UDK 004.9

O.M. Matviykiv, O.I. Faitas
Lviv Polytechnic National University,
Computer-Aided Design Department

DATA CLASSIFICATION OF SPECTRUM ANALYSIS USING NEURAL NETWORK

This article provides the comparison of libraries neural networks. Based on this analysis was determined to develop a neural network for classification of spectra based on Encog library, because it implemented many components and gives the best result with a small number of items for training. Showed the architecture of neural networks for data classification of spectral analysis.

Key words: neural network, ANN, Encog, Joone, FANN, Neuroph, Java, C#, LMA, Resilient Propagation, compare, spectroscopy, spectrum, classification.

Наведено аналіз бібліотек нейронних мереж. На основі аналізу запропоновано розробляти нейронну мережу для класифікації спектрів на основі бібліотеки Encog, оскільки вона реалізовує безліч компонентів та дає найкращий результат з невеликою кількістю даних для навчання. Подано архітектуру нейронної мережі для класифікації даних спектрального аналізу.

Ключові слова: нейронна мережа, ANN, Encog, Joone, FANN, Neuroph, Java, C#, Левенберга–Марквардта, еластичне поширення, порівняння, спектроскопія, спектр, класифікація.

Introduction

The problem of classification is one of the main tasks of neural networks. The ability to present a network abstraction allows multiple distorted versions of the image input, the network itself can create output perfect image, even if it never met him. With proper training, the neural network is insensitive to small changes in input signals (eg noise) and gives the desired result at the output.

Application of neural networks has the following advantages [1]:

- problem solving for unknown pre-dependencies between input and output parameters,
- possibility of constructing nonlinear dependencies,
- possibility of applying for a wide range of tasks,
- ability to simultaneously solve several problems a neural network with multiple outputs,
- relative simplicity of the algorithms used,
- ease paralleling for solving the problem.

The main criterion for classification data from spectral analysis is the result of the ratio of classification result and volume data for training the network.

Simplified problem of spectrum identification is XOR operator:

[0,0] -> [0]
[1,0] -> [1]
[0,1] -> [1]
[1,1] -> [0]

This problem will be the basis for comparison of next neural networks libraries:

- Encog v3.1.0;
- Joone v2.0.0RC1;
- Neuroph v2.6;
- FANN v2.2.0.

To solve this problem requires multilayer neural network with two input neurons, six neurons in hidden layer and one output neurons, which corresponds to XOR operator. You can also use a network with two output neurons. In this case, the first entry will match result – 0, and the second – 1, and the value of the neuron – probability. In which of the output neuron is probability to be higher – the result of XOR operator. In the case of one output neuron is the result of XOR operator.

We will train the neural network as long as the error is less than 1 %. The lower amount of data for training is the better network. We used sigmoid activation function.

Popularity signoid function cause such its properties [2]:

• ability to amplify weak signals stronger than large, and resist "saturation" of powerful signals;

• Monotone and differentiation across the x-axis;

• provides an opportunity to use a wide range of optimization algorithms.

As a method of neural network training method used backpropagation. In [3] and [4] shows the use of the method of back-propagation, which gave an opportunity to show the accuracy of greater than 99 %.

Comparing of libraries neural networks are given in [5] and [6], but the following comparison of new expanded library and used their newer versions that have affected the results.

Neural network frameworks

Encog

Encog – is an advanced neural network and machine learning framework. Encog contains classes to create a wide variety of networks, as well as support classes to normalize and process data for these neural networks. Encog trains using multithreaded resilient propagation. Encog can also make use of a GPU to further speed processing time. Encog is available for Java,.Net and Silverlight [7].

Encog release Levenberg Marquardt (LMA) training technique. LMA is one of the most advanced training techniques available for neural networks. It does not work well with all training sets, but it can learn the XOR in a fraction of the time other training techniques require. LMA requires that the output of the neural network be a single neuron [8].

In the spectrum classification is necessary to have a large sample for training to get better results, so we used elastic distribution Resilient Propagation.

