Applications of Neural Networks in Continuous Casting

GELU OVIDIU TIRIAN, CAMELIA BRETOTEAN PINCA
Department of Electrotechnical Engineering and Industrial Informatics
"Politechnica" University of Timisoara
Revolutiei str., no. 5, Hunedoara
ROMANIA
ovidiu.tirian@fih.upt.ro, camelia.bretotean@fih.upt.ro

Abstract: - This paperwork describes and refers to the structure of neuronal networks who make up the system we use for predicting wire breaking, the way they have been used and implemented; and the use and implementation of the entire system. Before testing the networks, we must identify the design of the RNA input curves. We should identify it experimentally, using the same measurements as for the continuous cast process. For using the serial-dynamic and space network, we need a large amount of data, more than the data that a thermo-couple uses during 120 seconds. Thus, specialists have had to design new software in order to stimulate the difference curves we should use for each network input. Because dynamic-serial networks follow the same pattern of data input, we have preferred to use only one serial-dynamic network and clone the others. We have performed the same in case of space networks whose input data are the same output data from two of the dynamic-serial networks.

Key-Words: neuronal system, prediction, samples, algorithm, training, crack, continuous casting.

1 Introduction

In the process of continuous casting, the melted steel from the melting pot is passed, through the intermediary of the distributor, in the water-cooled crystallizer tank. In this way, a crust forms here which is solidified at the exterior, and one of the great problems is its cracking or even its tearing, due to several factors [1].

When the portion that has suffered the crack gets out of the crystallizing apparatus, the cast iron pours out and the casting process must be stopped. Such an accident must be avoided by detecting all cracks and reducing the casting speed, allowing the iron to become solid [6], [11], [21].

There was established that when a crack occurs, the liquid steel touches the crystallizers wall, causing an increase in it's temperature. That's why, the crack can be detected by means of several heat sensors mounted on the crystallizer's wall both on its width and on the direction of casting [1], [14].

In the paperwork „Neural system for detecting cracks in the wire of the continuous casting", published during the 12th International Research/Expert Conference ″Trends in the Development of Machinery and Associated Technology TMT 2008, Istanbul, 2008, pp. 649-652, the author, Tirian G.O, has accomplished a prediction system of wire breaking, by using a number of temperature sensors mounted inside the crystallizing apparatus wall, and whose signals have been analised by a multiple-neuronal system who is able to read all the data received from the thermal-couples and to come up with an appropriate answer.

Neuronal networks prove to be useful for solving some difficult problems, such as: estimating, identifying, predicting, and controlling or for complex optimization [9], [10], [12].

Because each operation should be independent – we refer to all components of the system – connection patterns may work in parallel [22]. The way data is memorized and processed differentiate the artificial neuronal networks from all classic software. The classic software follows the instructions according to a pre-defined sequential order. Due to their features, which enable them solve any difficult problem, based on a large number of examples, connection systems we use for different occasions: shape acknowledge systems or signals, systems for controlling complex processes [24].

This paperwork describes and refers to the structure of neuronal networks who make up the system we use for predicting wire breaking, the way they have been used and implemented; and the use and implementation of the entire system.

2 Experimental determination of the temperature oscillations

Due to the fact that dynamic-serial networks follow the same pattern of the input data, we have preferred to use only one dynamic serial-network and cloning all the others. The same thing happenend in case of spatial networks, who use the output data of the other two dynamic-serial networks as input data.

We have identified the characteristics of the temperature oscillations when we consider some entry data for the continuous breakout prediction process [21]. This has been an experiment enabled due to measurements of the continuous moulding process.

On one aside of the crystallizing apparatus, we have installed some sensors for measuring the temperature – 12 lines and 4 columns-matrixes. Therefore, the number of sensors that have been installed was 48. We have used a special method: we have opened up the double walls of the crystallizing apparatus and have installed the temperature sensors inside.

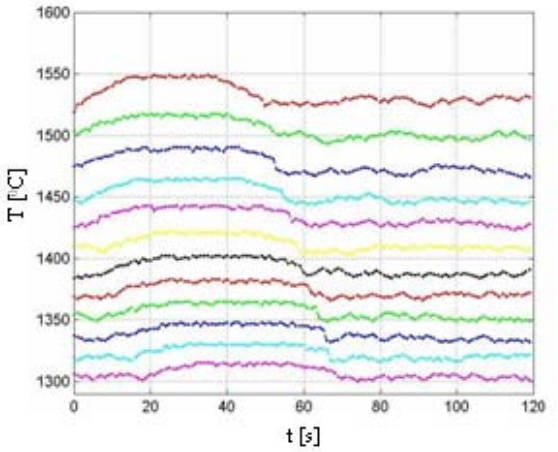

Fig. 1 Temperature curves at the beginning

Each sensor registers a temperature value, which we have monitored – it lasted for 1 second. We have registered the data to equal laps of time of 120 seconds. We have registered the data for different moments, such as for the moment we have started moulding (Fig. 1), and when the curve had a certain shape. Another curve method is the one registered at a certain moment in time, after we have started casting, but right after the curve had reached the normal value (Fig. 2) – thus, it has small variations.

