Convolutional hyper basis function neural network and its online learning in the image recognition task

Yevgeniy Bodyanskiy[a], Anastasiia Deineko[b], Oleksandr Hontsa[b] and Oleksandr Zeleniy[c]

[a] *Kharkiv National University of Radio Electronics, Control Systems Research Laboratory, Nauky av., 14, Kharkiv, 61166, Ukraine*
[b] *Kharkiv National University of Radio Electronics, Artificial Intelligence Department, Nauky av., 14, Kharkiv, 61166, Ukraine*
[c] *Kharkiv National University of Radio Electronics, Department of Media Systems and Technologies, Nauky av., 14, Kharkiv, 61166, Ukraine*

Abstract

In the article hyper basic function neural network (HBFN) in convolutional neural networks instead of fully connected perceptron layers, which solve the problem of classification is proposed. HBFNs are generalization of traditional radial-basic function networks, which like multilayer perceptrons are also universal approximators. The peculiarity of hyper basic function neural networks is that their receptive fields are hyperellipsoids with arbitrary orientation of the axes in the feature space. It is assumed that, along with the synaptic weights, receptive field parameters are adjusted: their centers and matrices of the axis orientation. This approach allows to reduce the total number of synaptic weights, and accordingly the number of R-neurons (activation functions) of the network. Using multidimensional V. Epanechnikov's functions with a hyperellipsoidal receptive field that can be adjusted during the learning process as activation functions permit to accelerate the learning process of investigated network. It should be noted that applying kernel functions in convolutional neural networks permits to avoid the undesirable effect of "exploiding gradient", which often occurs in fully connected layers. The computational experiment results on standard data sets confirm the effectiveness of the proposed approach and, first of all, the high learning speed.

Keywords

Convolutional neural network, deep learning, image recognition, radial-basis function network, hyper-basis function neural network, kernel activation function, multiclass classification

1. Introduction

For today the deep neural networks (DNN) are widely used for solving large class of tasks that are connected with Data Mining and especially Big Data Mining, first of all, pattern recognition-classification, forecasting, identification-emulation, natural language processing, etc. [1-4]. Here a special place the convolutional neural networks (CNN) have been occupied, that are designed to solve various problems connected with images of different natures processing. Usually, every CNN has multilayer feedforward architecture that can be divided onto two separate independent parts.

The first part is – autoencoder that is used for input signal-image compression, that is commonly given in the matrix form. As a result of the input 2D-signal processing by the convolutional and

IntelITSIS'2021: 2nd International Workshop on Intelligent Information Technologies and Systems of Information Security, March 24–26, 2021, Khmelnytskyi, Ukraine
EMAIL: yevgeniy.bodyanskiy@nure.ua (Yevgeniy Bodyanskiy); anastasiya.deineko@gmail.com (Anastasiia Deineko); aphontsa@gmail.com (Oleksandr Hontsa); oleksandr.zeleniy@nure.ua (Oleksandr Zeleniy)
ORCID: 0000-0001-5418-2143
View this author's ORCID profile (Yevgeniy Bodyanskiy)
© 2021 Copyright for this paper by its authors.
Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).
CEUR Workshop Proceedings (CEUR-WS.org)

CEUR-WS.org/Vol-2853/paper8.pdf

pooling layers in the output of the last pooling layer at this part of the system n-dimensional vector signal is formed. In fact, if the input signal-image has form of the $(n_1 \times n_2)$-matrix, then usually $n \ll n_1 n_2$, i. e. this part of the CNN realizes compression task of the input matrix signal. The second part of the CNN in general case solves classification task usually, it is standard multilayer perceptron (MLP) whose nodes are F. Rosenblatt's elementary perceptrons, with piecewise activation functions. Here it is necessary to note, that in practical implementation of the shallow neural networks (SNN) so-called
sigmoidal squashing functions [5] as activation function are used, moreover, to provide universal approximation properties usually only three fully connected layers are enough [6, 7].

At the same time, learning of the standard MLP with squashing activation function is met with extremely unpleasant effect of the vanishing gradient, that significantly complicates the process of synaptic weights adjustment based on the error back propagation method. That is why, in the DNN instead of classic sigmoids or hyperbolic tangents linear piecewise functions are used, that provide desired approximation quality [8], but at the same time require of the significant increasing of the network's layers, and accordingly increases the number of tuned synaptic weights. Moreover, the transfer learning does not always been a way to overcome this problem.