Table 1

The result of the neural network for XOR operator

	Training technique	
	Levenberg Marquardt	Resilient Propagation
Eproch	6	46
Error	0,009732929583710	0,00953490779616097
{0.0}->{0}	0,028887323286645	0,071271268850058
{1.0}->{1}	0,981453451136381	0,913933801069238
{0.1}->{1}	0,946280301657144	0,892294065748732
{1.1}->{0}	0,124103187936315	0,107168017702071

In Table 1 presents the results of a neural network for XOR operator using two described above methods of training. As seen from the results, Levenberg Marquardt method showed the better result for Resilient Propagation.

Joone

JOONE – consists of a component-based architecture based on linkable components that can be extended to build new learning algorithms and neural networks architectures. Beyond simulation, Joone also has to some extent multi-platform deployment capabilities. Joone has a distributed training environment that allows for neural networks to be trained on multiple remote machines [9].

Joone is designed so that it is difficult to configure it for better performance of neural network, contains extra parameters to be controlled, required to switch between learning of neural network and its usage.

The results of the neural network for XOR operator with Resilient Propagation:

Epoch: 1 Error: 0,5346437249501386

Epoch: 10358 Error: 0,0100001560936761

After 10,358 iterations, the neural network unable to reach an error less than 1 %.

Neuroph

Neuroph – is lightweight Java neural network framework to develop common neural network architectures. It contains well designed, open source Java library with small number of basic classes which correspond to basic NN concepts [10].

To build a neural network based Neuroph we need a library Encog that complicates the process of building the network. Unlike other frameworks here we will use the TANH activation function.

The results of the neural network for XOR operator with TANH activation function:

Epoch 624, Error = 0,009995507272238732

0,0 actual = 0,0133380765596412, ideal = 0

0,1 actual = 0,8950272722644724, ideal = 1

1,0 actual = 0,9112580115122504, ideal = 1

1,1 actual = 0,0153857736185622, ideal = 0

Neuroph need 624 iterations to achieve error less than 1 %, which is better than Joone, but worse than Encog.

FANN

FANN – is a free open source neural network library, which implements multilayer artificial neural networks in C with support for both fully connected and sparsely connected networks. Cross-platform execution in both fixed and floating point is supported. It includes a framework for easy handling of training data sets. FANN is available for Java,.Net, Python and other [11].

The results of the neural network for XOR operator with Resilient Propagation::

Epoch 1. Error: 0,2825847

Epoch 20. Error: 0,0010269

XOR (0, 0) -> 0,00325358138134, should be 0

XOR (0, 1) -> 0,99557708421431, should be 1

XOR (1, 0) -> 0,99623926833288, should be 1

XOR (1, 1) -> 0,00985109952545, should be 0

Results of the network based on FANN is one of the best and FANN is easy to use. However, in this library implemented only components for multilayer neural network structure, which limits its use.

Determination of the neurons number in the hidden layer

Neural network with a large number of layers is a deep schemes and training of such a network is challenging. Until recently, empirical studies have shown that deep networks generally performed no better and often worse than the neural network with one or two hidden layers [12]. Based on this assertion, for data classification of spectral analysis will use a network with one hidden layer.

Several empirical rules for determining the number of neurons in the hidden layers [13]:

– "A rule of thumb is for the size of this [hidden] layer to be somewhere between the input layer size... and the output layer size..." (Blum, 1992, p. 60);

– "To calculate the number of hidden nodes we use a general rule of: (Number of inputs + outputs) * (2/3)" (from the FAQ for a commercial neural network software company);

– "you will never require more than twice the number of hidden units as you have inputs" in an MLP with one hidden layer (Swingler, 1996, p. 53). See the section in Part 4 of the FAQ on The Worst books for the source of this myth.);

– "How large should the hidden layer be? One rule of thumb is that it should never be more than twice as large as the input layer." (Berry and Linoff, 1997, p. 323);

– "Typically, we specify as many hidden nodes as dimensions [principal components] needed to capture 70-90 % of the variance of the input data set." (Boger and Guterman, 1997)

For a start, these rules have been applied to our XOR operator.