The last type of curve that has been registered in case of cracks (Fig. 3) - in this case, the temperature increases very much along a parable, so that it should the highest, value (almost 100 °C over the normal limit) and it should decrease asymptotically under the normal value.

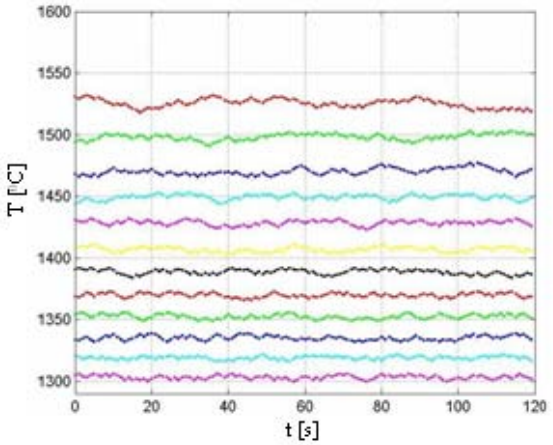

Fig. 2 Temperature curves established for a of the moulding certain moment during the moulding

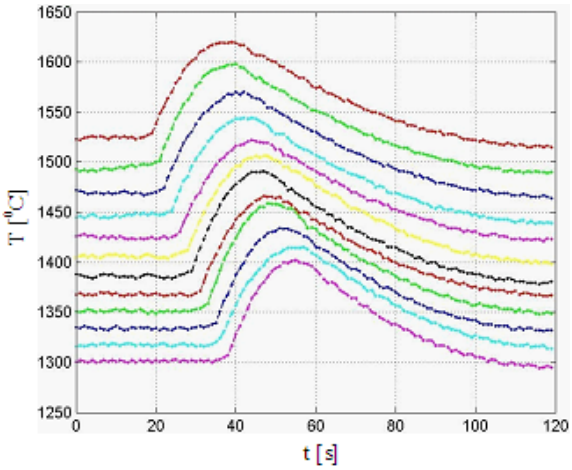

Fig. 3 Temperature curves in case of cracks

3 Network design

Most of the neuronal networks we use in practise are Multilayer Perceptron (RNA-MLP, Multilayer Perceptron) and use the back-propagation learning algorithm [2], [16], [23]. The back-propagation algorithm uses any error that might occur in case of current inputs (as the result of the calculation, propagating the input values written on the patterns) and the expected outputs (those enabled by the current pattern), in order to adjust each percentage. We sequentially adjust the percentage, from the last layer (the output layer) to the first layer (the input layer) [3], [15], [20].

In the case of the dynamic-serial network, which receives the serial-dynamic of the temperature of the individual thermo-couples as input data (both upper and bottom layer), as well as for the space network, which receives the inputs, the output values of the

two dynamic-serial networks have used a neuronal artificial network pattern - Multilayer Perceptron. This network is a totally connected feedforward network (who propagates the signal forwards) [7], [18].

A neuronal Multilayer Perceptron network is a complex structure of artificial neurons organized into levels and whom we connect, so that we cannot connect a neuron otherwise than to the next higher level [17] .

The feedforward totally connected networks (Fig.4) are characterized by the fact that a node from a certain level receives as field the entire next lower level and they are made up by [5], [25]:
- an input layer;
- a null or several hidden layers;
- the output layer.

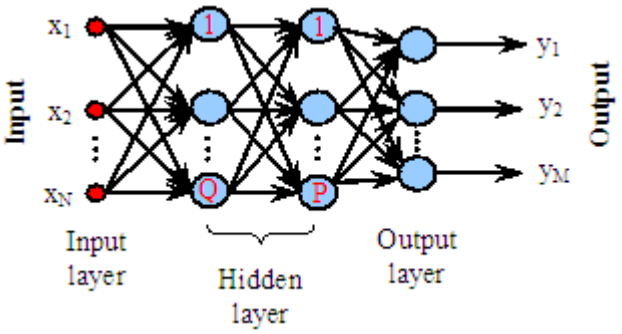

Fig. 4 Two-layer Perceptron

They are used for increasing the performance and solving out some difficult problems. The output of a level is the input for another level. The levels in-between the input pseudo-level and the output level are called hidden levels [8].

The interest for such neuronal networks has been caused by their ability of operating with data different from that of the training stage and by learning to use a random distribution of the synapse percentages of the network. Thus, such type of networks could be used successfully for different use that contains classifiers [23].

There are two stages who enable us accomplish the RNA application. The first stage is the training stage or learning stage [2], when we use pairs of input-output who should be correctly associated, and the RNA changes the free parameters in order to learn those associations. The second stage refers to using the RNA. In this case, we could use input vectors different from those used during the training stage, and the RNA must give the proper answer, based on the generalizing features.