The simplest way to overcome this problem is to implement instead of the standard MLP in the CNN, so-called, radial-basis function neural networks (RBFN), that include only one layer of the tuned synaptic weights and whose output signals are linearly depended on these weights, that allows to use for learning fairly simple optimization algorithms [9-12]. Such approach was implemented in the [13], there from the commonly known DNN were deleted perceptron's layers and instead of them to the outputs of the last pooling layer standard RBFN with *n* input and *m* inputs was connected where m determines number of the classes into which should be divided investigated data set.

However, the RBFN implementation instead of the MLP also does not solves all arising problems because, firstly, conventional radial-basis function neural networks suffer from the so-called "curse of dimensionality", when amount of the activation function in the R-neurons are exponentially increased dependently from the input space dimension, and, secondly, arises the problem of hyperspherical receptive fields centers distribution of the kernel activation functions usually as multidimensional Gaussians or Cauchyans type. The problem of the centers distribution usually is solved by using of one or another clustering algorithms (in [13], for example, standard K-means algorithm was used) that is connected with essential computational problems.

An alternative to traditional RBFN, in our opinion, can be so-called hyper basis neural networks (HBFN) [14], which in contrast to the symmetric kernel functions of RBFN have hyperellipsoidal receptive fields with arbitrary orientation of the axes in network input space. And if to realize online tuning of all hyperellipsoids parameters including its centers, using HBFN allows to refuse from standard clustering procedures for finding of these centers.

2. The hyper-basis function neural network and its online learning

The prototype of hyper basis function network (HBFN) is traditional radial-basis function network [15] that consist of only one hidden layer. This hidden layer is formed by so-called R-neurons which realize nonlinear transformation of the input signals with so-called radial-basis, bell-shaped, potential function which are based on the hypersphere of a fixed radius. In the Figure 1 architecture of this neural network is shown. Here in this network is assumed that coordinates of hyperspherical receptive fields centers are fixed and do not change during learning process. HBFN is generalization of the standard RBFN in which hyperellipsoids with arbitrary axis orientations are used instead of spherical receptive fields. Thus, the centers coordinate, and orientation of the receptive fields parameters are set apriori and do not specified in the learning process. It is clear that significantly expand the RBFN could by R-neurons learning in one- or another-way i. e. perform learning in the second hidden layer.

To the inputs of the hyper basis function neural network from the last pooling layer of the convolutional network the vector signals sequence $x(k) = (x_1(k), x_2(k), \ldots, x_i(k), \ldots, x_n(k))^T \in R^n$ is fed to processing, where $k = 1,2, \ldots, N -$ is an observations number or discrete time of the training set $X = \{x(1), x(2), \ldots, x(k), \ldots, x(N)\} \subset R^n$. This signal is passed on the R-neurons inputs, formed by kernel activation functions

$$\psi_l(x(k)) = \psi_l\left(\|x(k) - c_l\|^2_{\Sigma_l^{-1}}\right), l = 1,2,\dots,h$$

where $c_l -$ the receptive field center of the function $\psi_l(\cdot)$, $\Sigma_l^{-1} -$ positive $(n \times n)$ −matrix of the receptive field orientation, $\psi_0(x(k)) = 1$. Usually, it can be multidimensional Gaussian

$$\psi_l(x(k)) = exp\left(-\|x(k) - c_l\|^2_{\Sigma_l^{-1}}\right), \quad (1)$$

Cauchyan

$$\psi_l(x(k)) = exp\left(1 + \|x(k) - c_l\|^2_{\Sigma_l^{-1}}\right)^{-1}, \quad (2)$$

and other bell-shaped constructions.

