Received November 20, 2020, accepted November 30, 2020, date of publication December 7, 2020,
date of current version December 18, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3043035

Hamiltonian and Q-Inspired Neural Network-Based Machine Learning

WIESLAW CITKO AND WIESLAW SIENKO, (Life Member, IEEE)
Faculty of Electrical Engineering, Gdynia Maritime University, 81-225 Gdynia, Poland
Corresponding author: Wieslaw Citko (w.citko@we.umg.edu.pl)

This work was supported by the Gdynia Maritime University under Grant WE/2019/PZ/07.

ABSTRACT The goal of this study is to present a universal large-scale machine learning model based on spectral processing. By machine learning, we mean input-output mapping approximation generated by training sets. We treat tasks such as pattern recognition and classification as special problems in mapping approximation. The structures of the approximators are implemented using Hamiltonian neural network-based biorthogonal and orthogonal transformations. From a mathematical point of view, these structures can be seen as an implementation of non-expansive mappings. An interesting property of approximators is the reconstruction and recognition of incomplete or distorted patterns. The reconstruction property gives rise to a proposition of a superposition processor and reversible computations. Finally, the models of machine learning described here are adequate for processing data with real and complex values by defining Q-inspired neural networks.

INDEX TERMS Associative memory, data reconstruction, deep learning, Hamiltonian neural networks, machine learning, Q-inspired.

I. INTRODUCTION

The problem of learning represents a key to understanding intelligence in both brains and machines [1], [2]. By machine learning, we mean here input-output mapping approximation, where nodes of approximation are given by the set of training pairs $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{N}, \boldsymbol{x}_i \in X \subset R^n, \boldsymbol{y}_i \in Y \subset R^m$. Hence, one aims to realize the mapping $F : X \to Y$, where the value of such a mapping (or multivariable function $f(\cdot)$ for $\boldsymbol{y}_i \in Y \subset R$) is known at the training points, i.e.,

$$\boldsymbol{y}_i = F(\boldsymbol{x}_i), \quad i = 1, \ldots, N \tag{1}$$

Classification and pattern recognition issues can be seen as an important problem in mapping approximation. It should be noted, however, that deep learning is currently driving a renaissance of interest in neural network research and applications (e.g., AI, big data, deep convolutional neural networks) [3], [4]. Such neural networks used for the realization of $F(\boldsymbol{x})$ (1) take the form of multilayer nonlinear kernel machines. From a mathematical point of view, they set up a structure of algebraic mappings. Currently, most of learning algorithms are based on optimization procedures (often, however, without any constraints). Different forms of stochastic gradient descent (SGD) dominate optimization algorithms used today [5]–[7]. Thus, deep learning, technology that is widely used in commercial applications, can be seen as a special topic in optimization theory. Moreover, due to the massive amount of training data, these optimization methods are adapting to the evolving features of processed information [8]. It is worth noting that as a potential direction for future deep learning research, geometric deep learning methods have been proclaimed [9]. Nevertheless we claim that artificial neural networks (ANN) should constitute both universal algorithmic and physical models used in computational intelligence. However, currently, optimal architecture and implementation technology have not yet been developed. The main direction of research seems to be focused on three subjects:

1) Research on classical (non-quantum) computational models with real-valued parameters (RVNN);
2) Research on classical computational models with complex parameters (CVNN);
3) Research on quantum neural networks (QNNs), which are an alternative to quantum computers (QCs).

Note that so-called quantum-inspired (Q-inspired) neural networks are a non-quantum version of CVNN [10]–[12]. In a set of known real-valued neural networks, Hopfield-type neural networks fulfill an essential role [13], [14]. They are both physical and algorithmic models of neural computations. In this study we consider an extended model of Hopfield-type neural networks defined as follows:

$$\dot{x} = (\eta \boldsymbol{W} - w_0 \boldsymbol{1} + \varepsilon \boldsymbol{W}_s)\, \boldsymbol{\theta}(\boldsymbol{x}) + \boldsymbol{I}_d \tag{2}$$

The associate editor coordinating the review of this manuscript and approving it for publication was Ikramullah Lali.

where: $\boldsymbol{W}$—skew-symmetric orthogonal matrix
$\boldsymbol{W}_s$−real symmetric matrix
$\mathbf{1}$−identity matrix
$\boldsymbol{\theta}(\boldsymbol{x})$ −vector of activation functions
$\boldsymbol{I}_d$−input vector
ε, w_0, η−parameters

A model defined by (2) gives rise to the following types of neural networks:

a) Hamiltonian neural networks (HNNs) for $\varepsilon, w_0 = 0, \eta = 1$;
b) Classical Hopfield neural networks for $w_0, \eta = 0$.
c) Q-inspired Hopfield-type neural networks:

$$(\eta\, \boldsymbol{W} - w_{\mathbf{0}}\mathbf{1} + \varepsilon \boldsymbol{W}_H)\, \boldsymbol{\theta}(\boldsymbol{x}) + \boldsymbol{I}_d = \mathbf{0} \tag{3}$$

where: $\boldsymbol{W}_H$—Hermitian matrix ($\boldsymbol{W}_H = \boldsymbol{W}_{\boldsymbol{H}}^{\dagger}$)

If $\boldsymbol{W}_H$ is a real symmetric matrix, then (3) is an equilibrium equation of neural networks (2). The main purpose of this study is to illustrate that a mapping (1) at points $\boldsymbol{x}_i$ can be implemented in the form of a composition of extended Hopfield-type neural network-based biorthogonal and orthogonal spectra transformations. An important feature of this model is its universality, enabling the realization of the basic functions of large-scale learning systems, such as pattern association, pattern recognition, classification, and inverse modeling. The pattern recognition feature is illustrated in this study by an example that reconstructs a distorted signal (e.g., using images). Moreover, Q-inspired neural networks, i.e., complex-valued neural networks as defined by (3), gain the computational efficiency of machine learning models. Thus, the input-output mapping approximations (1) can be augmented to complex vector spaces: $\boldsymbol{x}_i \in C^n, \boldsymbol{y}_i \in C^m$. Recently formulated models, defined as quantum machine learning (QML), are quantum algorithms [15]. Their execution requires universal QCs that are not yet available. To our knowledge, Q-inspired neural networks are currently not available as physical objects, either. Hence they should be seen as algorithmic solutions. To summarize, we proposed in this paper a machine learning model, that makes use of biorthogonal and orthogonal transformations based on spectral processing, as alternative solutions to deep learning based on optimization procedures.

II. HNN-BASED ORTHOGONAL TRANSFORMATIONS

A general description of a Hamiltonian system is given by the following state space equation [16], [17]:

$$\dot{x} = \boldsymbol{J}\nabla H(\boldsymbol{x}) = \boldsymbol{v}(\boldsymbol{x}) \tag{4}$$

where: $\boldsymbol{x}$—state vector, $\boldsymbol{x} \in R^{2n}$
$\boldsymbol{v}(\boldsymbol{x})$−nonlinear vector field
$\boldsymbol{J}$−skew-symmetric, orthogonal matrix e.g., Poisson matrix
$\nabla H(\boldsymbol{x})$ −gradient of energy

The Hamiltonian function $H(\boldsymbol{x}) = E_k + E_p$ is the total energy (i.e., the sum of kinetic, E_k, and potential, E_p, energy) absorbed into the system. Because Hamiltonian systems are lossless (dissipationless), their trajectories in the state space can be quite complex and oscillatory for $t \in (-\infty, \infty)$. Equation (4) gives rise to the model of HNNs [18], [19] as follows:

$$\dot{x} = \boldsymbol{W}\nabla H(\boldsymbol{x}) = \boldsymbol{W\theta}(\boldsymbol{x}) + \boldsymbol{d} \tag{5}$$

where: $\boldsymbol{W}$—skew-symmetric, orthonormal weight matrix ($\boldsymbol{W}^2 = -\mathbf{1}$), dim $\boldsymbol{W} = 2^n$
$\nabla H(\boldsymbol{x}) = \boldsymbol{\theta}(\boldsymbol{x})$ −vector of activation functions (output $\boldsymbol{y} = \boldsymbol{\theta}(\boldsymbol{x})$)
$\boldsymbol{d}$−input vector
and Hamiltonian function:

$$H(\boldsymbol{x}) = \sum_{i=1}^{2n} \int_{-\infty}^{t} \dot{x}_i \Theta(x_i)\, d\tau + \sum_{i=1}^{2n} p_i x_i, \quad p_i \in R \tag{6}$$

One assumes here that activation functions are passive, i.e.,

$$\mu_1 \leq \frac{\theta(x_i)}{x_i} \leq \mu_2 : \mu_1, \mu_2 \in (0, \infty), \quad i = 1, \ldots 2^n \tag{7}$$

One can easily see that the HNN comprises compatible connections of n elementary building elements—lossless neurons. The state space description of a lossless, autonomous neuron is as follows:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & \pm w_1 \\ \mp w_1 & 0 \end{bmatrix} \begin{bmatrix} \theta(x_1) \\ \theta(x_2) \end{bmatrix}$$

The HNN described by (5) cannot be realized exactly as a macroscopic-scale physical, lossless object. Nevertheless, by introducing negative-feedback loops, equation (5) can be reformulated as follows:

$$\dot{x} = (\boldsymbol{W} - w_{\mathbf{0}}\mathbf{1})\, \boldsymbol{\theta}(\boldsymbol{x}) + \boldsymbol{d} \tag{8}$$

where: $w_{\mathbf{0}} > 0$
$\mathbf{1}$−identity matrix

Due to the assumed negative-feedback loop in (8), the neural networks considered here are not oscillatory. The stable (i.e. $|\boldsymbol{x}| \prec \infty$) equilibrium point of network (8) sets up an orthogonal transformation:

$$\boldsymbol{\theta}(x) = \boldsymbol{y} = \frac{1}{1 + w_0^2} (\boldsymbol{W} + w_{\mathbf{0}}\mathbf{1})\, \boldsymbol{d} \tag{9}$$

where: $\boldsymbol{W}^2 = -\mathbf{1}$ and $\boldsymbol{y}$ is a Haar spectrum of $\boldsymbol{d}$
$\boldsymbol{y}$−output vector

Note 1

A Haar spectrum is the result of a Haar transformation, where the transformation matrix {-1, 0, 1} is orthogonal but not skew-symmetric. On the other hand, the main challenge in HNN-based orthogonal transformation is to create the weight matrices, $\boldsymbol{W}$, skew-symmetric and orthogonal. The most adequate mathematical framework for this task seems to be an algebraic theory of Hurwitz-Radon matrices [20]. Hence, we show how Hurwitz-Radon matrices can be used in the construction of orthogonal transformations (filters), by defining matrices $\boldsymbol{W}$ as the superposition of Hurwitz-Radon matrices. Moreover, only for the matrix $\boldsymbol{W}_8$ does one have available eight free design parameters, $w_0, w_1, \ldots, w_7$, to synthesize any eight-dimensional orthogonal filter and to solve the following inverse problem. Thus, an eight-dimensional orthogonal transformation, referred to as an *octonionic module*, can be synthesized by the formula:

$$\boldsymbol{y} = \frac{1}{a^2}\left(\boldsymbol{W}_8 + w_0\boldsymbol{1}\right)\boldsymbol{d} = \frac{1}{a}\boldsymbol{T}_8\boldsymbol{d} \tag{10}$$

where: $a = \sqrt{\sum_{i=0}^{7} w_i^2}$ —scaling parameter

$\boldsymbol{T}_8 = \frac{1}{a}\left(\boldsymbol{W}_8 + w_0\boldsymbol{1}\right)$ —transformation matrix of octonionic module