3.1 The design of the dynamic-serial network

Choosing the inputs (their number and type)

Generally, the most difficult problem is to choose the inputs [25]. The outputs of the network are designed by the problem we analyse. An empirical rule of choosing the inputs is the following: "The more data, the better!" This rule is true in case of the network inputs and so is the number of training patterns.

Additional inputs do not influence the accuracy of the results we receive from the network, although some inputs prove to be less important for determining the right output. Although, all simulators have higher neurons limit–consequently, a higher input limit.

Thus, when we gather the data and establish the inputs of the network, we should not provide the second level with similar input vectors, because they might give confliction results at the output. It is important to gather the right amount of input data, as well as the way we use them within the network. Most of the simulation devices allow inputs whose value reach 0-1 or from -1 and +1. Therefore, real data should be pre-processed in order to belong to the same pattern. Most of the simulation devices come along with this process. The way they chose the most important inputs for the network and the way the simulator parameters are set up help us to come up with the best neuronal network or not.

For a dynamic-serial network**,** we could use the analogical inputs. Due to this process we are about to describe that such inputs have both negative and positive values. The network [21] receives the data from each individual thermo-couples from the upper and the bottom layer, meanwhile the temperatures the temperature sensors measure reach higher values than those of the input data allowed by the network, using a corresponding differentiating formula: $\Delta T = T(t_2) - T(t_1)$ the temperatures the sensors measure – for time t_1 and t_2; by scaling them, we are able to reach the right values of the input data for a dynamic-serial network. Such difference of input data is used for a dynamic-serial network and it should register any change of temperature. Once the data are differentiated, they are gathered into 10 buffers for each sample cycle and memorized into 10 data stock units, for each dynamic-serial network.

We should present the input data sequentially, by taking each step equal to one, for each input data of the 10 buffers (Fig. 5). This process may look like a pile, where the first element we introduce is the first element we take out of the pile. We put it

into the upper side of the pile, and we take it from the bottom side.

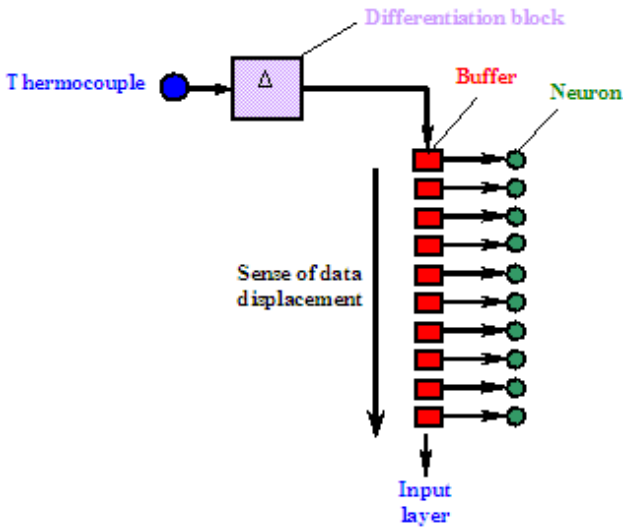

Fig.5 Presenting the data of a serial-dynamic input

The number of input neurons of a dynamic-serial network should be equal to ten.

Choosing the output

The number of outputs is generally required by the application. We need a number of neurons who should be equal to the number of distinct classes that the network should acknowledge.

In the case of a dynamic-serial network, the number of output neurons is equal to one. This neuron shows that the temperature features of an input neuron are suspicious and it might prove whether the wire might break or not. The result of the input neuron is a figure between 0...1, and it is the result of the acknowledge process of the results performed by the dynamic-serial network.

Determining the number of hidden layers and neurons to each hidden layer

The most appropriate number of hidden layers and neurons for each hidden layer is difficult to establish. Generally, one hidden layer is enough to solve most of the problems. In some cases, we could use two or three hidden layers.

As a rule, the number of neurons for each input layer and/or output layers is caused by the features of the application. The neurons of the hidden structures are very important for detecting the features, the legal and regular features contained by the training patterns [23].

As far as the classification problems are concerned (not approximation or shaping up), when the network acknowledges a class from an infinite set of possible classes it is enough for a hidden layer. We could use more hidden layers so that the network should be used faster.

A bigger number of hidden neurons for each layer negatively influence the ability of RAN generalization. It also leads to increasing the volume of data that are about to be processed and the time for the training stage. A smaller number of neurons are not enough for shaping up a less appropriate inner data representation – it may lead to an increased square average error, which corresponds to the test data, as well as to the training data [16].

To conclude, the most appropriate number of hidden neurons should be determined after some experiments.

In case of a dynamic-serial network, we have chosen one single hidden layer. During the first stage, we have chosen a large number of neurons for the hidden layer, which has led to a very big network training time. Secondly, we have chosen a smaller number of neurons, and the network capacity of prediction has decreased a lot – the network has problems when it has to acknowledge simple cases. Based on the experiments, we have reached the conclusion that a number of 8 neurons for a hidden layer are enough to enable a fast training process, and the current network is going to acknowledge the input patterns very accurately.