Table 2

The dependence of the error of the number iterations and the number of neurons in the hidden layer

Number of neurons in hidden layer	Epoch	Error, %
1	1000	24,77
2	103	0,99
	232	0,0997
	669	0,00999
	1000	0,00463
6	63	0,967
	174	0,0995
	425	0,00998
	1000	0,00189
12	41	0,995
	135	0,099
	310	0,00993
	765	0,00099
18	8	0.84
	10	0,092
	24	0,0090
	68	0,00099

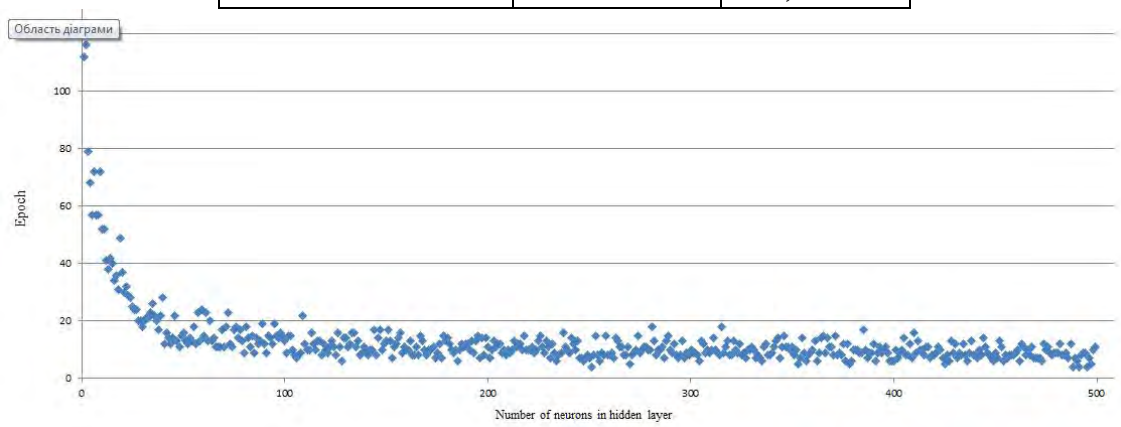


Fig. 1. Dependence of neural inner layer and iterations to achieve an error less than 1 % for a network with 2 input neurons

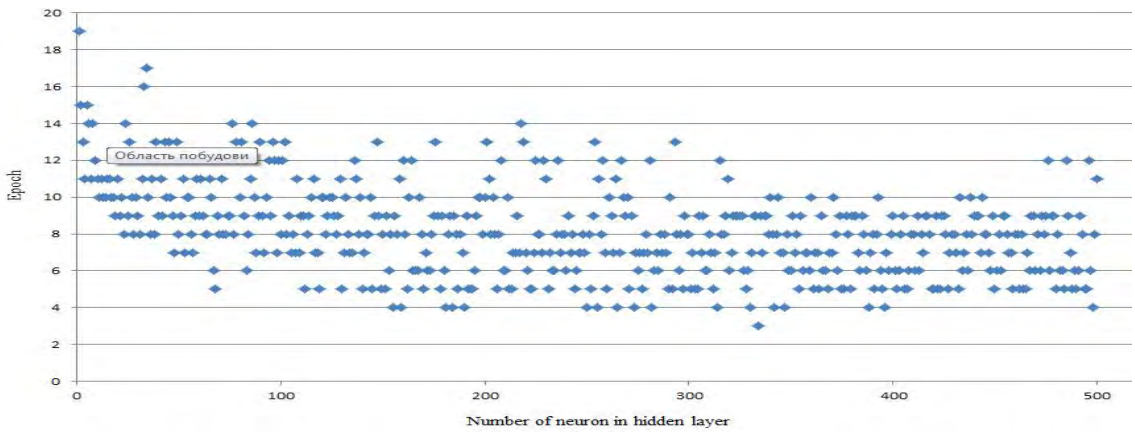


Fig. 2. Dependence of neural inner layer and iterations to achieve an error less than 1 % for a network with 4 input neurons

Dependence error of neurons in the hidden layer was considered to 500 neurons – Fig.1. The results show that increasing the number of neurons does not decrease after 30 iterations and neurons neural network shows redundancy.

As shown in table 2 and fig.1, these rules of thumb do not help in choosing the number of internal neurons. So it changed the number of neurons in the inner layer for this neural network.