Weight matrix $\boldsymbol{W}_8$ of octionic module:

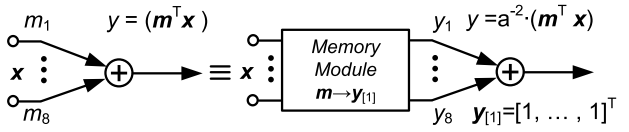
$$\boldsymbol{W}_8 = \begin{bmatrix} 0 & w_1 & w_2 & w_3 & w_4 & w_5 & w_6 & w_7 \\ -w_1 & 0 & w_3 & -w_2 & w_5 & -w_4 & -w_7 & w_6 \\ -w_2 & -w_3 & 0 & w_1 & w_6 & w_7 & -w_4 & -w_5 \\ -w_3 & w_2 & -w_1 & 0 & w_7 & -w_6 & w_5 & -w_4 \\ -w_4 & -w_5 & -w_6 & -w_7 & 0 & w_1 & w_2 & w_3 \\ -w_5 & w_4 & -w_7 & w_6 & -w_1 & 0 & -w_3 & w_2 \\ -w_6 & w_7 & w_4 & -w_5 & -w_2 & w_3 & 0 & -w_1 \\ -w_7 & -w_6 & w_5 & w_4 & -w_3 & -w_2 & w_1 & 0 \end{bmatrix} \tag{11}$$

and

$$\begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \end{bmatrix} = \frac{1}{\sum_{i=1}^{8} y_i^2} \begin{bmatrix} y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_7 & y_8 \\ -y_2 & y_1 & -y_4 & y_3 & -y_6 & y_5 & y_8 & -y_7 \\ -y_3 & y_4 & y_1 & -y_2 & -y_7 & -y_8 & y_5 & y_6 \\ -y_4 & -y_3 & y_2 & y_1 & -y_8 & y_7 & -y_6 & y_5 \\ -y_5 & y_6 & y_7 & y_8 & y_1 & -y_2 & -y_3 & -y_4 \\ -y_6 & -y_5 & y_8 & -y_7 & y_2 & y_1 & y_4 & -y_3 \\ -y_7 & -y_8 & -y_5 & y_6 & y_3 & -y_4 & y_1 & y_2 \\ -y_8 & y_7 & -y_6 & -y_5 & y_4 & y_3 & -y_2 & y_1 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \\ d_7 \\ d_8 \end{bmatrix} \tag{12}$$

It can be seen that (12) is a solution for the following inverse problem: for a given input vector $\boldsymbol{d} = [d_1, \ldots, d_8]^{\mathrm{T}}$ and a given output vector $\boldsymbol{y} = [y_1, \ldots, y_8]^{\mathrm{T}}$, find weight matrix $\boldsymbol{W}_8$ of an HNN-based orthogonal transformation (octonionic module). Matrix $\boldsymbol{W}_8$, belonging to the family of matrices, can be obtained by the superposition of seven Hurwitz-Radon matrices. Moreover, $\boldsymbol{W}_8$ can be seen as the best-adapted orthogonal basis. The output $\boldsymbol{y}$ in (10) is a Haar spectrum of the input vector $\boldsymbol{d}$. It is worth noting that an octonionic module sets up an elementary memory module as well. Designing, for example, an orthogonal filter using (11) and (12), which performs the following transformation:

$$\boldsymbol{y}_{[1]} = \frac{1}{a^2}\left(\boldsymbol{W} + w_0\boldsymbol{1}\right)\boldsymbol{m} = \frac{1}{a}\boldsymbol{T}_8\boldsymbol{m} \tag{13}$$



m_1
$y = (\boldsymbol{m}^{\mathrm{T}}\boldsymbol{x})$
$\boldsymbol{x}$
m_8
$\equiv$
$\boldsymbol{x}$
Memory
Module
$\boldsymbol{m} \rightarrow \boldsymbol{y}_{[1]}$
y_1
$y = a^{-2} \cdot (\boldsymbol{m}^{\mathrm{T}}\boldsymbol{x})$
y_8
$\boldsymbol{y}_{[1]} = [1, \ldots, 1]^{\mathrm{T}}$

FIGURE 1. Implementation of a linear perceptron by an octonionic module, $a = \sqrt{\sum_{i=0}^{7} w_i^2}$.

where: $\boldsymbol{y}_{[1]} = [1, \ldots, 1]^T$, i.e., synthesizing by (10) a flat Haar spectrum for the given input vectors, $\boldsymbol{m}$, so that:

$$w_0 = [1, \ldots, 1] \cdot \boldsymbol{m} > 0 \quad \text{i.e., } \sum_{i=1}^{8} m_i > 0 \tag{14}$$

yields an implementation of a linear perceptron, as shown in Fig. 1.

To summarize the basic considerations above, one can state that the octonionic module is a universal building block to realize very large-scale orthogonal filters and, in particular, memory blocks. Multidimensional, octonionic module-based orthogonal filters can be realized by using the family of Hurwitz-Radon matrices. Thus, a 16-dimensional orthogonal filter can be, for example, determined by the following matrix:

$$\boldsymbol{W}_{16} = \left[\begin{array}{c|c} \boldsymbol{W}_8 & \begin{matrix} w_8 & & \boldsymbol{0} \\ & \ddots & \\ \boldsymbol{0} & & w_8 \end{matrix} \\ \hline \begin{matrix} -w_8 & & \boldsymbol{0} \\ & \ddots & \\ \boldsymbol{0} & & -w_8 \end{matrix} & \boldsymbol{W}_8^{\mathrm{T}} \end{array}\right] = \begin{bmatrix} \boldsymbol{W}_8 & \boldsymbol{0} \\ \boldsymbol{0} & -\boldsymbol{W}_8 \end{bmatrix} + w_8 \begin{bmatrix} \boldsymbol{0} & \boldsymbol{1} \\ -\boldsymbol{1} & \boldsymbol{0} \end{bmatrix} \tag{15}$$

where: $w_8 \in R$, $\boldsymbol{W}_8$—weight matrix of an octonionic module.

Similarly, for the dimension $q = 2^k, k = 5, 6, 7, \ldots$ all Hurwitz-Radon matrices can be found, as:

$$\boldsymbol{W}_{2^k} = \begin{bmatrix} \boldsymbol{W}_{2^{k-1}} & \boldsymbol{0} \\ \boldsymbol{0} & -\boldsymbol{W}_{2^{k-1}} \end{bmatrix} + w_K \begin{bmatrix} \boldsymbol{0} & \boldsymbol{1} \\ -\boldsymbol{1} & \boldsymbol{0} \end{bmatrix} \tag{16}$$

where: $w_K \in R$, $\boldsymbol{1}$—identity matrix

To conclude, one can formulate the following statements:

- A q-dimensional HNN or a q-dimensional orthogonal basis can be created by a compatible connection of octonionic modules.
- The basic function of orthogonal filters is the Haar spectrum analysis of the input data $\boldsymbol{d}$. Particularly, an orthogonal filter performs the function of memory, as given by (13).

Matrix $\boldsymbol{W}_{2^k}$ can be designed as the best-adapted base by using (11) and (12) (i.e., $\boldsymbol{d}$—data, $\boldsymbol{y}$—demanded spectrum).

III. HOPFIELD NEURAL NETWORK-BASED BIORTHOGONAL TRANSFORMATION

It is well known that in data mining, signal processing, and machine learning, two transforms known as principal component analysis (PCA) [21]–[23] and independent component analysis (ICA) [24] are commonly used. PCA can be categorized as an orthogonal, and ICA can be categorized as a biorthogonal transform. Both transformations can be used as a lossless or lossy technique. Thus, for example, lossless PCA and ICA are widely used in blind signal separation (BSS). The Hopfield neural network described by (2) can perform a biorthogonal transformation and can be used for the implementation of a mapping given by training points. Thus, such a biorthogonal transformation can be formulated by a differential equation as follows:

$$\dot{x} = \left(\boldsymbol{W}_{2^k} - w_{\boldsymbol{0}}\boldsymbol{1} + \varepsilon \boldsymbol{W}_s\right)\boldsymbol{\theta}(\boldsymbol{x}) + \boldsymbol{I_d} \tag{17}$$

where: $\boldsymbol{W}_s$—symmetric matrix

$\boldsymbol{W}_{2^k}$ – orthogonal skew-symmetric matrix (16) $\varepsilon \in R$, $w_{\boldsymbol{0}} > 0$

The equilibrium points of the network (17) set up a Hopfield neural network-based biorthogonal filter. A special feature of such a filter is the vector field consisting of anti-symmetric $(\boldsymbol{W}_{2^k})$ and symmetric $(\varepsilon \boldsymbol{W}_s - w_{\boldsymbol{0}}\boldsymbol{1})$ components. This form of vector field can be referred to as a biological-like mechanism consisting of recombination (antisymmetric) and selection (symmetric) components. A neural network (17) can be seen as one possible extension of Hopfield-type neural networks. Moreover, by using such extended structures, some optimization problems (e.g., TSP) can be solved more effectively [25].

IV. BIORTHOGONAL TRANSFORMATION-BASED APPROXIMATION

As mentioned above, the equilibrium points of a biorthogonal filter can set up a nonlinear mapping $\boldsymbol{d} = F(\boldsymbol{x})$, as follows:

Given a set of training points, $S = \{\boldsymbol{x}_i, \boldsymbol{d}_i\}_{i=1}^{N}$, concatenating input vectors $\boldsymbol{x}_i \in R^n$ and output vectors $\boldsymbol{d}_i \in R^m$ in the form:

$$\boldsymbol{u}_i = \begin{bmatrix} \boldsymbol{x}_i \\ \boldsymbol{d}_i \end{bmatrix}; \quad i = 1, \ldots, N \tag{18}$$

where: $\dim \boldsymbol{u}_i = n + m$, $n + m = 2^k$, $k = 3, 4, \ldots$

and using the orthogonal transformation (9), one obtains a Haar spectrum $\boldsymbol{m}_i$ of $\boldsymbol{u}_i$, $i = 1, \ldots, N$ as:

$$\boldsymbol{m}_i = \frac{1}{2}\left(\boldsymbol{W}_{2^k} + \boldsymbol{1}\right)\boldsymbol{u}_i = T(\boldsymbol{u}_i) \tag{19}$$

where: $\boldsymbol{W}_{2^k}$—the Hurwitz-Radon matrix (16)

$\boldsymbol{W}_{2^k}^2 = -\boldsymbol{1}$.

