

# Average monthly liquid flow forecasting using neural networks

C. Barbălată\* and L. Leuştean\*\*

\* “Marin Preda ” High School, 17 Ruşetu Street, Bucharest, Romania,

\*\* National Institute for Research and Development in Informatics-ICI, 8-10 Averescu Avenue, Bucharest, Romania,  
E-mail:leo@u3.ici.ro

**Abstract:** The forecasting of different natural phenomena related to river flow is one of the priorities of hydrology in this moment. This paper presents an application of feed-forward neural networks and *Resilient Backpropagation (RPROP)* algorithm to forecast a time series consisting of the average monthly liquid flow measured at Borzii Vineţi hydrometric station from the Jiu River, situated in the Southwest of Romania, in the Petroşani Depression.

## Introduction

The forecasting of different natural phenomena related to river flow is one of the priorities of hydrology in this moment. In order to obtain an adequate prediction it is necessary to observe the phenomena a long period of time, usually more than 30 years.

This paper presents an application of feed-forward neural networks and *Resilient Backpropagation (RPROP)* algorithm to forecasting a time series consisting of the average monthly liquid flow measured at Borzii Vineţi hydrometric station from the Jiu River, situated in the Southwest of Romania, in the Petroşani Depression. The hydrological basin corresponding to this station has an area of 1222 kmp and a medium elevation of 1135 m [9,11]. The data are from the period 1950-1994 [12].

The hydrographic reservoir Jiu, in the Borzii Vineţi region, has a special importance because it collects the waters from a problematic region, mono-industrial, which is confronted with particular problems in the water supply for the population and the coal industry. Downstream of this section, the Jiu crosses another large coalfield; actually it overlaps the most important coal basins in Romania. It provides water for the industrial platform of the Craiova city as well as for three of the biggest thermo power stations in Romania. As a result, being aware beforehand of the flow capacity of the Jiu would favorize a better management of the water in this hydrographic reservoir.

The average monthly liquid flow represents the water volume that flows through the section of a river in a month, reported at the unity of time and it expresses in  $\text{m}^3/\text{sec}$ .

We must mention that in hydrology, for the bigger rivers, because of the inertness of the instruments used for measuring the speed of the water there is the probability that the measured liquid flow to be bigger or smaller then the real one. That's why it was established that a tolerance of  $\pm 10\%$  is in the normal limits of measurement. It follows that a forecasting is considered correct if the value of the deviation is at most  $\pm 10\%$  from the measured value.

This paper is part of a case study on applications of neural networks to forecasting the average monthly liquid flow at Borzii Vineți hydrometric station. In previous papers, we used feed-forward neural networks and *SuperSAB* algorithm [1] and recurrent neural networks and *Backpropagation-through-time* algorithm [2] to predict this time series. Here we use *RPROP*, a very promising algorithm for feed-forward neural networks, introduced by M. Riedmiller in 1993 [3,6]. Besides fast convergence, one of the main advantages of *RPROP* lies in the fact that for many problems the choice of at most one parameter is needed to obtain optimal or at least nearly optimal convergence times.

## 1. Feed-forward neural networks

### 1.1 Basic concepts

A neural network is a parallel distributed information processing structure consisting of processing elements (neurons) interconnected via unidirectional signal channels called connections. Each processing element has a single output connection that branches into as many collateral connections as desired; each carries the same signal - the processing element output signal, which can be of any mathematical type desired.

Neural networks develop information processing capabilities by learning for examples. Learning techniques can be roughly divided into two categories:

1. supervised learning;
2. unsupervised learning.

Supervised learning requires a set of examples for which the desired network response is known. The learning process consists then in adapting the network in a way that it will produce the correct response for the set of examples. The resulting network should then be able to generalize (give a good response) when presented with cases not found in the set of examples.

In unsupervised learning the neural network is autonomous; it processes the data it is presented with, finds out about some of the properties of the data set and learns to reflect these properties in its output. What exactly these properties are, that network can learn to recognize, depends on the particular network model and learning method.

One of the most popular neural net paradigms is the feed-forward neural network. In a feed-forward neural network, the neurons are usually arranged in layers [7]. A feed-forward neural net is denoted as  $N_I \times N_1 \times \dots \times N_i \times \dots \times N_L \times N_O$ , where:

- $N_I$  represent the number of input units;
- $L$  represent the number of hidden layers;
- $N_i$  represent the number of units from the hidden layer  $i$ ,  $i = \overline{1, L}$ ;
- $N_O$  represent the number of output units.