We have chosen the „trainscg" as a training algorithm. This training algorithm changes the percentage according to the method of the conjugated gradient we have previously presented. We could use this algorithm for training any neuronal network, as long as the percentage, inputs, and transfer functions could derivate. Backpropagation algorithm [13], [15], [25] is used for calculating the partial performance derivates reported to the percentage and to the „bias" value. This algorithm does not make up a line search up for each iteration.

We have used the following activating functions (Fig.6): Hyperbolic tangent sigmoid transfer function and Logarithmic sigmoid transfer function.

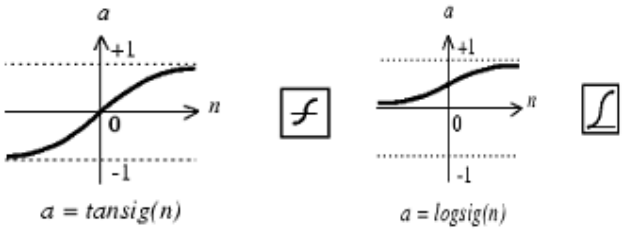

Fig. 6 Activating functions

3.2 The design of the space network

Choosing the input (their number and type)

The input of a space network is analogical. At the input, the space network receives the output values of the two serial-dynamic networks, in order to acknowledge the relationship among the thermo-couples. The serial-dynamic networks have an input result between 0 ... 1; space networks have their input data within the same values: 0 ... 1.

The input value from a dynamic network [21] (at the output) is used for sampling the 6 buffers (Fig.7) in addition, gathered by the 6 stock units. The highest value received from the 6 stock units is introduced into the input level of the space network. The highest value is introduced at the input level in order to correct the propagation time of a crack into the crust of the additional thermo-couples from the upper row.

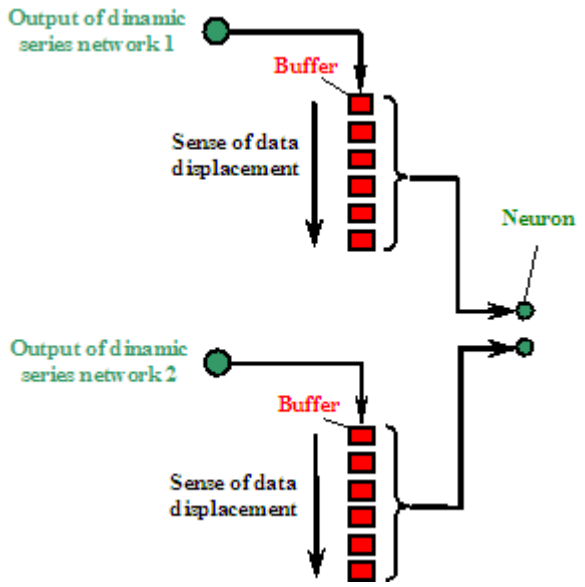

Fig. 7 The method of input data presentation of a space network

The input level of the space network is made by two neurons.

Choosing the output

The output level of a space network is made by one neuron and it starts the breaking alarm when the output value exceeds a limited pre-determined value.

The result of the network output is:

0 – in case there is no crack;

1 – in case there is a wire crack.

Determining the number of hidden layers and neurons for each hidden layer

We have tried to perform different experiments for the number of hidden layers and the number of neurons for each layer. We have studied a case and we have used two hidden layers, but the training time has increased a lot, and the accuracy of the network has been very low. The next step was to choose one hidden layer and establishes the number of neurons. There have been some problems that caused a huge error when acknowledging the input patterns. After several trials, we have reached to the conclusion that a single hidden layer who contains 4 neurons reduces the training time and the error is equal to zero.

The training algorithm we have used for the space network is the Levenberg–Marquardt method (trainlm) [4], [13]. This algorithm is used a lot.

We have used for activating function, the following functions: Hyperbolic tangent sigmoid transfer function and Logarithmic sigmoid transfer function.

3.3 Generating the input data for training of dynamic- serial network

All the programmes we use for generating training data in case of a dynamic serial-network has been accomplished using Matlab. We have generated two variables who had stocked the input data ("In.m") and output data ("Out.m"), as two matrixes, such as:

Table 1

Name of the variable	Name of the matrix	Dimensions
In.m	Pantr1	10 x n
Out.m	Tantr1	1 x n

where n represents the number of the columns of the two matrixes, according to the sampling.

The entry curve has been divided into several samples, each equal to 1, and the data has been gathered in both cases, and either there has been a crack or not. All the data has been studied for different moments in time (Fig.8).