The stable equilibria of the biorthogonal filter (17) constitute the following transformation $T_s(\cdot)$:

$$\left(\boldsymbol{W}_{\boldsymbol{2^k}} - w_0 \cdot \boldsymbol{1} + \varepsilon \cdot \boldsymbol{W}_s\right)\boldsymbol{\theta}(\boldsymbol{x}) + \boldsymbol{u} = \boldsymbol{0} \tag{20}$$

where: $\boldsymbol{u}$—input vector

For $w_0 = 2$, $\varepsilon = 1$, one obtains: $\boldsymbol{m_i} = T_s(\boldsymbol{u_i})$

$$\boldsymbol{m}_i = \left(2 \cdot \boldsymbol{1} - \boldsymbol{W}_s - \boldsymbol{W}_{2^k}\right)^{-\boldsymbol{1}}\boldsymbol{u_i} \tag{21}$$

i.e., $T_s(\cdot) = \left(2 \cdot \boldsymbol{1} - \boldsymbol{W}_s - \boldsymbol{W}_{2^k}\right)^{-\boldsymbol{1}}$

where: $\boldsymbol{W}_s = \boldsymbol{M}\left(\boldsymbol{M}^T\boldsymbol{M}\right)^{-1}\boldsymbol{M}^T$

and

$$\boldsymbol{M} = \{\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_N\}$$

is the spectrum matrix of $\boldsymbol{u_i}$ from (19).

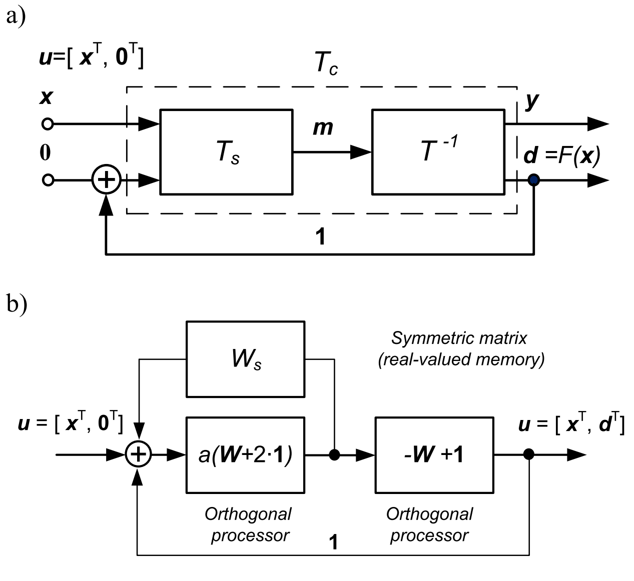


FIGURE 2. Block structure of the approximator (a = 0.2 scaling parameter for $\boldsymbol{W}^2 = -1$).

It should be noted that $\left(\boldsymbol{M}^T\boldsymbol{M}\right)^{-1}\boldsymbol{M}^T$ is the Moore-Penrose pseudoinverse matrix of $\boldsymbol{M}$, i.e., $\boldsymbol{M}^{(+)} = \left(\boldsymbol{M}^T\boldsymbol{M} + \mu\boldsymbol{1}\right)^{-1}\boldsymbol{M}^T$ always exists. Thus, $\boldsymbol{M}\left(\boldsymbol{M}^T\boldsymbol{M} + \mu\boldsymbol{1}\right)^{-1}\boldsymbol{M}^T$, $\mu \neq 0$ can be seen as Tikhonov's regularization [1]. It is clear that the transformation $T_s(\cdot)$ projects training points $\boldsymbol{u}_i$ into $\boldsymbol{m}_i$ as given by (21). Hence, one obtains an inverse transformation:

$$\boldsymbol{u_i} = T^{-1}(\boldsymbol{m}_i) = \left(-\boldsymbol{W}_{2^k} + \boldsymbol{1}\right)\boldsymbol{m}_i \tag{22}$$

i.e., $T^{-1}(\cdot) = \left(-\boldsymbol{W}_{2^k} + \boldsymbol{1}\right)$

The transformations $T_s(\cdot)$ and $T^{-1}(\cdot)$, arranged as a realization of a mapping $F(\boldsymbol{x})$, have the block structure as shown in Fig. 2.

The block structure with "distributed memory," presented in Fig. 2a, can be reconfigured to the form with "lumped memory," as shown in Fig. 2b.

Note 2

It is worth noting that, according to the structure from Fig. 2, such an approximator performs the function of spectrum estimation $\{\hat{\boldsymbol{m}}_i\}$:

$$\hat{\boldsymbol{m}}_i = T_s\left(\begin{bmatrix} \boldsymbol{x}_i \\ \cdots \\ \boldsymbol{0} \end{bmatrix}\right), \quad i = 1, \ldots, N \tag{23}$$

and $\hat{\boldsymbol{d}}_i = \hat{F}(\boldsymbol{x}_i)$ – estimation of the output $\boldsymbol{d}_i$.

Hence, due to the feedback loop action, one implements a recurrence:

$$\begin{aligned} &\hat{\boldsymbol{m}}_i \to \boldsymbol{m}_i, \hat{\boldsymbol{y}}_i \to \boldsymbol{x}_i \\ &\hat{\boldsymbol{d}}_i \to \boldsymbol{d}_i = F(\boldsymbol{x}_i), \quad i = 1, \ldots, N, \end{aligned} \tag{24}$$

at the output of this approximator.

It is easy to note that the structure from Fig. 2 implements an input-output mapping $\phi(\cdot)$, as follows:

$$\boldsymbol{u}_i = \phi(\boldsymbol{u}_i), \quad i = 1, \ldots, N \tag{25}$$

Thus, vectors $\boldsymbol{u}_i, i = 1, \ldots, N$ are invariant points of $\phi(\cdot)$, and vectors $\boldsymbol{d}_i$ are asymptotic centers of attractors $i = 1, \ldots, N$. Moreover, mapping $\phi(\cdot)$ is given by the following matrix transformation:

$$\phi(\cdot) = \boldsymbol{L}(\varepsilon) = \left(-\boldsymbol{W}_{2^k} + \mathbf{1}\right)\left(2 \cdot \mathbf{1} - \boldsymbol{W}_{2^k} - \varepsilon \boldsymbol{W}_s\right)^{-1} \quad (26)$$

and its Lipschitz constant k fulfills:

$$k \le 1 \quad \text{for } \varepsilon \le 1 \quad (27)$$

Hence, $\phi(\cdot)$ is a non-expansive mapping. Note the block $\boldsymbol{T}_c$ in Fig. 2 implementing this mapping. The recurrence is convergent under the linear independence of patterns, and the number of patterns N fulfills:

$$N < 0.5\,(n + m) \quad (28)$$

where: $n + m = \dim \boldsymbol{u}_i$ (18).

V. FEEDFORWARD MODEL OF APPROXIMATOR

One of the general frameworks unifying the different learning algorithms has been formulated by considering a functional of the form [1], [2]:

$$H(f) = \frac{1}{N}\sum_{i=1}^{N} V(y_i, f(\boldsymbol{x}_i)) + \lambda \|f\|_k^2 \quad (29)$$

where: $f : X \to Y \subset R$

$V(\cdot,\cdot)$ –a loss function

Λ—a regularization parameter

$\|f\|_k^2$ –a norm in RKHS (Reproducing Kernel Hilbert Space)

The approximated function $f(\cdot)$ corresponds to the minimum of a functional H for a different loss function $V(f)$ i.e.,

$$H(f) \quad (30)$$

Thus, (29) represents the classical optimization problem solved in Tikhonov's regularization theory [1]. Based on the general framework (29), another model of an approximator has been published [18]. Indeed, given a training set $\{\boldsymbol{x}_i, d_i\}, i = 1, \ldots, N; \boldsymbol{x}_i \in R^n, d_i \in R$, the function $F(\boldsymbol{x})$ can be implemented as:

$$F(\boldsymbol{x}) = \sum_{i=1}^{N} c_i K_i(\boldsymbol{x}_i, \boldsymbol{x}) \quad (31)$$

where: $K_i(\boldsymbol{x}_i, \boldsymbol{x})$ are defined by the function:

$$K_i(\boldsymbol{x}_i, \boldsymbol{x}) = \Theta(\langle \boldsymbol{x}_i, \boldsymbol{W}_n, \boldsymbol{x}\rangle) \quad (32)$$

and: $\boldsymbol{x}_i \in R^n$ is the i-th training vector

$\boldsymbol{W}_n$–skew-symmetric matrix

$\Theta(\cdot)$ –an odd function (e.g., sigmoidal)

$\langle \boldsymbol{x}_i, \boldsymbol{W}_n \boldsymbol{x}\rangle$ -a scalar product

Thus, the Gram matrix $(N \times N)$:

$$\boldsymbol{K} = \{K_{ij}\} = \{K(x_i, x_j)\} \quad (33)$$

is skew-symmetric, and the key design equation is as follows:

$$\boldsymbol{K}_R \boldsymbol{c} = \boldsymbol{d}$$

and

$$\boldsymbol{c} = \boldsymbol{K}_R^{-1} \boldsymbol{d} \quad (34)$$

where: $\boldsymbol{d}^T = [d_1, \ldots, d_N]$

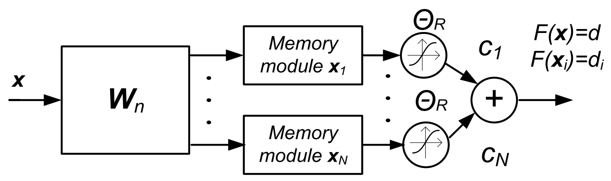


FIGURE 3. Block structure of a feedforward approximator.

$\boldsymbol{K}_R$–regularized matrix i.e., $\boldsymbol{K}_R = \boldsymbol{K} + diag\boldsymbol{R}_i R_i \neq 0, i = 1, \ldots, N$

Hence:

$$F(\boldsymbol{x}) = \sum_{i=1}^{N} c_i \theta_{R_i}(\boldsymbol{x}_i, \boldsymbol{W}_n \boldsymbol{x}) \quad (35)$$

where: $\theta_{R_i} = \theta(\cdot) + R_i \delta(\cdot)$, $\delta(\cdot)$—the Kronecker function.

A block structure of a feedforward approximator is shown in Fig. 3.

It is worth noting that due to the skew symmetry of the Gram matrix in (32), regularization parameter R_i can be used as a smoothness regularizer. Moreover, for $\boldsymbol{W}_n = \boldsymbol{W}_{2^n}$ (16),(32) $K_i(\boldsymbol{x}_i, \boldsymbol{x})$ consists of the scalar product of training vectors with their Haar spectra.

VI. ON FEATURES OF APPROXIMATORS

The essential function of approximators described in this study is the implementation of a mapping defined by a training set. However, some properties of the approximator shown in Fig. 2 are worth noting. They can be used to solve tasks typical for big data processing and machine learning.