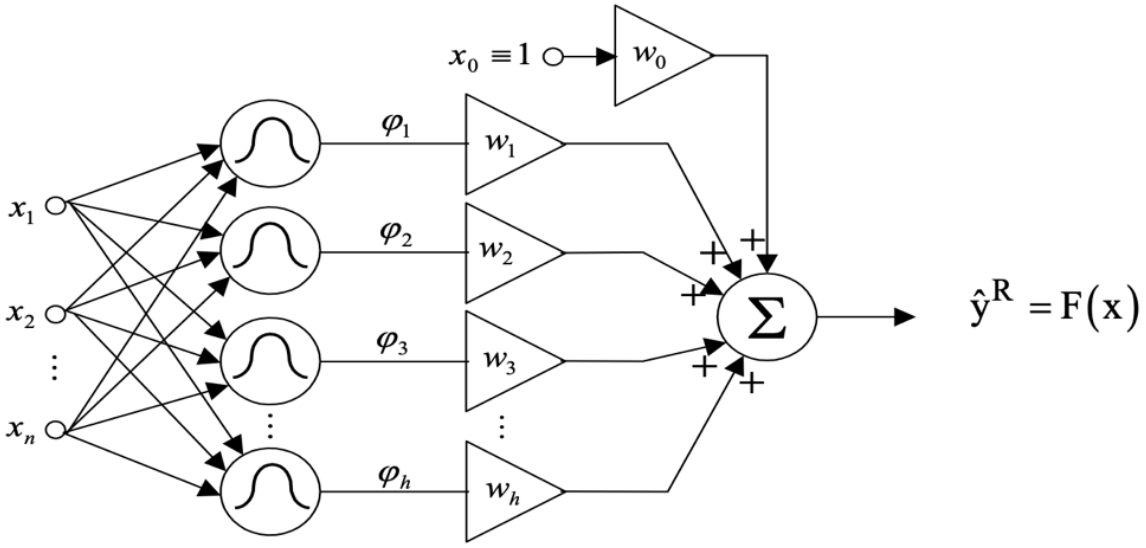

Figure 1: Standard radial-basis function network

Visualization of the Gaussian with different width parameters is shown in the Figure 2.

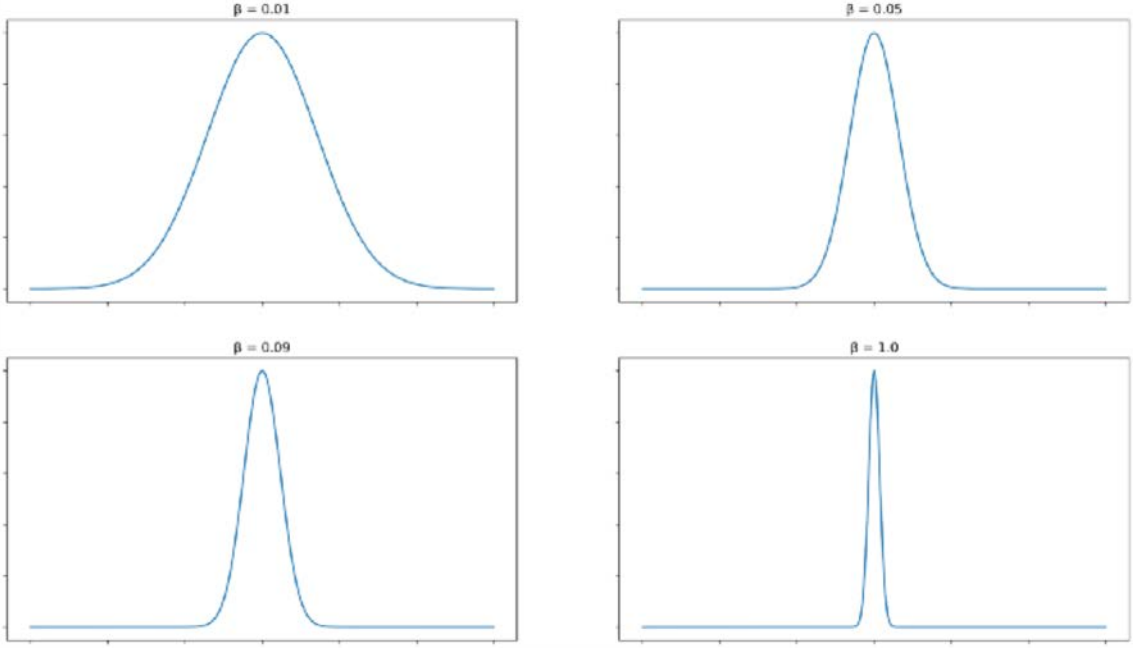

Figure 2: Standard Gaussian with different width parameters

The outputs of the all R-neurons are fed to the inputs of the *m* adaptive linear associators (ALA), that in fact are essentially ordinary adders with adjusteable synaptic weights at the inputs. Thus, at the outputs of the adders the signals are appeared in the form

$$\hat{y}_j(k) = w_{j0} + \sum_{l=1}^{h} w_{jl}\psi_l(x(k)) = \sum_{l=0}^{h} w_{jl}\psi_l(x(k)) = w_j{}^T\tilde{\psi}(x(k)) \quad (3)$$

where $\tilde{\psi}(x(k)) = (1, \psi^T(x(k)))^T$, $\psi(x(k)) = \left(\psi_1(x(k)), \ldots, \psi_l(x(k)), \ldots, \psi_h(x(k))\right)^T$, $j = 1,2,\ldots,m$, $h > n -$ determines the amount of the hyper basis functions, that are included in the network.