Table 3

Dependence of errors of the number of neurons in the hidden layer

Number of neurons in hidden layer	Epoch	Error, %	Input data
1	30	0,95	3/8
	42	0,095	
	1000	0,000354	
2	21	0,87	
	30	0,099	
	1000	0,000087	
3	18	0,75	
	29	0,097	
	1000	0,000065	
6	16	0.76	
	25	0,095	
	1000	0,000049	
9	15	0,96	
	22	0,091	
	1000	0,000059	
18	12	0,63	
	22	0,0790	
	1000	0,000037	
72	9	0,35	
	12	0,034	
	1000	0,0000037	
1	14	0,97	4/12
	26	0,093	
	1000	0,00012	
2	12	0,92	
	22	0,079	
	1000	0,00075	
4	12	0,093	
	17	0,086	
	1000	0,00019	
8	12	0,083	
	17	0,065	
	1000	0,00019	
16	11	0,63	
	15	0,057	
	1000	0,00016	
64	11	0,14	
	16	0,00063	
	1000	0,0000028	

Based on the neural networks, which are presented in table 3, we may form the following conclusions:

1. The reduction error speed decreases with increasing number of neurons in the inner layer, however:

– For some number of the internal neurons the number of iterations for obtaining error less than 1 % is nearly identical;

– When a large number of internal neurons the neural network can converge several times for errors less than 1 %, which indicates its redundancy;

2. At 18 iterations 2 inputs and 4 data for training, we got the optimum settings. Further increasing didn't show any practical improvement. By analyzing the dependence of iterations and data volume, we have received some regularity (18 * 4 = 72):

– In the network with 3 inputs and 8 data to train one of the best value of internal neurons is 9 (72/8 = 9);

– In the network with 4 inputs and 12 data for training one of the best value of internal neurons is 6 (72/12 = 6);

– Taking it as a rule, we have analyzed network for 6 and 8 input neurons, result respectively 4 and 3.

Table 4

Error dependence on the number of neurons in the hidden layer for neural network with 6 and 8 input neurons and 2 output

Number of neurons in hidden layer	Epoch	Error, %	Number of input neurons
1	1000	6,4	6
2	15	0,54	
	1000	1e-4	
4	12	0,25	
	1000	5e-5	
6	11	0,8	
	1000	2e-6	
8	9	0,099	
	1000	8,7e-7	
1	1000	24,4	8
2	9	0,40	
	1000	0,0000063	
3	10	0,11	
	1000	0,0000011	
6	9	0,67	
	1000	8,9e-9	
8	9	0,024	
	1000	5,2e-9	

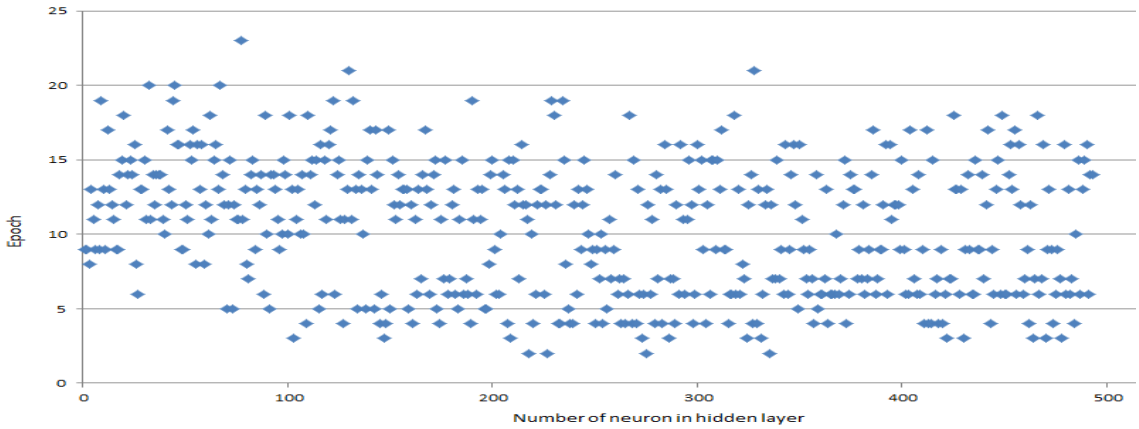
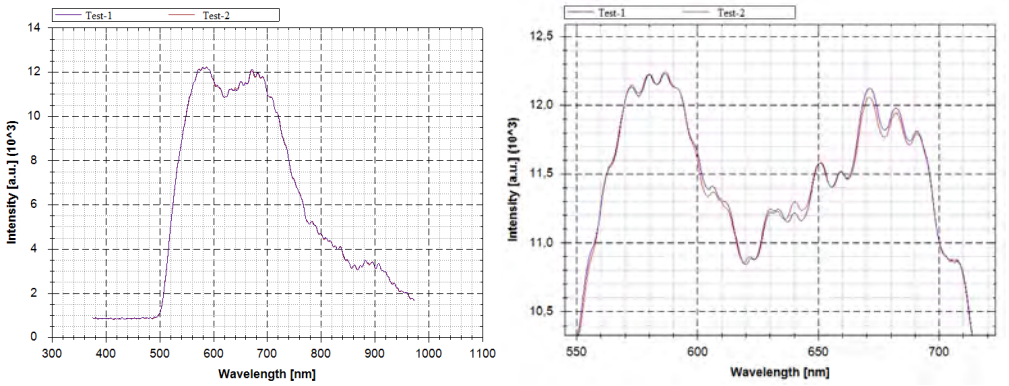