A. DATA RECONSTRUCTION

Given a training set $S = \{\boldsymbol{x}_i, \boldsymbol{d}_i\}, i = 1, \ldots, N$ where $\boldsymbol{x}_i \in R^n, \boldsymbol{d}_i \in R^m, n + m = 2^k, k = 3, 4, \ldots$, the mapping $\boldsymbol{d}_i = F(\boldsymbol{x}_i)$ can be implemented as a biorthogonal filter equipped with the following data reconstruction property:

Let us assume that input patterns $\boldsymbol{x}_p$ are a distorted or compressed form of $\boldsymbol{x}_i$. Hence, changing the input vectors and the structure of the feedback loop, as follows:

$$\begin{bmatrix} \boldsymbol{x}_i \\ \cdots \\ \mathbf{0} \end{bmatrix} \begin{matrix} \}n \\ \}m \end{matrix} \to \begin{bmatrix} \boldsymbol{x}_p \\ \cdots \\ \mathbf{0} \end{bmatrix} \begin{matrix} \}pn \\ \}(1-p)\,n + m \end{matrix} \quad (36)$$

where fraction p > 0.1 (according to numerical tests) and $\boldsymbol{x}_p$ is the preserved part of $\boldsymbol{x}_i$, one achieves the full pattern reconstruction, i.e.,$\boldsymbol{d}_i = F(\boldsymbol{x}_p)$. This property is illustrated by Lena's photo reconstruction.

B. IMPLEMENTATION OF ASSOCIATIVE MEMORY

Implementation of associative memory by the approximator:

$$\text{i.e.,} \quad F(\boldsymbol{x}_i) = z_i, \quad i = 1, \ldots N \quad (37)$$

where: $\boldsymbol{x}_i$—key vectors

z_i–memory vectors

is realizable by defining a training set of the form:

$$S = \{\boldsymbol{x}_i, z_i\}, \quad i = 1, \ldots, N \quad (38)$$

and $\boldsymbol{u}_i = \begin{bmatrix} \boldsymbol{x}_i \\ z_i \end{bmatrix}$.

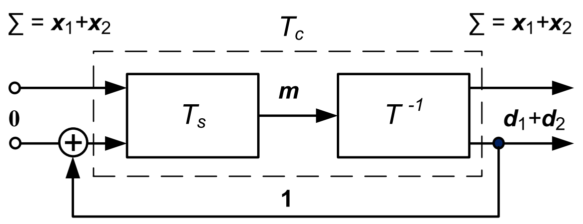


FIGURE 4. Approximator as an analog processor: addition.

It is easy to see that due to the reconstruction property mentioned above, the memorized vectors z_i can be retrieved by the distorted or incomplete key vectors $\boldsymbol{x}_p$ (36). However, note that the stored vectors $\boldsymbol{u}_i$ are not attractors. Hence, any input vector $\boldsymbol{v}_{in} \neq x_i$ is not associated with z_i, $i = 1, \ldots, N$.

This means that:

$$\phi \begin{bmatrix} \boldsymbol{x}_i \\ \boldsymbol{0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_i \\ z_i \end{bmatrix}, \quad i = 1, \ldots, N$$

However, for $\boldsymbol{v}_{in} \neq \boldsymbol{x}_i$ one obtains:

$$\phi \begin{bmatrix} \boldsymbol{v}_{in} \\ \boldsymbol{0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{v}_{out} \\ \boldsymbol{m}_{out} \end{bmatrix}$$

Thus, such an input vector is a key a vector only if $\boldsymbol{v}_{in} = \boldsymbol{v}_{out}$, and vector $\boldsymbol{m}_{out}$ belongs to the memory.

Moreover, any superposition $\boldsymbol{x}_s$ of key vectors retrieves the associated superposition of the memorized vectors z_i:

$$\boldsymbol{0} \neq \boldsymbol{x}_s = \sum\nolimits_{k=1}^{N} \alpha_k \boldsymbol{x}_k \tag{39}$$

$$F(\boldsymbol{x}_s) = \sum\nolimits_{k=1}^{N} \alpha_k z_k$$

where: $\alpha_k \in R$.

This property allows one to use the approximator as a processor, as pointed out below.

C. MODEL OF ANALOG PROCESSOR

The model from Fig. 2 can be referred to as a superposition processor, performing the addition and multiplication of real numbers. Indeed, for the simplicity of presentation, let us consider the component-wise addition of two vectors $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$. Defining two system vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ with an *a priori* known sum: $\sum = \boldsymbol{x}_1 + \boldsymbol{x}_2$ $(\boldsymbol{x}_1 \neq \boldsymbol{x}_2)$, one implements $(\boldsymbol{d}_1 + \boldsymbol{d}_2)$, as presented in Fig. 4.

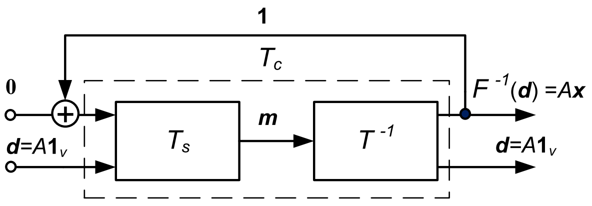
Note that:

$$\phi \begin{bmatrix} \boldsymbol{x}_1 \\ \cdots \\ \boldsymbol{0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_1 \\ \cdots \\ \boldsymbol{d}_1 \end{bmatrix}, \phi \begin{bmatrix} \boldsymbol{x}_2 \\ \cdots \\ \boldsymbol{0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_2 \\ \cdots \\ \boldsymbol{d}_2 \end{bmatrix}$$

and

$$\phi \begin{bmatrix} \sum \\ \cdots \\ \boldsymbol{0} \end{bmatrix} = \phi \begin{bmatrix} \boldsymbol{x}_1 + \boldsymbol{x}_2 \\ \cdots \\ \boldsymbol{0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_1 + \boldsymbol{x}_2 \\ \cdots \\ \boldsymbol{d}_1 + \boldsymbol{d}_2 \end{bmatrix} \tag{40}$$

It is clear that the functioning of the processor is based on the data reconstruction property. This schema can be extended to the addition of N vectors. Multiplication by such a superposition processor can be realized as the inverse operation $\boldsymbol{d} = \boldsymbol{F}^{-1}(\boldsymbol{x})$ using the structure from Fig. 2 with a modified feedback loop as presented in Fig. 5.



1
T_c
0
$F^{-1}(\boldsymbol{d}) = A\boldsymbol{x}$
T_s
m
T^{-1}
$\boldsymbol{d}=A\boldsymbol{1}_v$
$\boldsymbol{d}=A\boldsymbol{1}_v$

FIGURE 5. Approximator as an analog processor: multiplication.

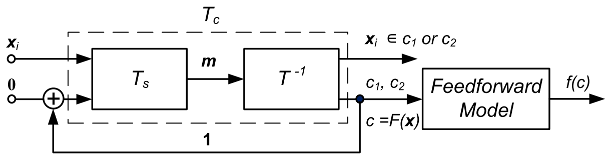


FIGURE 6. Structure of a pattern recognizer.

Thus, a component-wise multiplication of a given vector $\boldsymbol{x}$ by a number $A \in R$ i.e., $A \cdot \boldsymbol{x}$, can be realized as follows:

$$\phi \left(\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{0} \end{bmatrix} \right) = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{1}_v \end{bmatrix} \quad \text{i.e.,} \; F(\boldsymbol{x}) = \boldsymbol{1}_v \tag{41}$$

where: $\dim x = \dim 1_v = \frac{1}{2} 2^n$, $n = 3, 4, \ldots$

$$\boldsymbol{1}_v = [1, \ldots, 1]^T$$

Hence, the inverse operation (Fig. 5) is realized by the same T_c, changing only the feedback loop:

$$\phi \left(\begin{bmatrix} \boldsymbol{d} \\ \boldsymbol{0} \end{bmatrix} \right) = \phi \left(\begin{bmatrix} \boldsymbol{1}_v \\ \boldsymbol{0} \end{bmatrix} \right) = \begin{bmatrix} \boldsymbol{1}_v \\ \boldsymbol{x} \end{bmatrix},$$
$$\phi \left(\begin{bmatrix} A \cdot \boldsymbol{1}_v \\ \boldsymbol{0} \end{bmatrix} \right) = \begin{bmatrix} A \cdot \boldsymbol{1}_v \\ A \cdot \boldsymbol{x} \end{bmatrix}, \tag{42}$$

i.e., $F^{-1}(\boldsymbol{1}_v) = \boldsymbol{x}$ and $F^{-1}(A \cdot \boldsymbol{1}_v) = A \cdot \boldsymbol{x}$

D. PATTERN RECOGNITION

Pattern recognition can be implemented by the approximator defining a training set $S = \{\boldsymbol{x}_i, \boldsymbol{d}_i\}$ as a set of patterns/vectors assigned to one of a prescribed number of classes. For simplicity of presentation, let us consider two classes of recognition, i.e.

$$S = \{\boldsymbol{x}_i, \boldsymbol{d}_i\}_{i=1}^{N} = \left\{ \{\boldsymbol{x}_i, \boldsymbol{c}_1\}_{i=1}^{N_1}, \{\boldsymbol{x}_i, \boldsymbol{c}_2\}_{i=N_1+1}^{N} \right\} \tag{43}$$

where: $\boldsymbol{c}_1, \boldsymbol{c}_2$—a signature of prescribed classes $(\boldsymbol{c}_1 \neq \boldsymbol{c}_2)$
$\boldsymbol{x}_i$—input patterns.

The structure of such a pattern recognizer is shown in Fig. 6.

Note that the structure of the approximator can be augmented as in Fig. 6. by the above-described feedforward model [18]. Thus, vectors $\boldsymbol{c}$ can be interpreted as features of input patterns. Moreover, this pattern recognizer can classify incomplete vectors deploying the reconstruction property. It is worth noting that the dimension of the Gram matrix (33) is defined by the number of prescribed classes.

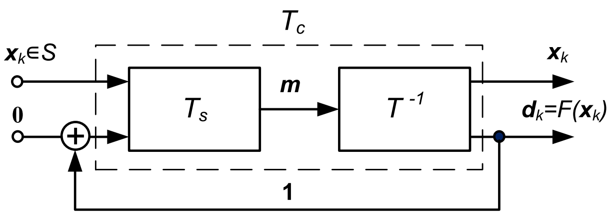


FIGURE 7. Machine learning model featuring "parallelism."

E. SUPERPOSITION-BASED PARALLELISM

It is well known that quantum computations, and hence QNN, are based on quantum parallelism, which is the result of quantum state superposition. The structure of the machine learning presented in Fig. 2 can be categorized as a computation model featuring "parallelism" as well, because it features superpositions of the processed vectors. Indeed, let us assume that a given set of training vectors $S = \{\boldsymbol{x}_i, \boldsymbol{d}_i\}_{i=1}^N$ is generated by a linear mapping $F(\cdot)$:

$$F(\boldsymbol{x}_i) = \boldsymbol{d}_i, \quad i = 1, \ldots, N \tag{44}$$

where: $\dim(\boldsymbol{x}_i) = m$.