In the case when the quadratic error is used as a learning criterion, the signals (3) are the outputs of the all network, if the cross-entropy criterion and reference signal one hot coding is used, at the outputs of the system the every observation membership level for the all possible classes $j = 1,2,\ldots,m$ are additionally appear.

Next it is easy to write the gradient procedure of the RBFN's synaptic weights tuning:

$$w_{jl}(k+1) = w_{jl}(k) + \eta_w(k+1)\left(y_j(k+1) - w_j{}^T(k)\tilde{\psi}(x(k+1), c_l(k), \Sigma_l^{-1}(k))\right) \times \times \psi_l\left(\|x(k+1) - c_l(k)\|^2_{\Sigma_l^{-1}(k)}\right) \quad (4)$$

(here $w_k{}^T(k) = \left(w_{j0}(k), w_{j1}(k), \ldots, w_{jh}(k)\right)^T$, $\eta_w(k+1) -$ the learning rate parameter of the synaptic weights), kernel functions centers [15]:

$$c_l(k+1) = c_l(k) - \eta_c(k+1)\left(y_j(k+1) - w_j{}^T(k+1)\tilde{\psi}(x(k+1), c_l(k), \Sigma_l^{-1}(k))\right) \times \times \psi'\left(\|x(k+1) - c_l(k)\|^2_{\Sigma_l^{-1}(k)}\right)\Sigma_l^{-1}(k)(x(k+1) - c_l(k)), \quad (5)$$

(here $\eta_c(k+1) -$ the learning rate parameter of the activation function centers) and matrix of the receptive hyperelipsoids orientation [16]

$$\Sigma_l^{-1}(k+1) = \Sigma_l^{-1}(k) + \eta_\Sigma(k+1) \times \times \left(y_j(k+1) - w_j{}^T(k+1)\tilde{\psi}\left((k+1), c_l(k+1), \Sigma_l^{-1}(k)\right)\right) \times \times \psi\left(\|x(k+1) - c_l(k+1)\|^2_{\Sigma_l^{-1}}\right)(x(k+1) - c_l(k+1))(x(k+1) - c_l(k+1))^T \quad (6)$$

(here $\eta_\Sigma(k+1) -$ the learning rate of the receptive hyperellipsoids tuning).

The learning algorithms (4)-(6) are enough simple form the computational point of view, but they are not protected from the possible effect of the "exploding gradient" in the case when the derivatives of the function type (1), (2) acquire small values.

3. The HBFN learning based on the quadratic activation function

In the [13] as an activation functions of the radial-basis neural network (RBFN) using the quadratic functions was proposed

$$\psi_l(x(k)) = 1 - \frac{\|x(k) - c_l\|^2}{\sigma_l^2} \quad (7)$$

that in fact are kernel V. Epanechnikov's function [17], where parameter σ_l^2 determines hyperspherical receptive field radius of this function.

In this paper the modified V. Epanechnikov's function with hyperellipsoidal receptive field with arbitrary orientation is proposed to use instead of (7):

$$\psi_l(x(k)) = 1 - \|x(k+1) - c_l\|^2_{\Sigma^{-1}} \quad (8)$$

whose derivatives acquire zero value only at one point, where $x(k+1) = c_l$ and automatically does not suffer from the "exploding gradient" effect. In this situation the gradient learning procedure (4)-(6) takes very simple form:

$$w_{jl}(k+1) = w_{jl}(k) + \eta_w(k+1) \times \times \left(y_j(k+1) - w_j{}^T(k)\tilde{\psi}\left(x(k+1), c_l(k), \Sigma_l^{-1}(k)\right)\right) \times \times \left(1 - \|x(k+1) - c_l(k)\|^2_{\Sigma_l^{-1}(k)}\right), \quad (9)$$

$$c_l(k+1) = c_l(k) - \eta_c(k+1) \times \times \left(y_j(k+1) - w_j{}^T(k+1)\tilde{\psi}\left(x(k+1), c_l(k), \Sigma_l^{-1}(k)\right)\right) \times \quad (10)$$

$$\times w_{jl}(k+1)\Sigma_l^{-1}(k)\big(x(k+1)-c_l(k)\big),$$
$$\Sigma_l^{-1}(k+1)=\Sigma_l^{-1}(k)-\eta_c(k+1)\times$$
$$\times\left(y_j(k+1)-w_j{}^T(k+1)\tilde{\psi}\left(x(k+1),c_l(k+1),\Sigma_l^{-1}(k)\right)\right)\times \quad (11)$$
$$\times w_{jl}(k+1)\big(x(k+1)-c_l(k+1)\big)\big(x(k+1)-c_l(k+1)\big)^T$$

that essentially simplifies its numerical implementation.