We have studied some sets of 10-values (because the dynamic serial network has 10 inputs), and they have been successively used at the input. For each set of 10 values, the software has generated (in case of the output variable) the right answer the network should come up with at the output. Thus, we have been able to make up the following scale of input data:

Table 2

Input date (x_i)	Output date (y_i)
$x_i \leq 0$	0
$x_i \in (0, 10)$	$x_i/10 \Rightarrow x_i = 0.1 \ldots 0.9$
$x_i \geq 10$	1

We have considered that the exceeded limit value ($y_i = 0.6$) of the dynamic-serial network should admit any cracks as the outcome of a dynamic temperature variation. Table 3 refers to some examples of data sets who are possible at the input of the dynamic-serial network [21], as well as the results it should give at the output. This data have been simulated with Matlab.

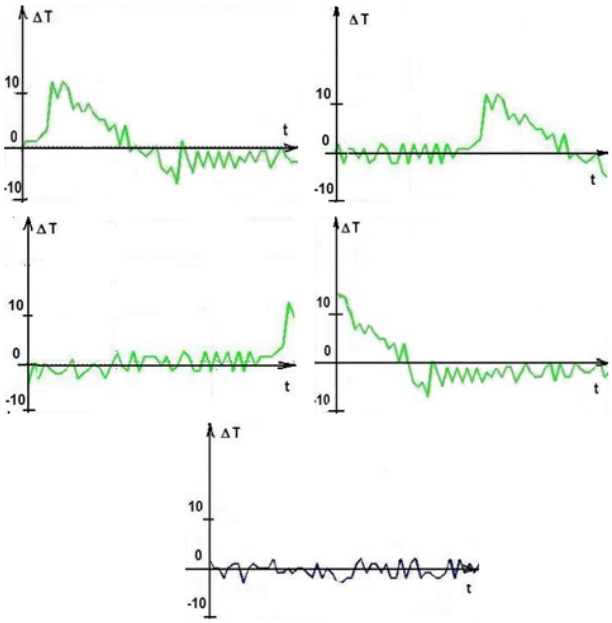

Fig. 8 Sampling modes for the input curve in the dynamic-serial network

Table 3

Input date	3	1	3	-1	-1	3
	1	3	2	-1	2	1
	3	2	2	2	1	2
	2	2	3	1	2	4
	2	3	1	2	3	8
	3	1	1	3	11	7
	1	1	1	11	9	7
	1	1	1	9	10	6
	1	1	0	10	8	7
	1	0	-1	8	7	8
Output of RNA 1	0.3	0.3	0.3	1.0	1.0	0.8

3.4 Training of the dynamic serial-network

After we had designed the neuronal system, we have obtained a number of 48 dynamic-serial network and 33 spatial networks [21]. Considering that dynamic-serial networks have 10x8x1 neurons, meanwhile the spatial networks have 2x4x1 neurons, the system lasts for a long time. Because of the research, we have performed, the system made up of 81 networks lasts for almost 30 hours, and the training data set has been limited to a certain size, thus it has been not enough for the system to perform correctly.

If we study the temperature pattern that the dynamic-serial and spatial networks use, we could see that the dynamic-serial networks of the temperature follow a certain temperature pattern (pattern 1), which corresponds to the dynamic increase or decrease of the temperature who has been measured by the thermocouples. Better said, the input data represent the difference between the corresponding thermocouples, either the cracks occur or not. Spatial networks follow another pattern (pattern 2), who represents the result obtained at the output of the first type of network. Because all dynamic-serial networks follow the same temperature pattern, meanwhile all spatial networks follow another temperature pattern, we could reduce each type of neuronal network at the time when using the system. Thus, the dynamic-serial network works apart from the spatial network.

Because all dynamic-serial networks follow the same output pattern, it is enough to use one dynamic-serial network, meanwhile cloning the others. Thus, the time a network uses to work is reduced within a few minutes. The data we use for enabling the network could be used for all cases in order to use the network correctly, either the data set is rather large.

The program we use for dynamic-serial networks has been made up by Matlab. We have designed the features of the network – we refer to the number of neurons for each input layer: = 10, the number of hidden neurons: = 8, and the number of output layer neurons: = 1. We have established the number of the columns of the input and output matrix - equal to 33. The next stage of the programme refers to generating the input and output data. We should check out if the data has been already generated, and if they have been then we should save them directly on the hard disk. In case this data has not been yet generated, it should be generated by launching the corresponding scripts automatically.

After we have uploaded the data, we should define the dynamic-serial network and all the corresponding parameters. Thus, learning rate η=10e-3, momentum α = 0.01, min_degree =1e-16, number of stages = 5000, and goal = 0.01. We should also point out the training functions as: 'tansig' and 'logsig', and the training algorithm: trainscg.

The next step is to highlight the training mode – to highlight the network we should used, the number

of the input variables, and the name of the output variables [8].

Fig. 9 represents the result of the using progress of the neuronal network, by using the training algorithm - Scaled conjugate gradient back propagation. We are able to see the difference between the output value and the signal we need - 0.001 – for a number of stages equal to 76 – it is much smaller than the one showed by the programme – 5000.