Under the assumption that in the set S there are m linearly independent vectors: $\boldsymbol{x}_i, \ldots, \boldsymbol{x}_m$, one obtains the following superposition:

$$\boldsymbol{x}_k = \propto_1 \boldsymbol{x}_1 + \ldots + \propto_m \boldsymbol{x}_m, \quad \boldsymbol{x}_k \in S \tag{45}$$

and: $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_m]^T = \boldsymbol{X}^{-1}\boldsymbol{x}_k$

where: $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m]$—computational basis matrix.

Hence, the structure of a mapping $F(\cdot)$ is supported by m vectors:

$$\boldsymbol{u}_i = \begin{bmatrix} \boldsymbol{x}_i \\ \boldsymbol{d}_i \end{bmatrix}, \quad i = 1, \ldots, m$$

and

$$\phi \begin{bmatrix} \boldsymbol{x}_k \\ 0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{d}_k \end{bmatrix}, \quad \boldsymbol{x}_k \in S \tag{46}$$

where: $\boldsymbol{x}_k = \sum_{i=1}^m \alpha_i \boldsymbol{x}_i$ and $\boldsymbol{d}_k = \sum_{i=1}^m \alpha_i \boldsymbol{d}_i$.

Thus, the look-up table $S = \{\boldsymbol{x}_i, \boldsymbol{d}_i\}_{i=1}^N$ can be, according to (44), implemented by the structure shown in Fig. 7.

It is worth noting that a structure similar to the mapping $F(\cdot)$ is used for solutions to different linear equations (see further examples below) and linear transformations.

VII. COMPUTATIONAL VERIFICATIONS

Example 1:

The model of machine learning described above was used for image reconstruction of incomplete and distorted patterns. As a test image, a grey photo of Lena was used, having resolution of 512 × 512 pixels, which is commonly used to investigate algorithms for image compression and processing (Fig. 8). This photo was written in the MATLAB program in the form of matrix $\boldsymbol{X}_{lena}$ with a size of 512 × 512. The $\boldsymbol{X}_{lena}$ was a source of a different number of patterns (columns or rows). The following sets of columns were analyzed: N = 4, 16, 32, 64. The analysis of the full image was therefore a sequence of 128, 32, 16, and 8 partial analyses, respectively.

FIGURE 8. Test image.

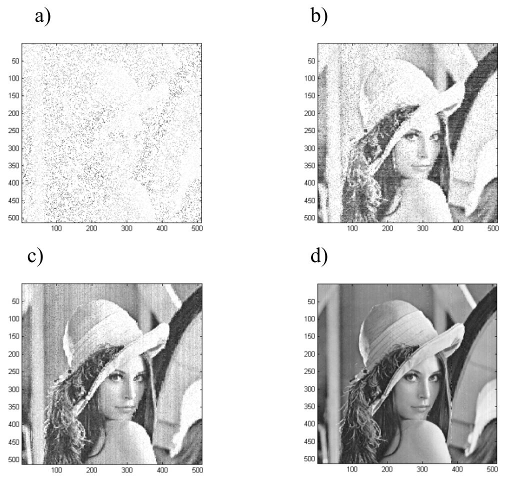


FIGURE 9. Image reconstruction with a level of distortion of 90%, 4-column set of patterns: a) distorted image, b) reconstruction result after 5 iterations, c) reconstruction result after 20 iterations, d) reconstruction after 100 iterations.

Using the approximator from Fig. 2, the input patterns were (36):

$$\begin{bmatrix} \boldsymbol{x}_p \\ \cdots \\ \boldsymbol{0} \end{bmatrix} \begin{matrix} \}52 \\ \}460 \end{matrix}$$

where: $\boldsymbol{x}_p$—the preserved part of the photo columns.

This means that the input patterns were distorted by randomly removing about 90% of the pixels. The convergence of the iteration process is illustrated in Figs. 9, 10, and 11. The images show the result of photo reconstruction after 5, 20, and 100 iterations, respectively. The best reconstruction results were achieved for N = 4.

In the next experiment, some information was removed from the analyzed image, leaving only a narrow band of row patterns. From a formal point of view, in matrix $\boldsymbol{X}_{lena}$, the rows from 200 to 280 were left. The original image and the reconstruction results after 5, 20, and 100 iterations are presented in Fig. 12. It is worth noting that in the case of a simultaneous analysis of the whole image, the dimension of the approximator should be appropriately increased to

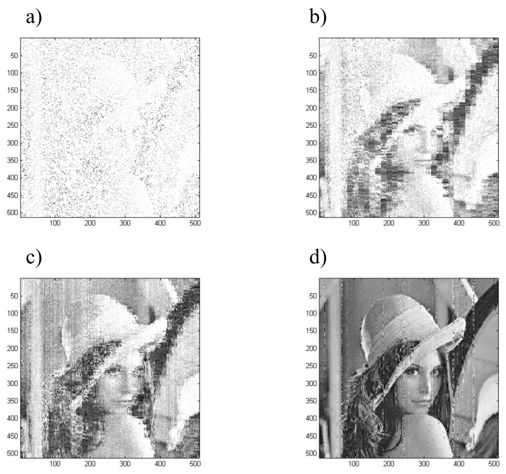


FIGURE 10. Image reconstruction with a level of distortion of 90%, 16-column set of patterns: a) distorted image, b) reconstruction result after 5 iterations, c) reconstruction result after 20 iterations, d) reconstruction after 100 iterations.

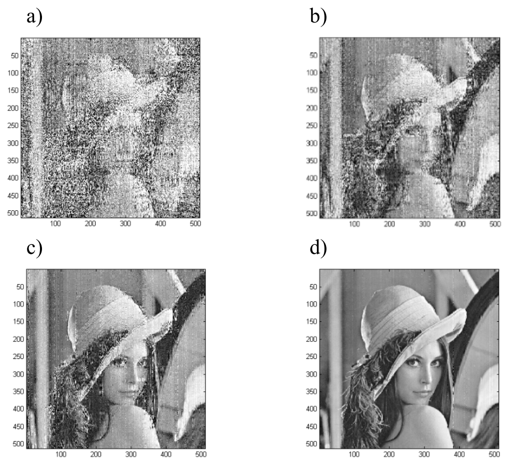


FIGURE 11. Image reconstruction with a level of distortion of 90%, and after 100 iterations: a) 64-column sets, b) 32-column sets, c) 16-column sets, d) 4-column sets.

allow the processing of an image as a single column vector. Moreover, in these numerical experiments, matrix $\boldsymbol{W}_{2^k} = \{1, -1, 0\}$ was deployed.

Example 2:

Many machine learning algorithms and applications are based on solutions of linear equations [15]. One considers here the following problem:

$$F(\boldsymbol{x}) : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{47}$$

where: $\boldsymbol{A}$—$(m \times n)$ real matrix, $m \neq n$.

$\boldsymbol{b}$−(m $\times$1) real vector, $\boldsymbol{x}$- $(n\times 1)$ real vector

$m + n = 2^k, \quad k = 3, 4, \ldots,$

$m < \frac{1}{2}(m + n)$

One considers two cases.

Case 1: $m < n$.

To solve this equation using the structure from Fig. 2, one first generates a training set: $\{\boldsymbol{x}_i, \boldsymbol{b}_i\}_{i=1}^{N}$,

i.e.,

$$\boldsymbol{A}\boldsymbol{x}_i = \boldsymbol{b}_i, \quad i = 1, \ldots, N = m \tag{48}$$

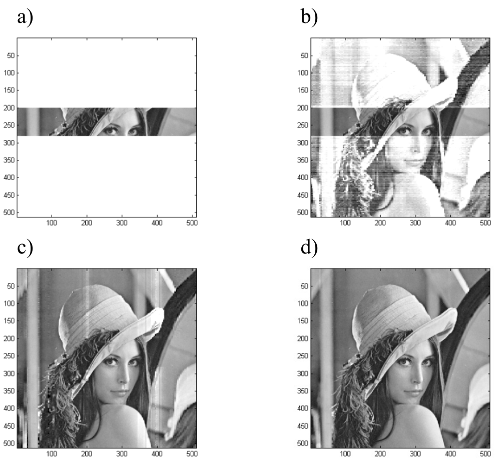


FIGURE 12. Image reconstruction—rows 200–280 were left in the original image: a) distorted image, b) reconstruction after 5 iterations, c) reconstruction after 20 iterations, d) reconstruction after 100 iterations.

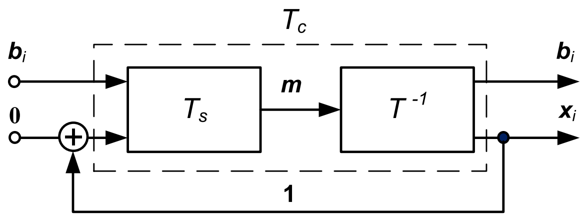


FIGURE 13. Block presentation of equation (49).

Under the assumption $\boldsymbol{b}_i \neq \boldsymbol{b}_k$, for $i \neq k$, the synthesis of a mapping $F(\boldsymbol{x})$ gives:

$$\phi \begin{bmatrix} \boldsymbol{b}_i \\ \boldsymbol{0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_i \\ \boldsymbol{x}_i \end{bmatrix}, \quad \text{i.e., } F^{-1}(\boldsymbol{b}_i) = \boldsymbol{x}_i,\ i = 1, \ldots, m \tag{49}$$

where: $\begin{bmatrix} \boldsymbol{b}_i \\ \boldsymbol{x}_i \end{bmatrix} = \boldsymbol{u}_i, i = 1, \ldots, m,$

as shown in Fig. 13.

Formulating $\boldsymbol{b}$ as a superposition of pattern $\boldsymbol{b}_i$; i.e.,

$$\boldsymbol{b} = \sum\nolimits_{i=1}^{m} \alpha_i \boldsymbol{b}_i \tag{50}$$

where: $[\alpha_1, \ldots, \alpha_m]^T = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m]^{-1}\boldsymbol{b} = \boldsymbol{B}^{-1}\boldsymbol{b}$

$\boldsymbol{B} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m]$ −non-singular quadratic matrix $(m \times m)$ (m-linearly independent vectors),

one obtains a solution to (47) as follows:

$$\phi \begin{bmatrix} \sum_{i=1}^{m} \alpha_i \boldsymbol{b}_i \\ \boldsymbol{0} \end{bmatrix} = \phi \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b} \\ \sum_{i=1}^{m} \alpha_i \boldsymbol{x}_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{x} \end{bmatrix} \tag{51}$$

Hence, the structure shown in Fig. 13 is the model of a mapping ϕ and it sets up the solutions to (47).