4. The computational experiments results

For numerical analysis of the approach under consideration, the convolutional neural network VGG-16 in which the fully connected layers of the multilayer perceptron were replaced by a hyper basic function neural network with a different number of R-neurons and, accordingly, tuned synaptic weights was used as a prototype. Architecture of the proposed convolutional hyper basis function network is shown in the Figure 3.

For the program realization of the proposed convolutional hyper basis function network model programming environment "Python 3.8" and “Keras” library were used [18, 19].

For solving binary classification task data set “Dogs&Cats” from “Kaggle” was used. This data set consists of the 25000 colorful dogs and cats images. The output set was divided into tree subsets: the training data set – 17500 images, test data set – 3750 images and validation one – 3750 images.

For the multiclass classification data set “CIFAR-10”, that consists of the 60000 colorful images 32×32 from different classes: planes, cars, birds, cats, dogs and others with 6000 images in every class. At the same time training subset consists of the 45000 samples, test subset – 5000 samples and validation one 10000 samples.

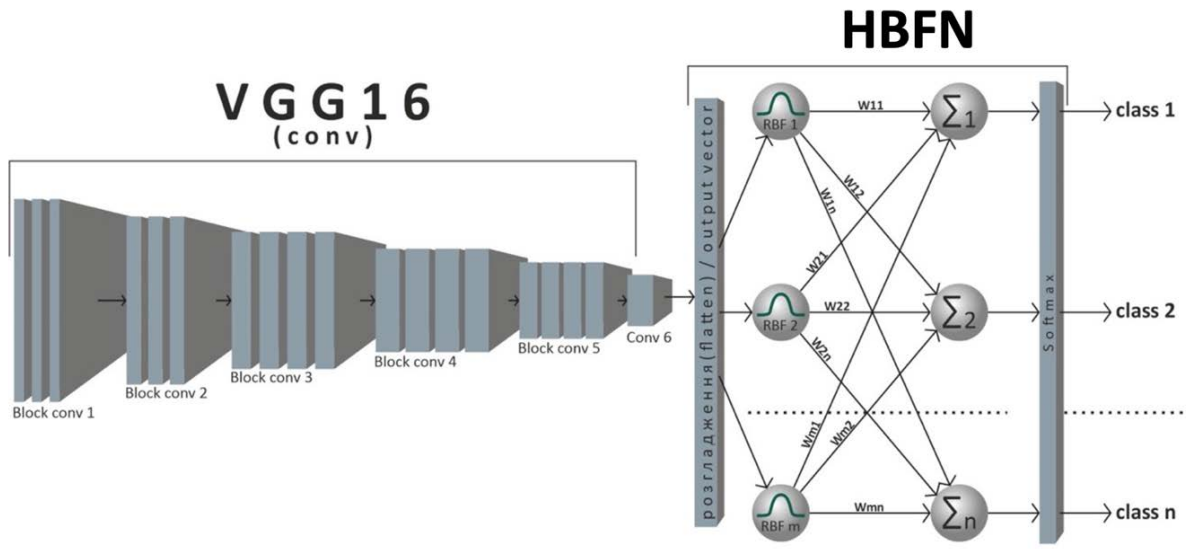

Figure 3: Convolutional hyper basis function network is shown at the Figure 1.

In the learning process for solving binary classification task designed hyper basis function neural networks were tuned by five epochs. The results of this process are shown in the Figure 4. The results of the multiclass classification are shown in the Figure 5 and for network learning in this situation 50 epoch were used.

```
Train Accuracy:0.910
Test Accuracy:0.913
Wall time: 2min 26s
```