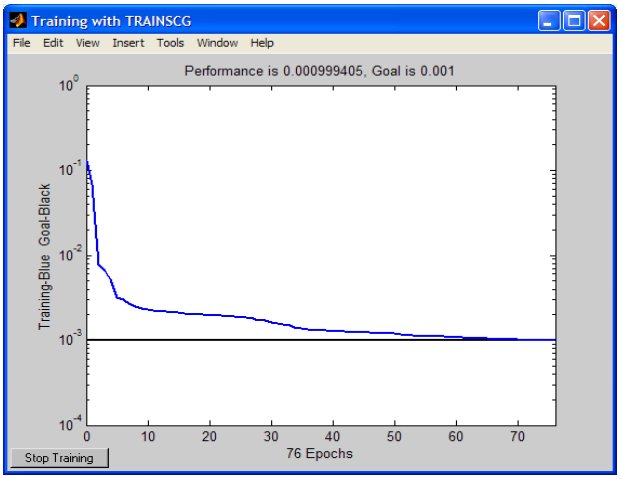

Fig. 9 The training of the dynamic series network using Scaled conjugate gradient backpropagation

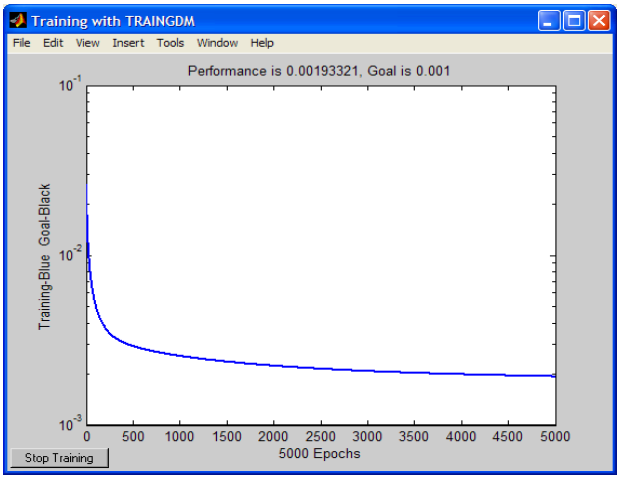

Fig. 10 The training of the dynamic series network using Gradient descent with momentum backpropagation

By using another training algorithm – Gradient descent with momentum back propagation, (Fig. 10), the difference between the output of the neuronal network and the signal we need - 0,001 – has not been reached, either we have performed the 5000 training stages.

The network has been used successfully, with the help of the data generated by the corresponding Matlab programme. By testing the network using different series of real data (either crack should occur or not), and the data we have measured during the industrial process, the network admits all the cases at the input – cracks or no cracks, with a 100% precision.

When using the dynamic-serial network, we use 330 samples; 50 samples in case there is any crack, and the rest in case there has not been any crack. The process stops when the difference between the outputs value of the neuronal network and the desired signal has reached 0.1 or less for each sample; or, when calculating the frequentative convergence, we reach to 5000 stages.

4 Input data for the spatial network training

4.1 Generating the input data for training of spatial network

Input data for spatial network is made up by the output data of the two corresponding dynamic-serial networks. The result of such networks is a number within 0...1; thus, the input data for spatial networks reach the same interval.

The spatial network has two nodes at the entry. One of the input values received from the output of a dynamic-serial network is used for enabling the samples of the 6 buffers and stocked by 6 stock units. The highest value of the 6 stock units is introduced into the input level of the spatial network.

If we know the way the input data looks like (in case of the second type of neuronal networks) – if we know the pattern of the output data of the dynamic-serial network, we are able to fulfil the simulation of a number of data bigger than the real one. This data should be used for enabling one spatial network.

The training data-generating programme we use in case of spatial networks have also been generated by Matlab. We have generated two variables and we have stocked the input data (“Pin.m”) and the output data (“Tout.m”) as some matrixes, such as:

Table 4

Name of the variable	Name of the matrix	Dimensions
Pin.m	Pantr2	2 x n
Tout.m	Tantr2	1 x n

where n represents the number of the columns of the two matrixes.

The programme has generated some numbers within 0...1, considering some conditions and cases where data could occur. The network has only two input neurons, so we need to have a two-line matrix. In order to generate the answer of the network, according to the input data we have used, we have performed the following: in case the two input values are bigger than a limit value, for instance 0.5, the neuronal spatial network should come up with 1-value at the output. In other case, when the value reaches 0, the neuronal network does not come up with the breaking value. If we test the network for different series of data at the output of the dynamic-serial networks, this data should indicate that there are no cracks; the network admits all the cases it deals with a 100% precision. Table 5 presents a few examples of data sets at the input of the spatial network, as well as the results it should produce at the output. All the data has been simulated with Matlab.

Table 5

Input date	0.8	0.3	0.7	0.2	0.5	0.7	0.1	1.0	0.1	0.9	0.0
	0.9	0.6	1.0	0.1	0.6	0.6	0.6	0.9	0.0	0.6	0.5
Output of RNA 2	1	0	1	0	0	0	0	1	0	1	0

4.2 Training of the spatial network

The input data we use for the spatial network is situated within the 0...1, and those is the results of the auxiliary dynamic-serial networks' outputs. Thus, we are able to say that spatial networks have the same temperature pattern. Thus, we could use only one spatial network, meanwhile the rest are simple clones of the one we use.