It is worth noting that:

1. $\boldsymbol{x}_i$ can be chosen as orthogonal vectors, $i = 1, \ldots, m$.
2. The structure shown in Fig. 14 gives solutions $\boldsymbol{x}$ to the linear equation (47) for any $\boldsymbol{b} \prec \infty$. It is clear that values of the vector $[\alpha_1, \ldots, \alpha_m]^T$ are not necessary to solve this linear equation by using the structure from Fig. 14. However, they must exist, i.e., $|\alpha_i| < \infty, i = 1, \ldots, m$.

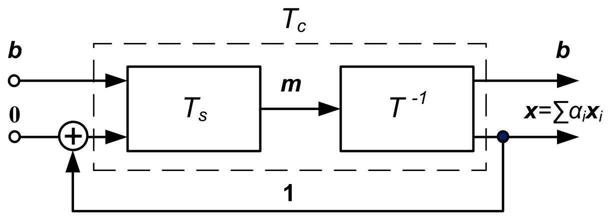


FIGURE 14. Structure of an inverse mapping model $F^{-1}(\boldsymbol{b}) = \boldsymbol{x}$.

3. Once designed, a structure of the mapping model presented in Fig. 14 can deliver any number of exact solutions for the equation $\boldsymbol{Ax} = \boldsymbol{b}$ by generating different training sets: $S_k = \{\boldsymbol{x}_i, \boldsymbol{b}_i\}_{i=1}^{N}$, $k = 1, 2, \ldots$
4. It is well known [26] that the least-squares (LS) solution to (47) is:

$$\boldsymbol{x}^* = \boldsymbol{A}^T(\boldsymbol{AA}^T)^{-1}\boldsymbol{b}$$

Case 2: $m > n$.

Linear equations with dimensions fulfilling $m > n$ can be solved only approximately. The training set $\{\boldsymbol{x}_i, \boldsymbol{b}_i\}_{i=1}^{m}$ constitutes matrix $\boldsymbol{B}$ (50), which is singular. This means that due to the singularity of matrix $\boldsymbol{B}$ ($m \times m$), a mapping $F^{-1}(\boldsymbol{b}_i) = \boldsymbol{x}_i$ cannot be determined.

However, the solutions to (47) for $m > n$ can be obtained by using an LS estimator of the superposition parameters α_i (50). Thus, for any given vector $\boldsymbol{b}$ ($m \times 1$), n superposition parameters α_i can be set up by the least-squares approximation formula:

$$\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_n]^T = \left[\boldsymbol{B}^T\boldsymbol{B} + \lambda\boldsymbol{1}\right]^{-1}\boldsymbol{B}^T\boldsymbol{b} \tag{52}$$

where: $\boldsymbol{1}$—identity matrix

$\lambda \geq 0$

Hence:

$$\boldsymbol{b} \approx \boldsymbol{b}_s = \sum\nolimits_{i=1}^{n} \alpha_i\boldsymbol{b}_i \; (\lambda = 0)$$

and

$$\boldsymbol{b}_s(\lambda) = \sum\nolimits_{i=1}^{n} \alpha_i(\lambda)\boldsymbol{b}_i, \; (\lambda > 0). \tag{53}$$

It is worth noting that a positive parameter λ(*i.e.*, $\lambda > 0$) can be for $\boldsymbol{b}_i \succ 0$ set up to obtain $\alpha_i > 0$, $i = 1, \ldots, n$, and hence $\boldsymbol{x} \succ 0$. The machine learning model for a solution of linear equations $\boldsymbol{Ax} = \boldsymbol{b}$ is presented in Fig. 15. It is well known [26] that the LS solution of equation (47) is given by:

$$\boldsymbol{x}^* = \left(\boldsymbol{AA}^T\right)^{-1}\boldsymbol{Ab}$$

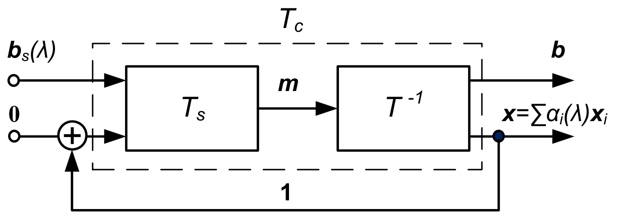
Thus one obtains:

$$\left\|\boldsymbol{x} - \boldsymbol{x}^*\right\| = 0.$$

Note that for the "regularized" least squares in (53), i.e.,

$$\boldsymbol{b}_s(\lambda) = \sum\nolimits_{i=1}^{n} \alpha_i(\lambda)\boldsymbol{b}_i, \quad \lambda > 0$$

the exact solution $\boldsymbol{x}$ is also set up by the structure from Fig. 15. It is well known that nonnegative matrix factorization (NMF) has recently been used for modeling many real-life applications from the field of signal and data processing [22]. NMF is formally defined as



T_c
$\boldsymbol{b}_s(\lambda)$
$\boldsymbol{0}$
T_s
$\boldsymbol{m}$
T^{-1}
$\boldsymbol{b}$
$\boldsymbol{x}=\sum\alpha_i(\lambda)\boldsymbol{x}_i$
1

$\boldsymbol{x}$ – exact solution: $\boldsymbol{Ax} = \boldsymbol{b}_s(\lambda)$
$\boldsymbol{x}$ – approximate solution of the equation: $\boldsymbol{Ax} = \boldsymbol{b}, \boldsymbol{x} \succ 0, \lambda > 0$

FIGURE 15. Machine learning model for a solution to linear equation $\boldsymbol{Ax} = \boldsymbol{b}, \boldsymbol{m} > \boldsymbol{n}$.

$$\boldsymbol{B}_d \approx \boldsymbol{AX}, \boldsymbol{A} \in R^{m\times r}, \quad \boldsymbol{X} \in R^{r\times n} \tag{54}$$

or as

$$min\,\|\boldsymbol{B}_d - \boldsymbol{AX}\|_F^2 \tag{55}$$

where: $\boldsymbol{B}_d$—data matrix

$\boldsymbol{A}$– basis matrix, $r \leq \min\{m, n\}$.

A nice review paper on NMF can be found in [27]. It should be clear that (47) for case 2 (i.e., $m > n$) can be interpreted as an NMF subproblem under the assumptions: $\boldsymbol{x} \succ 0$ (column of $\boldsymbol{X}$), $\boldsymbol{A} \succ 0, \boldsymbol{A} \in R^{m\times r}$, and $\boldsymbol{b}$—column of $\boldsymbol{B}_d$. Thus, one generates the training set $\{\boldsymbol{x}_i, \boldsymbol{b}_i\}_{i=1}^{m}$, where $\boldsymbol{x}_i \succ 0$, $\boldsymbol{b}_i \succ 0$. Due to the above-described regularization, the machine learning model for NMF is shown as in Fig. 15. The model is adequate for all columns of the data matrix $\boldsymbol{B}_d$.

Example 3:

The structure of machine learning presented in Fig. 2. can be categorized as a computation model featuring "parallelism." Hence, by using this model, "oracle" or "black box" problems [28] can be solved by performing only one query to the oracle. Indeed, a black box computes a function

$$f : \{\boldsymbol{x}_i\}_{i=1}^{2N} \to \{0, c\} \tag{56}$$

where: $\boldsymbol{x}_i \in R^n$, $n + 1 = 2^k$, $k = 4, 5, \ldots$.

$2N < \frac{1}{2}(n + 1)$.

Moreover, one assumes that f is either constant ($f(\boldsymbol{x}) = c$ for all $\boldsymbol{x}$) or balanced ($f(\boldsymbol{x}) = 0$ for 1/2 of the possible input vectors). It is clear that the decision problem of whether f is constant or balanced can be solved by accessing the black box from Fig. 2 only once, using input $\boldsymbol{s}$ in the form of superposition:

$$\boldsymbol{s} = \sum\nolimits_{i=1}^{2N} \boldsymbol{x}_i \tag{57}$$

Hence:

$$\begin{aligned}\phi\begin{bmatrix}\sum_{i=1}^{2N}\boldsymbol{x}_i\\0\end{bmatrix} &= \begin{bmatrix}\sum_{i=1}^{2N}\boldsymbol{x}_i\\2N\cdot c\end{bmatrix} \quad \text{for } f \text{ constant}\\ &= \begin{bmatrix}\sum_{i=1}^{2N}\boldsymbol{x}_i\\N\cdot c\end{bmatrix} \quad \text{for } f \text{ balanced.}\end{aligned} \tag{58}$$

This means that one query suffices to recognize a constant and balanced function.

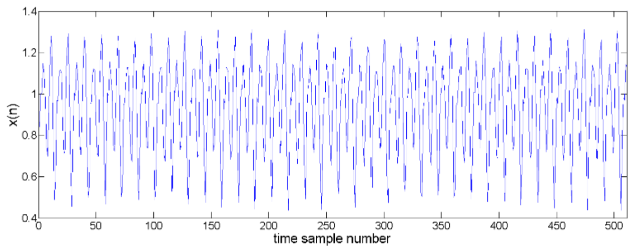


FIGURE 16. Time series generated by the Mackey–Glass system.

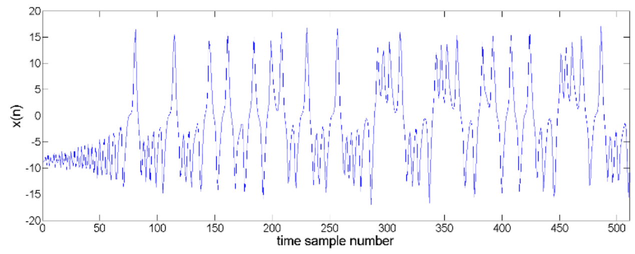


FIGURE 17. Time series generated by the Lorenz system.

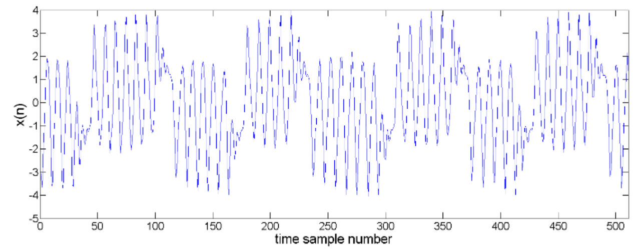


FIGURE 18. Time series generated by the Chua circuits.

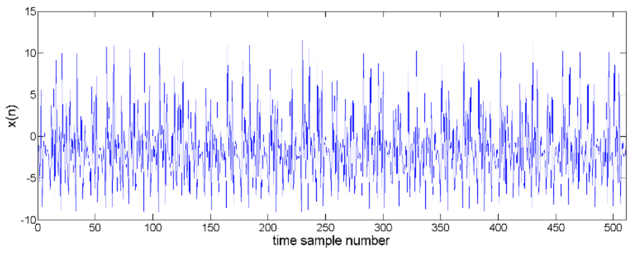


FIGURE 19. Time series generated by the Rössler equation.