Fig. 3. Dependence of neural hidden layer and iterations to achieve an error less than 1 % for a network with 8 input neurons

As it is shown in Figure 3, optimal number of neurons in hidden layer could be 8 for both networks, ie the previous network number in hidden layers must be 4 – 12 neurons, for 3 – 33 neurons, and for 2 – 66 neurons, which in principle may be considered as optimal.

Neural network for spectrum classification

We have two classes of spectrum water, which differs only in the concentration of iron – 1,3 % and 1.5 %. On their basis will develop the neural network architecture.



а *б*

Fig. 4. The spectra of the two classes of water: a) whole spectrum, b) the information window of the spectrum

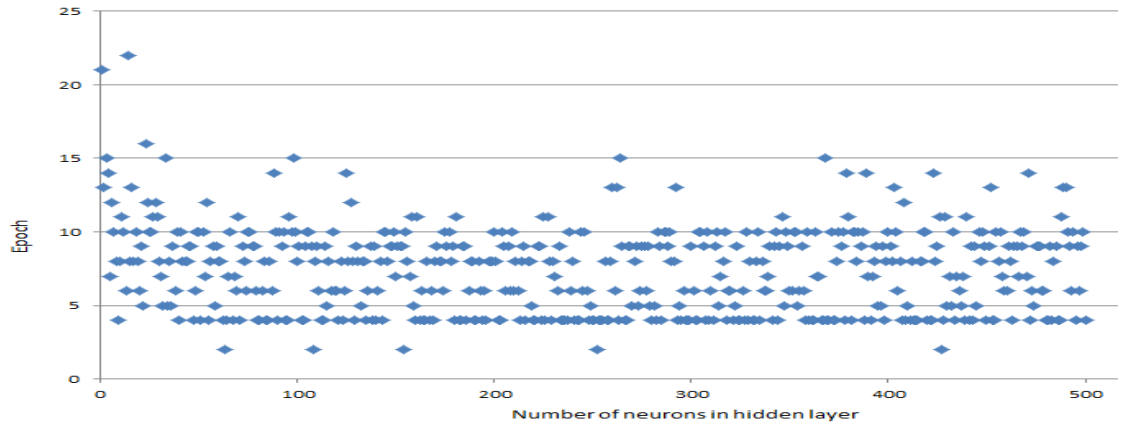


Fig. 5. Dependence of neural hidden layer and iterations to achieve an error less than 1 % for a network with 100 input neurons