Using the spatial network, we use 200 samples. Thirty samples of the 200 are used in case any crack should occur. The rest we use in case there is no crack. We may stop sing them when the difference between the output value of the neuronal network and the signal we need has reached 0.001 or less for each sample, or when calculating the frequentative convergence and reaching to 10000 stages.

The training programme for the spatial neuronal network has been performed using the Matlab and we call it Training_RNA_2. We have established the design of the network, the number of neurons from the input layer = 2, the number of hidden neurons = 4, and the number of neurons from the output layer = 1.

We have established the number of the columns of the input matrixes is equal to 200. The next stage of the programme is to generate the input and output data. We check out if the data has been already generated; and if they have been, the data should be directly uploaded on the hard disk. If this data has not been generated yet, it should be generated by launching the corresponding scripts automatically.

After we have uploaded the data, we establish the dynamic-neuronal network and all the corresponding parameters. Thus, learning rate η= 10e-3, momentum α=0.01, min_grad=1e-16, number of stages = 10000, and goal = 0.001. We have also specified the training functions as: 'tansig' and 'logsig'**;** as well as the training algorithm: trainslm. The next step is to point out the way the training should be performed. We should point out the network we should use, the number of the input variables, and the numbers of the output variables for the network.

Fig.11 represents the outcome of the training process of the neuronal network, using the training algorithm Levenberg - Marquardt back propagation [4],[19]**.** We are able to see the difference between the output value of the neuronal network and the signal of 0.001 – for a number of stages equal to 76, much smaller than the one shown by the software – 10000.

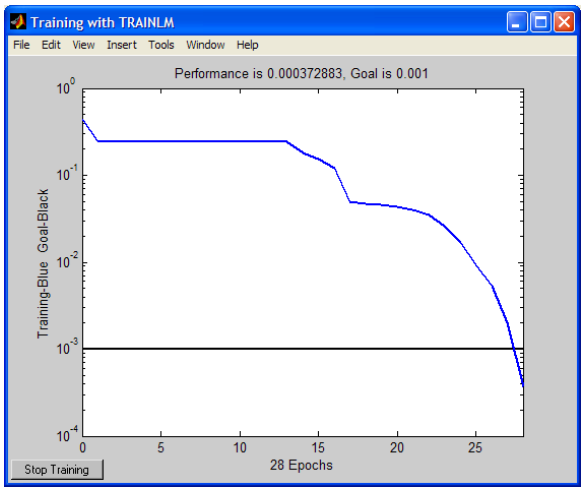

Fig. 11 Training of the dynamic series network using the Levenberg – Marquardt backpropagation

While testing the spatial network with a new set, the network acknowledges the crack with 100% accuracy; as well as in case of normal cases, where there has been no crack.

We should mention that the spatial network and the dynamic-serial network have been trained separately.

After training and testing the two types of networks, we have tested the multiple-neuronal system for predicting the crack in case of continuous

casting. We have initiated the variables we have used, such as the number temperature samples, the number of layer thermocouples, the number of layers of thermocouples, the number of networks directly connected to the output, the number of dynamic-serial and spatial networks (they are calculated using the formulas we have previously described), and the number of directly-connected networks. We have established the design of each neuronal network - the number of neuronal for each input, hidden, and output layer. After we have established all these features, we have generated and loaded the types of training. We have performed these along with performing the separate training for the two networks – we have looked up for the hard disk for the data. If we have found the data then, we have loaded it directly. If not, we have generated it. The next step was to implement the dynamic-serial and spatial neuronal networks. Then, we have trained them. Once we have finished training the networks, we could test the completely neuronal system. The system has acknowledged all the cases brought up to the input with 100% accuracy.

5 Conclusions

In this paperwork, we have established the design of the dynamic-serial network of each neuronal network, as well as for training the networks with the help of the Matlab software.

We have also performed the training and implementing of the entire neuronal system so that it should be able to predict the cracks that might occur throughout the process of continuous casting. The system acknowledges all the cases it should deal with. Thus, 120 cases of the entire number of cases has not suffered any crack, meanwhile 30 cases has suffered from some cracks. The neuronal system uses a completely set of data from the testing data previously described, but it nevertheless acknowledged any crack or not with a 100% accuracy.

References:

[1] J. Adamy , Device for early detection of run-out in continuous casting, *United States Patent*, No.5, 904, 202, Date of Patent 18 may, 1999.

[2] C. Avila , Y. Tsuji, Crack width prediction of RC structures by Artificial neural networks, *Adaptive and Natural Computing Algorithms*, Springer Viena, pg.92-95, 12.dec. 2005.