Example 4

The model from Fig. 2 was used for the recognition of patterns generated by four deterministic chaos systems, i.e., Mackey–Glass's [29], Lorenz's [30], Chua's [31], and Rössler's [32] structures. Thus, sixteen 512-dimensional patterns/vectors were used as learning vectors to determine four classes: class (1) (Mackey–Glass), class (2) (Lorenz), class (3) (Chua), class (4) (Rössler). Hence, four learning vectors for every class were determined as:

$$\boldsymbol{u}_i = [\boldsymbol{x}_i; d_i]^T = [x(n_i - 511), x(n_i - 510), \ldots, x(n_i); d_i]^T$$

where: $i = 1, 2, 3, 4$

n_i–the number of a time sample, $n_i = 505 + 7i$ (time shift equals 7)

d_i– class signature

1. $d_i = 5$ for vectors of class (1), and $x(n)$ are the time samples from the solution to the Mackey–Glass equation.
2. $d_i = -5$ for vectors of class (2), and $x(n)$ are the time samples from the solution to the Lorenz equations.
3. $d_i = -2$ for vectors of class (3), and $x(n)$ are the time samples from the solution to the Chua equations.
4. $d_i = 2$, for vectors of class (4), and $x(n)$ are the time samples from the solution to the Rössler equations.

As mentioned above, the main feature of the model from Fig. 2 is that it can classify incomplete patterns. Indeed, with only about 150 components of the 512-dimensional vectors, recognition and classification were correct. Moreover, using the data in this example, the oracle-like problem was solved as an illustration of parallelism. Indeed, the membership of a group of vectors in a given class can be set up by performing only one query.

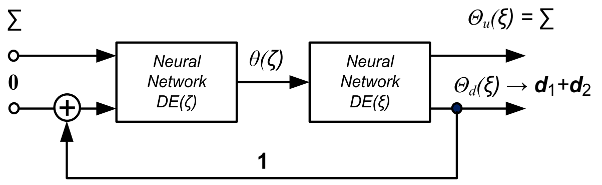


FIGURE 20. Implementation of the approximator constituting the analog processor.

VIII. IMPLEMENTATION OF AN APPROXIMATOR BY A DYNAMIC NEURAL NETWORK MODEL

As mentioned above, the basic structure of the approximator shown in Fig. 2 resolves the equilibria of Hopfield neural network-based biorthogonal filters. Hence, the structure of the approximator from Fig. 2 and the analog processor can be implemented as a ring of neural networks, as shown in Fig. 20.

It is easy to see that the structure from Fig. 20 is described by the following differential equations:

$$DE(\boldsymbol{\zeta}): \frac{d\boldsymbol{\zeta}}{dt} = (\boldsymbol{W} - 2\cdot\boldsymbol{1} + \boldsymbol{W}_s)\,\boldsymbol{\theta}(\boldsymbol{\zeta}) + \begin{bmatrix} \boldsymbol{\Sigma} \\ \cdots \\ \boldsymbol{0} + \boldsymbol{\theta}_d(\boldsymbol{\xi}) \end{bmatrix}$$

$$DE(\boldsymbol{\xi}): \frac{d\boldsymbol{\xi}}{dt} = \frac{1}{2}(-\boldsymbol{W} - \boldsymbol{1}) \begin{bmatrix} \boldsymbol{\theta}_u(\boldsymbol{\xi}) \\ \boldsymbol{\theta}_d(\boldsymbol{\xi}) \end{bmatrix} + \boldsymbol{\theta}(\boldsymbol{\zeta}) \qquad (59)$$

where: $\boldsymbol{\theta}(\boldsymbol{\zeta}) = \begin{bmatrix} \boldsymbol{\theta}_u(\boldsymbol{\zeta}) \\ \boldsymbol{\theta}_d(\boldsymbol{\zeta}) \end{bmatrix}$.

Hence, the steady-state solutions of these equations are:

$$\theta_d(\boldsymbol{\xi}) \to \boldsymbol{d}_1 + \boldsymbol{d}_2$$
$$\boldsymbol{\theta}_u(\boldsymbol{\xi}) \to \sum \qquad (60)$$

In order to realize the unlimited number range of the processor, one can assume the linearity of activation functions: $\boldsymbol{\theta}(\boldsymbol{\zeta}) = \boldsymbol{\zeta}$, $\boldsymbol{\theta}(\boldsymbol{\xi}) = \boldsymbol{\xi}$. It is worth noting again that in the examples of computational verifications, the orthogonal matrix $\boldsymbol{W}$ has the form $\boldsymbol{W} = \{-1, 0, 1\}$.

IX. Q-INSPIRED MODEL OF MACHINE LEARNING

As mentioned in the Introduction, Q-inspired neural networks are a non-quantum version of CVNN. In this study, we show that the extended model of Hopfield-type neural networks can be a source of different types of computational models. Thus, the Q-inspired Hopfield-type neural network is given by (3). The machine learning model, implemented by such

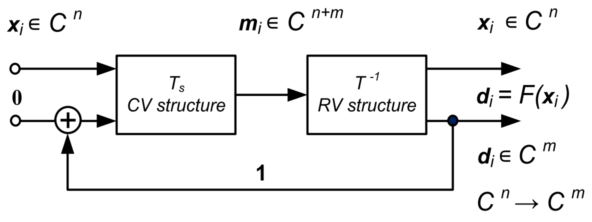


FIGURE 21. Structure of the complex-valued approximator.

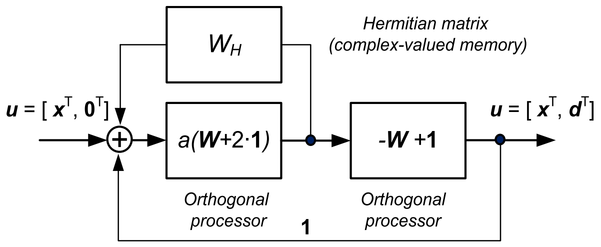


FIGURE 22. Block structure of the complex-valued approximator (a = 0.2—scaling parameter for $W^2 = -1$).

a Q-inspired neural network, is determined by the following statements:

Statement 1

The real-valued octonionic module defined by (10),(11), (12) is for complex input vectors $\boldsymbol{d} \in C^8$ transformed to a complex-valued module, because its transformation matrix $\boldsymbol{T}_8$ is unitary, i.e., $\boldsymbol{T}_8 \cdot \boldsymbol{T}_8^\dagger = \mathbf{1}$. Hence, the orthogonal matrix $\boldsymbol{W}_{2^k}$ (15),(16) consisting of the octonionic modules can be transformed to a unitary matrix by using unitary octonionic modules.

Statement 2

Given a set of complex-valued training points $\{\boldsymbol{x}_i, \boldsymbol{d}_i\}_{i=1}^N$, where $\boldsymbol{x} \in C^n, \boldsymbol{d_i} \in C^m, n+m = 2^k, k = 3, 4, \ldots$, using the above considered biorthogonal transformation, one obtains an implementation of a mapping given by complex training points, i.e., $C^n \rightarrow C^m$. The complex-valued structure of an approximator is shown in Fig. 21.

It is clear that the complex-valued approximator can be implemented as a complex-valued neural network or as an algebraic mapping: $C^n \rightarrow C^m$. Similar to the block structure in Fig. 2b, the complex-valued approximator can be represented as in Fig. 22, where "lumped memory" is implemented by the Hermitian matrix.

The computational efficiency of the complex-valued approximator is greater than that of the real-valued approximator. To illustrate such a feature, we construct in Example 5 a discrete Fourier transform using a property called superposition-based parallelism, described above (Section 6, E).

Example 5

Given a set of training vectors $S = \{\boldsymbol{x}_i, \boldsymbol{d}_i\}_{i=1}^L$, one aims to compute:

$$DFT\,\{\boldsymbol{x}_i\} = \boldsymbol{d}_i, \quad i = 1, \ldots, L \tag{61}$$

where: $dim\boldsymbol{x}_i = dim\boldsymbol{d}_i = m, m = 2^k, k = 3, 4, \ldots$

$\boldsymbol{d}_i \in C^m$, m−point spectrum of $\boldsymbol{x}_i (m > 8)$

by using a Q-inspired model.

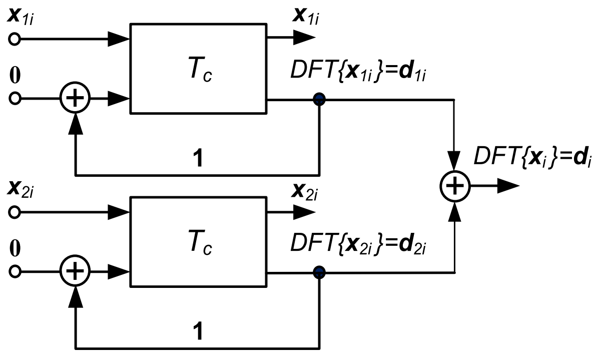


FIGURE 23. Block structure of DFT processor.

Taking m linearly independent vectors, $z_1, \ldots, z_m$, $dimz_i = m$, one obtains the following superposition:

$$\boldsymbol{x}_i = \alpha_1 z_1 + \ldots + \alpha_m z_m, \quad \boldsymbol{x}_i \in S \tag{62}$$

and $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_m]^T = \boldsymbol{Z}^{-1}\boldsymbol{x}_i$

where nonsingular matrix $\boldsymbol{Z} = [z_1, \ldots, z_m]$.