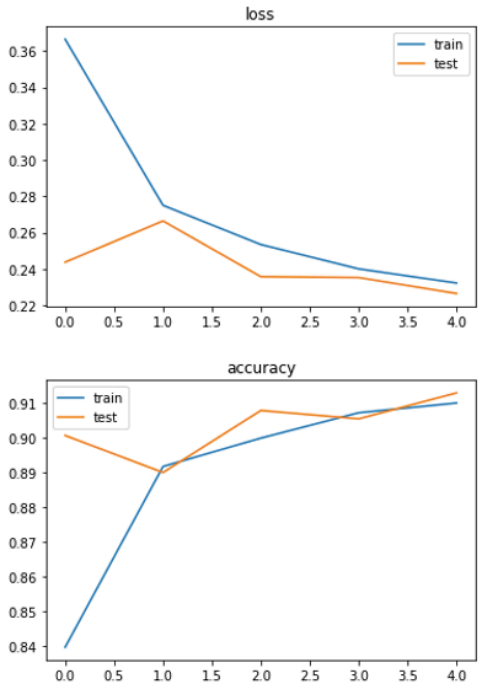

Figure 4: Results of the network learning for binary classification

```
Train Accuracy:0.558
Test Accuracy:0.595
Test Loss:16.632
Wall time: 21.1 s
```

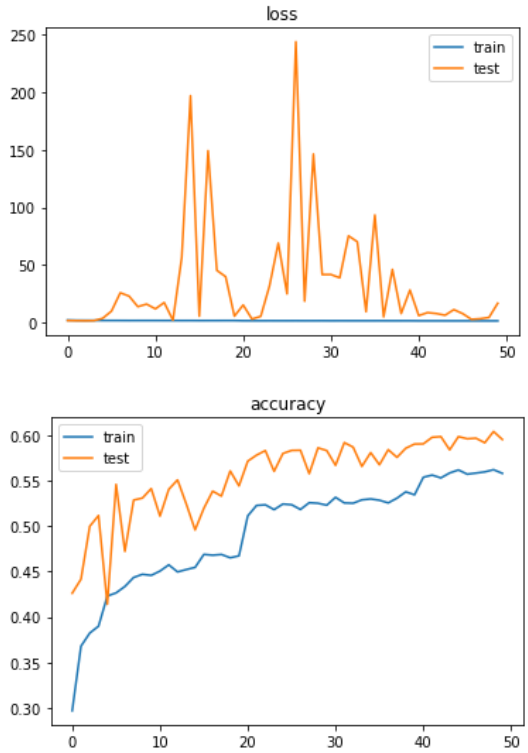

Figure 5: The network learning results for multiclass classification

To improve classification accuracy the learning rate parameters were significantly reduced and assumed as a constant value at the level $\eta(k) = 10^{-6}$ for all tuned variables. The results of this learning process are demonstrated in the Figure 6 and Figure 7. Summarized numerical results of HBFN training are shown in the Table 1.

Table 1
Numerical results of the HBFN training

	Accuracy results	
Training stage	Binary classification	Multiclass classification
First training stage	89.71 %	58.56 %
Second training stage	92.99 %	66.65%

The quality of the classification results significantly depends on the R-neurons number in the network. In the Table 2 the binary classification results are demonstrated, at the same time in the second column the synaptic weights tuning results are shown and in the third column – the tuning results of the synaptic weights and receptive fields are presented.

The training results of convolutional neural network with hyper basis layer is shown in the Figure 8. Average time for learning model of the proposed convolutional hyper basis neural network was one hour and 9 minutes.

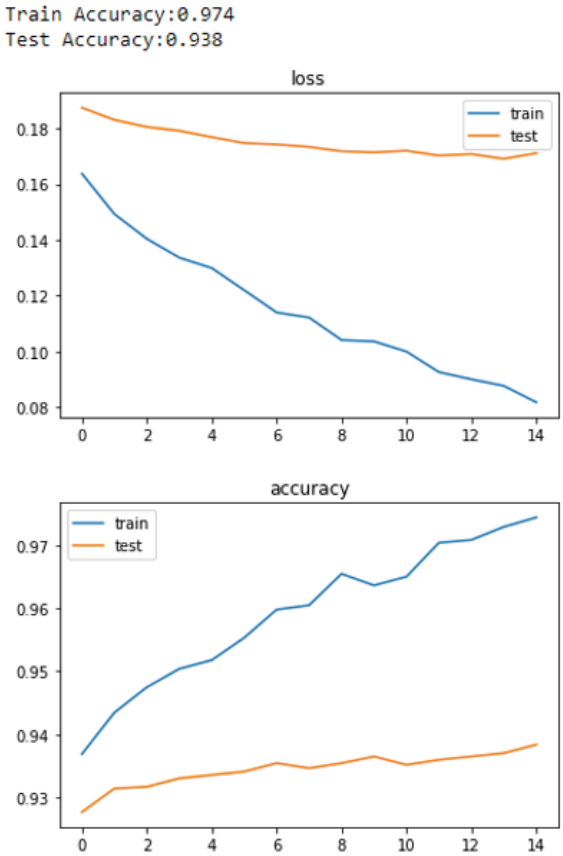

Figure 6: The results of the second stage of training for binary classification network

```
Train Accuracy:0.639
Test Accuracy:0.650
Test Loss:15.335
Wall time: 42.2 s
```