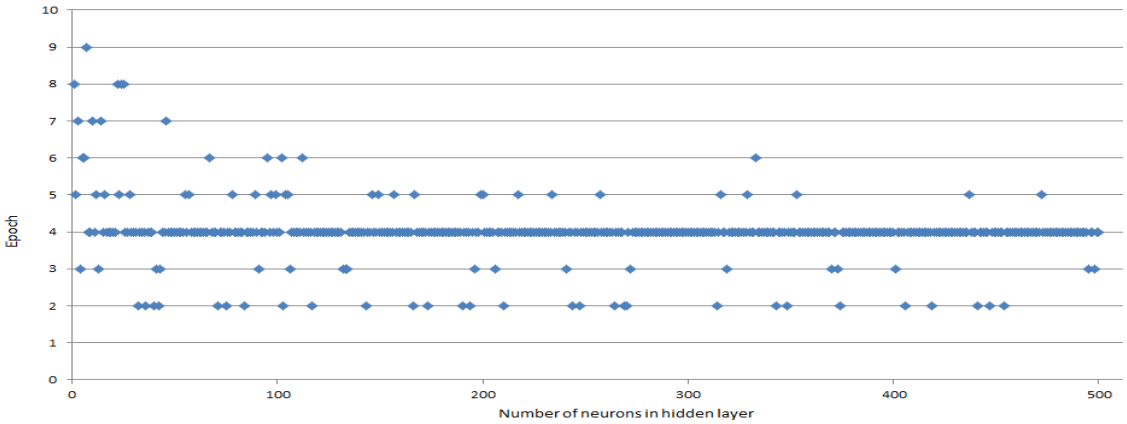


Fig. 6. Dependence of neural hidden layer and iterations to achieve an error less than 1 % for a network with 200 input neurons

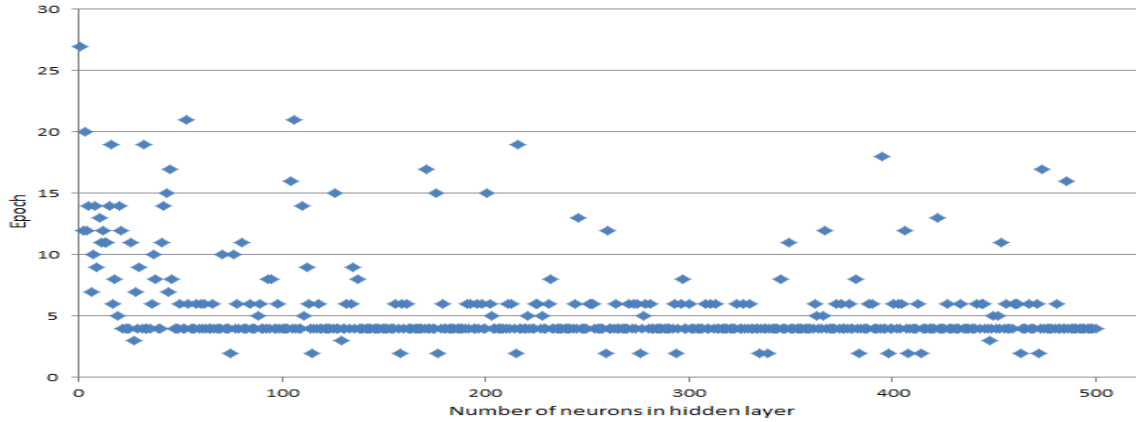


Fig. 7. Dependence of neural hidden layer and iterations to achieve an error less than 1 % for a network with 600 input neurons

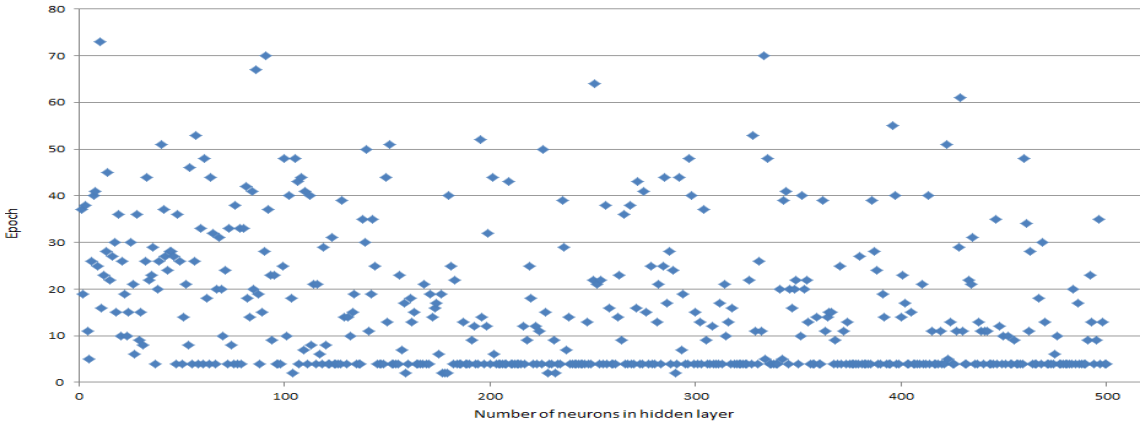


Fig. 8. Dependence of neural hidden layer and iterations to achieve an error less than 1 % for a network with 2048 input neurons

Based on these spectra will analyze the structure of a neural network for 100, 200, 600 and 2048 input neurons with 20 data for training. This analysis will determine the parameters of the neural network for spectrum classification.

