effectively on a single criterion (Eq. 2), our modular reinterpretation (Fig. 2) reveals that the proximity-measure block and the decision block are functionally decoupled — the key insight enabling direct realization of the whole family S1 – S7.

Comparison with the Hamming distance (Table 2) clarifies why the bipolar network cannot be reused for binary encoding: the Hamming distance and the affinity functions respond in opposite directions, and functions such as Jaccard (8), Sokal–Michener (9), Kulczynski (10) – (11), Dice (12) and Yule (13) require the joint quantities a, b, e, f, which the single scalar-product neuron (Fig. 3) cannot produce. Our method resolves this by materializing a, b, e, f as separate blocks (Figs. 4 – 5) and composing them with adders, multipliers and dividers (Figs. 6 – 7), generalizing the ART-1 use of the binary scalar product [1, 2] far beyond the single feature a.

Replacing Maxnet by a network isolating several equal maximal signals [3, 11] allows reporting multiple equidistant reference images — particularly relevant since different measures can produce ties.

A recognized limitation, consistent with [4 – 10], is that no single affinity function is universally optimal; the choice remains task-dependent and is settled through simulation. The proposed method does not remove this need but provides a uniform, replicable way to instantiate whichever function is selected, lowering the cost of experimentation.

5. Conclusions

The Hamming neural network is an effective tool for recognizing and classifying discrete objects encoded with a bipolar alphabet, where the proximity measure is the difference between the number of identical bipolar components and the Hamming distance. However, it cannot be applied under binary encoding, nor can it evaluate affinity using the Jaccard, Sokal–Michener, Kulczynski and similar functions.

To overcome this, a method for synthesizing neural networks that assess binary-vector proximity using these affinity measures has been developed. Building on elementary blocks that compute a, b, e, f, the method constructs networks for arbitrarily complex functions, as demonstrated for

Kulczynski 2 (Fig. 6) and Yule (Fig. 7). This extends the applicability of neural networks to tasks relying on finer proximity features of binary-encoded objects.

Moreover, because the modular architecture allows Maxnet to be replaced by a network that can determine several solutions (when they exist), it is possible to synthesize networks returning a predefined number of solutions. The main significance of the study is twofold: it broadens the class of similarity criteria realizable by Hamming-type networks, and it provides a uniform, replicable synthesis procedure for both the proximity-measure and the decision stages.

References

[1] Ke-Lin Du , M.N.S. Swamy. Neural Networks and Statistical Learning, 2nd edition. Springer (2019).

[2] M.T. Hagan, H.B. Demuth, M.H. Beale, O. De Jesús. Neural Network Design, 2nd edition (2014).

[3] Dmitrienko, V.D., Zakovorotny, A.Yu., and Leonov, S.Yu. Hamming neural network for solving problems with several solutions, Bulletin of NTU "KPI, , Kharkiv, NTU "KPI", No. 50 (1271) (2017). 119-129.

[4] Babichev, S.A. Theoretical and practical principles of information technology for processing gene expression profiles for gene network reconstruction. Dissertation for the degree of Doctor of Technical Sciences (2018).

[5] Dmitrienko, V.D., Khavina, I.P., Zakovorotny, A.Yu., Lipchansky, M.V., and Mezentsev, N.V. Methods and algorithms of artificial intelligence systems, Kiev, Kafedra, (2014).

[6] Richard O. Duda, Peter E. Hart, David G. Stork — Pattern Classification, 2nd edition. Wiley (2001).

[7] Christopher M. Bishop. Pattern Recognition and Machine Learning. Springer (2006).

[8] Andrew R. Webb, Keith D. Copsey. Statistical Pattern Recognition, 3rd edition. Wiley (2011).

[9] Lynne Billard, Edwin Diday. Symbolic Data Analysis: Conceptual Statistics and Data Mining. Wiley (2006).

[10] Charu C. Aggarwal, Chandan K. Reddy (eds.) Data Clustering: Algorithms and Applications. CRC Press (2013).

[11] N. Ikeda, P. Watta, M. Artiklar, M. Hassoun. A two-level Hamming network for high performance associative memory. Neural Networks (2001).

JMMME

JOURNAL OF MATHEMATICAL MODELS AND METHODS IN ENGINEERING

$$\frac{dx}{dt} = Ax + b(t)$$

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u + f$$

$$\sum_{i=1}^{n} a_i x_i + b$$

$$f(x) = \int_a^b f(x)\,dx$$

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

$$\sum_{i=1}^{n} w_i x_i$$

CONTENTS

TIIAME National Research University