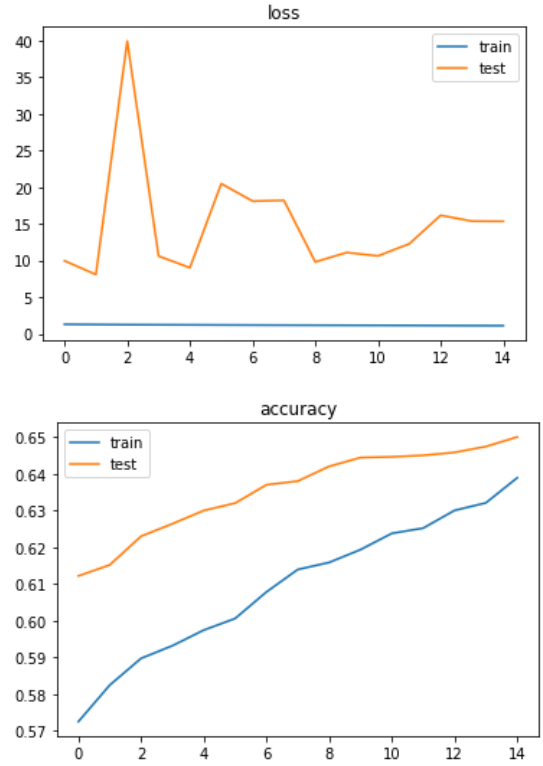

Figure 7: The results of the second stage of network training the multiclass classification

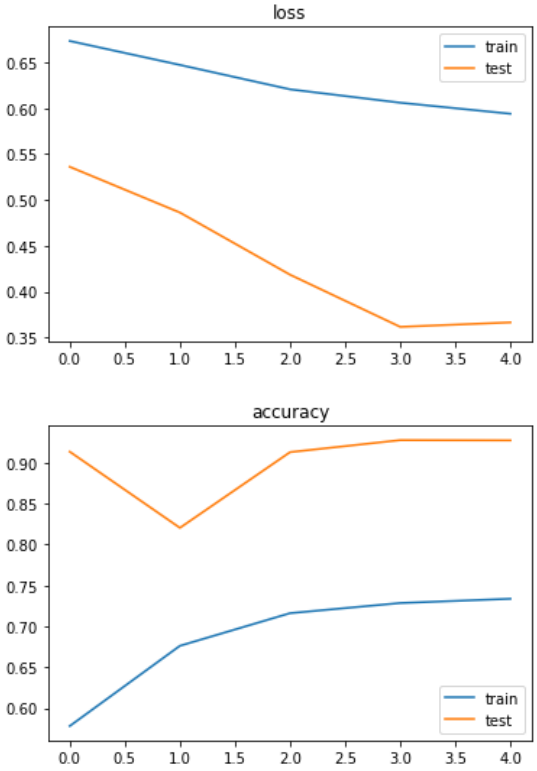

Figure 8: The results of training convolutional neural network with hyper basis layer.

Table 2
The results of the binary classification

Numbers of R-neurons	Only synaptic weights tuning	Tuning of the synaptic weights and receptive fields
2048	93.94%	94.07%
1024	92.83%	93.31%
516	88.43%	92.88%
256	87.07%	92.37%
128	87.12%	92.35%
64	84.96%	90.48%
32	80.34%	92.55%
16	72.72%	92.56%

Table 3 shows the results of multiclass classification.

Table 3
The results of the multiclass classification

Numbers of R-neurons	Only synaptic weights tuning	Tuning of the synaptic weights and receptive fields
4096	67.27%	67.77%
2048	67.11%	68.12%
1024	66.52%	67.32%
512	66.57%	67.35%
256	67.02%	67.20%
128	66.66%	66.94%
64	65.92%	66.70%
32	64.10%	66.62%

Final comparison results of binary and multiclass classification are shown in the Table 4.