Hence, the structure of the approximator should be supported by m vectors:

$$\boldsymbol{u}_i = \begin{bmatrix} z_i \\ DFT(z_i) \end{bmatrix}; \quad i = 1, \ldots, m, dim\, \boldsymbol{u}_i = 2m \tag{63}$$

and

$$\phi \begin{bmatrix} z_i \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} z_i \\ DFT(z_i) \end{bmatrix}, \quad i = 1, \ldots, m. \tag{64}$$

However, due to the conditions of (28), the number of patterns that must be fulfilled is:

$$N < 0.5\,(2m) = m$$

i.e., in the case considered here, one obtains:

$$m < 0.5\,(2m) = m \text{ (contradiction).}$$

Hence to compute $DFT\,\{x_i\}, i = 1, \ldots, L$, one needs at least two approximators. Indeed, any vector $\boldsymbol{x}_i \in S$ can be expressed as:

$$\boldsymbol{x}_i = \boldsymbol{x}_{1i} + \boldsymbol{x}_{2i}, \quad i = 1, \ldots, L \tag{65}$$

where:

$$\boldsymbol{x}_{1i} = \alpha_1 z_1 + \ldots + \alpha_l z_l, \quad l = m/2$$
$$\boldsymbol{x}_{2i} = \alpha_{l+1} z_{l+1} + \ldots + \alpha_m z_m$$

and

$$DFT\,\{\boldsymbol{x}_i\} = DFT\,\{\boldsymbol{x}_{1i}\} + DFT\,\{\boldsymbol{x}_{2i}\}.$$

Therefore, one obtains the structure of two approximators supported by $m/2$ vectors, respectively:

$$\boldsymbol{u}_i = \begin{bmatrix} z_i \\ DFT(z_i) \end{bmatrix}; \quad i = 1, \ldots, l \tag{66}$$

$$\boldsymbol{u}_i = \begin{bmatrix} z_i \\ DFT(z_i) \end{bmatrix}; \quad i = l+1, \ldots, m, l = m/2. \tag{67}$$

Algorithm 1

```
1. Declaration:
   Input the set of training points:
   $S = \{\boldsymbol{x}_i, \boldsymbol{d}_i\}, i = 1, 2, \ldots, N,$
   $\boldsymbol{x}_i \in C^n, \boldsymbol{d}_i \in C^m, n + m = 2^k, k = 3, 4, \ldots$
2. System design:
   Create system vectors $\boldsymbol{u}_i$ :
   $\boldsymbol{u}_i = \begin{bmatrix} \boldsymbol{x}_i \\ \boldsymbol{d}_i \end{bmatrix}, dim\boldsymbol{u}_i = n + m$
   Calculate the spectrum $\boldsymbol{m}_i$ of system vectors $\boldsymbol{u}_i$ :
   $\boldsymbol{m}_i = \frac{1}{2}\left(\boldsymbol{W}_{2^k} + \boldsymbol{1}\right)\boldsymbol{u}_i$
   Create the spectrum matrix $\boldsymbol{M}$ :
   $\boldsymbol{M} = [\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_N]$
   Calculate the Hermitian matrix $\boldsymbol{W}_H$:
   $\boldsymbol{W}_H = \boldsymbol{M}\left(\boldsymbol{M}^T\boldsymbol{M}\right)^{-1}\boldsymbol{M}^T$
   Calculate the orthogonal transformation $T(\cdot)$:
   $T(\cdot) \equiv \boldsymbol{T} = \frac{1}{2}\left(\boldsymbol{W}_{2^k} + \boldsymbol{1}\right)$
   Calculate the biorthogonal transformation $T_s(\cdot)$:
   $T_s(\cdot) \equiv \boldsymbol{T}_s = \left(2 \cdot \boldsymbol{1} - \boldsymbol{W}_H - \boldsymbol{W}_{2^k}\right)^{-1}$
3. Recursive procedure:
   for i = 1:N
       $\hat{\boldsymbol{d}}_i^{(0)} = \boldsymbol{0}$
       while $\left\|\hat{\boldsymbol{d}}_i^{(l)} - \hat{\boldsymbol{d}}_i^{(l-1)}\right\| \geq eps$
           $\begin{bmatrix} \hat{\boldsymbol{x}}_i \\ \hat{\boldsymbol{d}}_i \end{bmatrix}^{(l)} = \boldsymbol{T}^{-1}\boldsymbol{T}_s\left(\begin{bmatrix} \boldsymbol{x}_i \\ \boldsymbol{0} \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} \\ \hat{\boldsymbol{d}}_i \end{bmatrix}^{(l-1)}\right)$
       end
   end
   ($l = 1, 2, \ldots$ steps of recurrence)
   Final results of recurrence: $\hat{\boldsymbol{d}}_i = \boldsymbol{d}_i; \hat{\boldsymbol{x}}_i = \boldsymbol{x}_i$
```

Thus, the approximator shown in Fig. 23 computes the DFT spectrum $\boldsymbol{d}_i$ for any vector $\boldsymbol{x}_i \in S$ and, moreover, it computes the spectrum $\boldsymbol{d}$ for any vector $\boldsymbol{x} \prec \infty(\dim \boldsymbol{x} = m)$. It is clear that through inverse modeling, the same approximator computes IDFT. Finally, it is worth noting that the Q-inspired model of machine learning described in Statement 2, which was designed as complex-value approximator, can be used as a model of an analog processor (Section 6, C) for the addition and multiplication of complex numbers.

X. CONCLUSION

This study proposed a general model of a machine learning system. This model has the following universal features: it makes it possible to realize the typical, basic functions of learning systems, such as association, pattern recognition and classification, as well as inverse modeling. This study also focused on one aspect of the application of the presented model: its usefulness for the signal reconstruction of incomplete patterns. This aspect of the application was illustrated with the results of the analyses, which showed that image reconstruction is possible even in a case where 90% of information was randomly removed from the signal. This means that the model can be used as a device for data compression as well. Moreover, as a side result of the research, the structure of a superposition processor was proposed. However, research on VLSI realizability, and the practical meaning of such a processor is beyond the scope of this study. Nevertheless, the structures proposed in this study can be seen either as mathematical algorithms, or as models of physically realizable VLSI networks. Thus, unlike deep learning algorithms, which are basically multilayer nonlinear kernel machines designed by very-large-scale optimization tools (e.g., SGD), the approximators proposed in this study, as physical objects, could implement reversible computations and "parallelism." Finally, it should be noted that the Q-inspired networks considered in this study and implemented as complex-valued structures are more powerful than the real-valued realizations. However, the technological realizability of CVNN is currently unknown, and thus, they can be viewed only as mathematical algorithms.

APPENDIX

SUMMARY OF ALGORITHM

REFERENCES

[1] T. Poggio and S. Smale, "The mathematics of learning: Dealing with data," *Notices AMS*, vol. 50, no. 5, 537-544, May 2003.

[2] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *J. Mach. Learn. Res.*, vol. 7, pp. 2399–2434, Nov. 2006.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[4] H. Mhaskar, Q. Liao, and T. Poggio, "Learning functions: When is deep better than shallow," Center Brains, Minds Mach., MIT, Cambridge, MA, USA, CBMM Memo 045, May 2016.

[5] C. C. Aggarwal, *Neural Networks and Deep Learning*. Cham, Switzerland: Springer, 2018.

[6] L. Bottou and O. Bousquet, "The tradeoffs of large scale learning," in *Optimization for Machine Learning*. Cambridge, MA, USA: MIT Press, 2012, pp. 351–368.

[7] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

[8] W. U. Bajwa, V. Cevher, D. Papailiopoulos, and A. Scaglione, "Machine learning from distributed, streaming data," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 11–13, May 2020.

[9] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond Euclidean data," *IEEE Signal Process. Mag.*, vol. 34, no. 4, pp. 18–42, Jul. 2017.

[10] N. H. Nquyen and E. C. Behrman, "Benchmarking neural networks for quantum computation," 2018, *arXiv:1807.03253v3*. [Online]. Available: https://arxiv.org/abs/1807.03253

[11] F. Qi and C. Chen, "Qubit neural tree network with applications in nonlinear system modeling," *IEEE Access*, vol. 6, pp. 51598–51606, 2018.

[12] A. Manju and M. J. Nigam, "Applications of quantum inspired computational intelligence: A survey," *Artif. Intell. Rev.*, vol. 42, no. 1, pp. 79–156, Jun. 2014.

[13] S. Haykin, *Neural Networks and Learning Machines*, 3rd ed. London, U.K.: Pearson, 2009.

[14] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci. USA*, vol. 79, 2554-2558, Apr. 1982.

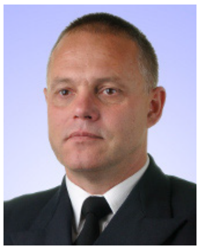
[15] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, pp. 195–202, Sep. 2017.

[16] J. D. Meiss, *Ordinary Differential Systems*. Philadelphia, PA, USA: SIAM, 2017.

[17] K. R. Meyer and G. R. Hall, *Introduction to the Theory of Hamiltonian Dynamical Systems and N-Body Problem*. New York, NY, USA: Springer-Verlag, 1992.

[18] W. Sienko and W. Citko, "Hamiltonian neural networks based networks for learning," in *Machine Learning*, A. Mellouk and A. Chebira, Eds. Austria, Vienna: I-Tech, 2009, pp. 75–92.

[19] W. Sienko and D. Zamojski, "Hamiltonian neural networks based classifiers and mappings," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN)*, Vancouver, BC, Canada, Jul. 2006, pp. 794–798.
[20] B. Eckmann, "Topology, algebra, analysis-relations and missing links," *Notices AMS*, vol. 46, no. 5, pp. 520–527, 1999.
[21] I. T. Jolliffe, *Principal Component Analysis*. New York, NY, USA: Springer-Verlag, 1988.
[22] A. Cichocki, R. Zdunek, and S. Amari, *Nonnegative Matrix and Tensor Factorizations*. Hoboken, NJ, USA: Wiley, 2009.
[23] D. G. Chachlakis, A. Prater-Bennette, and P. P. Markopoulos, "L1-norm tucker tensor decomposition," *IEEE Access*, vol. 7, pp. 178454–178465, 2019.
[24] P. Comon, "Independent component analysis, a new concept?" *Signal Process.*, vol. 36, no. 3, pp. 287–314, Apr. 1994.
[25] A. Luksza and W. Sienko, "Using passive neural networks to solve TSP," in *Proc. IEEE 2nd Int. Conf. Cybern. (CYBCONF)*, Gdynia, Poland, Jun. 2015, pp. 79–84.
[26] S. Boyd, L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[27] X. Fu, K. J. Huang, N. D. Sidiropoulos, and W. K. Ma, "Nonnegative matrix factorization for signal and data analytics: Identifiability, algorithms, and applications," *IEEE Signal Process. Mag.*, vol. 36, no. 2, pp. 59–79, Mar. 2019.
[28] J. Preskill, *Quantum Information and Computation*, in (Lecture Notes in Physics), vol. 229. Scotts Valley, CA, USA: CreateSpace, 2015.
[29] L. Glass and M. C. Mackey, "Pathological conditions resulting from instabilities in physiological control systems*," *Ann. New York Acad. Sci.*, vol. 316, no. 1, pp. 214–235, Feb. 1979.
[30] E. N. Lorenz, "Deterministic nonperiodic flow," *J. Atmos. Sci.*, vol. 20, no. 2, pp. 130–141, Mar. 1963.
[31] L. O. Chua and G.-N. Lin, "Canonical realization of Chua's circuit family," *IEEE Trans. Circuits Syst.*, vol. 37, no. 7, pp. 885–902, Jul. 1990.
[32] O. E. Rössler, "An equation for continuous chaos," *Phys. Lett. A*, vol. 57, no. 5, pp. 397–398, Jul. 1976.



WIESLAW CITKO received the M.S. degree in solid state physics from the Faculty of Applied Physics and Mathematics, Gdansk University of Technology, and the Ph.D. degree in electronics from the Lodz University of Technology. He is currently an Assistant Professor with the Department Electrical Engineering, Gdynia Maritime University, Gdynia, Poland. His current research interests include artificial intelligence, machine learning, neural networks, and PLL networks.

WIESLAW SIENKO (Life Member, IEEE) received the M.S. and Ph.D. degrees in electronics from the Gdansk University of Technology. He is currently a Professor with the Department of Electrical Engineering, Gdynia Maritime University, Gdynia, Poland. His current research interests include artificial intelligence, machine learning, quantum signal processing, and digital signal processing.