[3] S.B. Cho, J. H. Kim, Rapid Backpropagation Learning Algorithms, *Circuits Systems Signal Process*, Vol.12, No.2, 1993, pp.155-175.

[4] M. T. Hagan and M. Menhaj, Training feedforward networks, with the Marquardt algorithm, *IEEE Trans. Neural Networks*, vol.5, pp. 989-993, Nov.1994.

[5] M. T. Hagan, H. B. Demuth, M. H. Beale, Neural Network Design, *Boston, MA: PWS Publishing*, 1996.

[6] L. Herbert Gilles, J. Shipman, Method and apparatus for controlling heat removal by varying casting speed, *United States Patent*, No.4, 235,276, Date of Patent 25 nov. 1980.

[7] C. K. Liew, M. Veidt, Guided Waves Identification in Beams with Test Pattern Dependent Series Neural Network Systems, *WSEAS Transactions on Signal Processing*, Issue 4, Vol.4, pp.86-96, April 2008.

[8] V. Lupu, C. Lupu, N. Morariu, The implementation of clean production and the use of neural networks in forecasting waste management, *WSEAS Transactions on Systems and Control*, Issue 9, Vol.3, pp.722-736, September 2008.

[9] P. Makvandi, J. Jassbi, S. Khanmohammadi, Aplication of Genetic Algorithm and Neural Network in Forecasting with Good Data, *WSEAS Transactions on Systems*, 4(4), pp.337-342, 2005.

[10] K. S. Narendra and K. Parthasarathy, Identification and control of dynamic systems using neural networks, *IEEE Trans. Neural Networks*,Vol.1, pp. 4-27, 1990.

[11] T. Nakamura, K. Kazuho, Breakout prediction system in a continuous casting process, *United States Patent*, No.5, 548,520, Date of Patent 20 aug.1996.

[12] Y. Ozel, I. Guney, E. Arca, Neural Network Solution to the Cogeneration System by Using Coal, *12th WSEAS International Conference on Circuits*, Heraklion, Greece, July 22-24, 2008.

[13] P. Pivoňka, J. Dohnal, On-line Identification Based on Neural Networks Using of Levenberg-Marquardt Method and Back-propagation Algorithm, *WSEAS Transactions on Systems*, Issue 2, Vol.3, pp.381-385, April 2004.

[14] F.P. Pleschiutschnugg, Method and apparatus for the early recognition of ruptures in continuous casting of steel with an oscillating mold, *United States Patent*, No.US 6,179,041 B1, Date of Patent 30 jan. 2001.

[15] M. Riedmiller and H. Braun, A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm, *Proceedings of the 1993 IEEE International Conference on Neural Networks*, 1993, pp. 586-591.

[16] O. Sang – Hoon, L. Youngjik, A Modified Error Function to Improve the Error Back-Propagation Algorithm for Multi-Layer Perceptrons, *ETRI Journal*, Vol.17, No.1, April 1995.

[17] M. Sarevska, A.B.M. Ssalem, N. Mastorakis, Null Steering Beamformer Based on RBF Neural Networks, *12th WSEAS International Conference on Communications*, Heraklion, Greece, July 23-25, 2008.

[18] H. T. Shandiz, H. G. Tehrani, H. Hadizadeh, Using Multi Layer Perceptron Network to Classify Road Cracks, *WSEAS Transactions on Systems*, 4 (4), pp.355-358, 2005.

[19] A. A. Suratgar, M. B. Tavakoli, and A. Hoseinabadi, Modified Levenberg-Marquardt Method for Neural Networks Training, *Proceedings of World Academy of Science, Engineering and Technology*, vol.6, 2005.

[20] S. Sureerattanan, H. N. Phien, et. al, The Optimal Multi-layer Structure of Backpropagation Networks, *Proceedings of the 7th WSEAS International Conference on Neural Networks*, Cavtat, Croatia, June 12-14, 2006.

[21] G.O. Tirian, Neural system for detecting craks in the wire of the continuous casting" *12th International Research/Expert Conference "Trends in the Development of Machinery and Associated Technology TMT 2008*, Istanbul, Turkey, 26 – 30 august, pp. 649-652, 2008.

[22] D. Thukaram, H. P. Khincha, H. P. Vijaynarasimha, Artificial Neural Network and Support Vector Machine Approach for Locating Faults in Radial Distribution Systems, *IEEE Transactions on Power Delivery*, Vol.20, No.2, April 2005.

[23] C. Volosencu, Sisteme fuzzy si neuronale, *Editura Politehnica*, Timisoara, 2007.

[24] C. Volosencu, Identification of Distributed Parameter Systems, Based on Sensor Network and Artificial Intelligence, *WSEAS Transactions on Systems*, Issue 6, Vol. 7, pp.785-801, June 2008.

[25] W. Xudong, Y. Man, C. Xingfu, Development of Prediction Method for Abnormalities in Slab Continuous Casting Using Artificial Neural Network Models, *ISIJ International*, Vol.46(2006), No.7, pp. 1047-1053, 2006.