Table 4
Comparison results of binary and multiclass classification

Comparison parameters	Binary classification		Multiclass classification		
	Standard CNN	CNN + HBFN	Standard CNN	CNN + HBFN	
Model accuracy	92.99%	92.35%	66.65%	67.02%	66.62%
number of parameters	24 449	24 449	68 362	68 362	8 549

Here is interesting to note that if the less is number of R-neurons amount in the HRFN then more important is tuning of the receptive fields parameters, then more is speed – then more is velocity of process tuning.

5. Conclusions

The convolutional hyper basis function neural network was proposed. This CNN differs from another because instead of the fully connected perceptron layers that solve classification task, it contains in its architecture HBFN that is generalization of conventional RBFN but besides synaptic weights it can tune too parameters of the receptive fields that are in common case hyperellipsoids with arbitrary axes orientation in feature space. This approach permits automatically to solve the problem of activations kernel functions centers distribution, to reduce their amount i. e. to increase the learning

speed that is very important essentially for deep neural network. The computational experiments confirm effectiveness of the approach under consideration.

6. References

[1] Y. LeCun, Y. Bengio, G. Hinton, Deep learning. Nature 521 (2015) 436–444, doi:10.1038.

[2] J. Schmidhuber, ”Deep Learning in Neural Networks: An Overview”, Neural Networks, 61, (2015), 85-117, doi:10.1016/j.neunet.2014.09.003.

[3] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, The MIT Press, 2016, doi:10.1007/s10710-017-9314-z.

[4] D. Graupe, Deep Learning Neural Networks: Design and Case Studies, New Jersey: World Scientific, 2016.

[5] G. Cybenko, ”Approximation by superposition of a sigmoidal function”, Math. Control Signals Systems 2, (1989), 303-314, doi:10.1007/BF02551274.

[6] K. Hornik, M. Stinchkombe, H. White, “Multilayer feedforward networks are universal approximators”, Neural Networks, 2 (1989), 359-366.

[7] K. Hornik, ”Approximation capabilities of multilayer feedforward networks”, Neural Networks, 4 2 (1994), 251-257, doi:10.1016/0893-6080(91)90009-T.

[8] Ch. Huang, “ReLU Networks Are Universal Approximators via Piecewise Linear or Constant Functions”, Neural Computation, vol. 30, (2020), 1-30, doi:10.1162/neco_a_01316.

[9] J. Moody, C. J. Darken, “Fast learning in networks of locally tuned processing units”, Neural Computation 1 (1989), 281-294, doi:10.1162/neco.1989.1.2.281.

[10] T. Poggio, F. Girosi, Network for approximation and learning, Proc. IEEE, 79 (9), 1990, pp. 1481-1497, doi:10.1002/ecjc.4430760808.

[11] J. Park, I. W. Sandberg, “Universal approximation using radial-basis function network,” Neural Computation, 3 (1991), 246-257, doi:10.1007/978-3-7908-1826-0_1.

[12] J. A. Leonard, M. A. Kramer, L. H. Ungar, Using radial-basis function to approximate a function and its error bouns, IEEE Trans. on Neural Networks, 3, 1992, pp. 594-603, doi: 10.1109/72.143377.

[13] M. Amirian, F. Schwenker, Radial basis function networks to learn similarity distance metric and improve interpretability, IEEE Access, 8, 2020, pp. 123087-123097, doi: 10.1109/ACCESS.2020.3007337.

[14] Y. Zhou, T. Mu, Zh-H. Pang, Ch. Zheng, “A survey on hyper basis function neural network”, System Science & Control Engineering, 7 1 (2019), 495-507, doi:10.1080/21642583.2019.1699474.

[15] Ye. Bodyanskiy, O. Tichenko, A. Deineko, An evolving radial basis neural network with adaptive learning of its parameters and architecture, Automatic Control and Computer Science, 49 5 (2015), 255-260.

[16] R. Tkachenko, P. Tkachenko, I. Izonin, V. Vitynskyi, N. Kryvinska, Y. Tsymbal, Comittee of the combined RBF-SGTM neural-like structures for prediction tasks, Lecture Notes in Computer Science, 16th International conference on mobile web and intelligent information systems, MobiWIS 2019, Istanbul, Turkey, August 26–28, vol. 11673, 2019, pp. 267–277

[17] V. A. Epanechnikov, Nonparametric estimation of multivariate probability density, Probability Theory and its Application, 14 2 (1968), pp. 156-161.

[18] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016. pp. 2818-2826.

[19] K. He, X. Zhang, S. Ren, J. Sun, “Deep residual learning for image recognition", Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2016. 836 p, doi:10.1364/OE.395866.