In the spectrum the useful information is hold in their peaks. So if one know the wavelengths that are responsible for the peaks, he need only 600 neurons (in this case) instead of 2048, which meet throughout the spectrum.

Based on the dependence parameters of the neural network structure for data classification spectral analysis will look like:

Table 5

Parameters of the neural network structure

Number of input neurons	Number of neurons in hidden layer	Set of data for training
600	75	40
2048	97	40

The article [14] shows that in a following settings: a set of 200 data for training and 120 neurons in the inner layer and 2000 input neurons, the ANN required up to 13 cycles to reach 0.48 % error.

Using this pattern we have made the following network setup:

– 2000 input neurons;

– 50 data for training;

– 90 neurons in the inner layer, while at 27 iterations returned 0.83 %.

By comparing these values with the previous, one can see that with decreasing data and internal neurons the number of iteration have increased in order to achieve an error less than 1 %. Consequently, these two settings are among the possible optimal settings.

Conclusions

The conducted analysis of neural network frameworks showed that Encog provides one of the best results and is easy to use. Determined by empirical rules the number of neurons in the inner layer gave no positive result. The analysis of the number of internal neurons for networks with 100, 200, 600, 2048 input neurons was conducted. This allows to determine the optimal number of neurons. The developed structure of the neural network allows to classify the spectral analysis data with high accuracy.

1. Faitas O., Matviykiv O., Lobur M. Compare Neural Networks Framework for Spectrum Classification // Technologies and Methods in MEMS Design (MEMSTECH), VIIIth International Conference MEMSTECH`2012, 18-21 April 2012, Lviv-Polyana, Ukraine: proc. – Lviv: Publishing House Vezha&Co, 2012. – P. 203-205. 2. Використання нейромережі для побудови систем керування електроприводом верстата-гойдалки / А.В. Маляр, А.С. Андреїшин 3. Burr D. J. 1987. Experiments with a connecnionlist text reader. In Proceedings of the IEEE First International Conference on Neural Networks, eds. M. Caudill and C.Butler. – Vol. 4. – P. 717-24. San Diego, CA: SOS Printing. 4. Sejnowski T. J., Rosenberg C. R. 1987. Parallel networks that learn to pronounce English text. Complex Systems 1:145-68. 5. Comparing Neural Networks in Neuroph, Encog and JOONE http://www.codeproject.com/ Articles/85385/Comparing-Neural-Networks-in-Neuroph-Encog-and-JOO 6. Benchmarking and Comparing Encog, Neuroph and JOONE Neural Networks http://www.codeproject.com/ Articles/85487/Benchmarking-and-Comparing-Encog-Neuroph-and-JOONE 7. Encog home page – http://www.heatonresearch.com/encog 8. Jorge J. Moré, The Levenberg-Marquardt algorithm: Implementation and theory, Lecture Notes in Mathematics, 1978, Volume 630/1978, 105-116, 9. Joone home page – http://www.sourceforge.net/projects/joone 10. Neuroph home page – http://www.neuroph.sourceforge.net/ 11. FANN home page – http://leenissen.dk/fann/wp/ 12. Bengio, Y. & LeCun, Y., 2007. Scaling learning algorithms towards AI. Large-Scale Kernel Machines, (1). – P. 1–41. 13. Boger Z. and Guterman H. Knowledge extraction from artificial neural network models // IEEE Systems, Man, and Cybernetics Conference, Orlando, FL. – 1997. 14. Faitas O., Matviykiv O., Lobur M. Neural Networks for Spectrum Classification Based on Encog // Technologies and Methods in MEMS Design (MEMSTECH), VIIIth International Conference MEMSTECH`2012, 18–21 April 2012, Lviv-Polyana, Ukraine: proc. – Lviv: Publishing House Vezha&Co, 2012. – P. 225.