By convention, the input layer does not count, since the input units are not processing units, they simply pass on the input vector  $x$ . Units from the hidden layers and output layer are processing units. Figure 1 gives a typical fully connected 2-layer feed-forward network with a  $3 \times 4 \times 3$  structure.

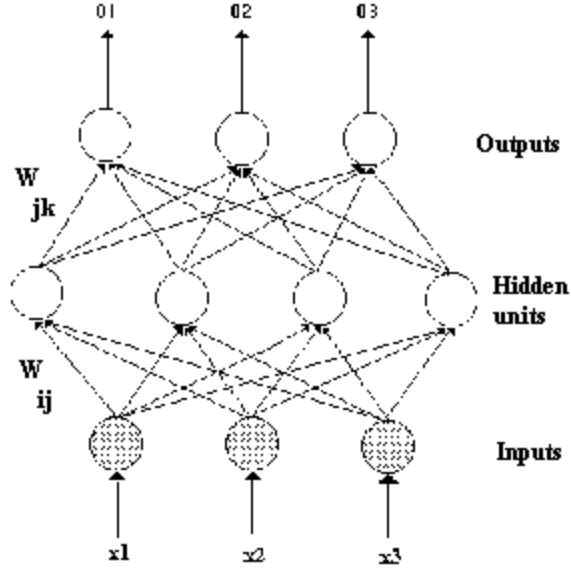


Figure 1: A 3x4x3 feed-forward neural network.

Each processing unit has an activation function that is commonly chosen to be the sigmoid function:

$$f(x) = \frac{1}{1 + e^{-x}}.$$

The net input to a processing unit  $j$  is given by:

$$net_j = \sum_i w_{ij} x_i + \theta_j,$$

where  $x_i$ 's are the outputs from the previous layer,  $w_{ij}$  is the weight (connection strength) of the link connecting unit  $i$  to unit  $j$ , and  $\theta_j$  is the bias of unit  $j$ , which determines the location of the sigmoid function on the  $x$ -axis.

The activation value (output) of unit  $j$  is given by:

$$a_j = f(net_j) = \frac{1}{1 + e^{-net_j}}.$$

The objective of different supervised learning algorithms is the iterative optimization of a so called *error function* representing a measure of the performance of the network. This error function is defined as the mean square sum of differences between the values of the output units of the network and the desired target values, calculated for the whole pattern set. The error for a pattern  $p$  is given by

$$E_p = \sum_{j=1}^{N_o} (d_{pj} - a_{pj})^2,$$

where  $d_{pj}$  and  $a_{pj}$  are the target and the actual response value of output neuron  $j$  corresponding to the pattern  $p$ .

The total error is

$$E = \sum_{p=1}^P \frac{1}{2} E_p = \frac{1}{2} \sum_{p=1}^P \sum_{j=1}^{N_o} (d_{pj} - a_{pj})^2,$$

where  $P$  is the number of the training patterns.

During the training process a set of pattern examples is used, each example consisting of a pair with the input and corresponding target output. The patterns are presented to the network sequentially, in an iterative manner, the appropriate weight corrections being performed during the process to adapt the network to the desired behavior. This iterating continues until the connection weight values allow the network to perform the required mapping. Each presentation of the whole pattern set is named an *epoch*.

One of the most popular supervised learning algorithms for feed-forward neural networks is *Backpropagation* [7]. In this algorithm the minimization of the error function is carried out using a gradient-descent technique. The necessary corrections to the weights of the network for each moment  $t$  are obtained by calculating the partial derivative of the error function in relation to each weight  $w_{ij}$ . A gradient vector representing the steepest increasing direction in the weight space is thus obtained. The next step is to compute the resulting weight update. In its simplest form, the weight update is a scaled step in the opposite direction of the gradient. Hence, the weight update rule is

$$\Delta_p w_{ij}(t) = -\varepsilon \cdot \frac{\partial E_p}{\partial w_{ij}}(t),$$

where  $\varepsilon \in (0,1)$  is a parameter determining the step size and is called the *learning rate*.

A *momentum* may be used with the idea of incorporating in the present weight update some influence of the past iteration. The weight update rule becomes

$$\Delta_p w_{ij}(t) = -\varepsilon \cdot \frac{\partial E_p}{\partial w_{ij}}(t) + \alpha \cdot \Delta_p w_{ij}(t-1),$$

where  $\alpha$  is the momentum term and determines the amount of influence from the previous iteration to the present one.

## 2. The algorithm *Resilient Backpropagation (RPROP)*

### 2.1 Description

The algorithm *Resilient Back-propagation (RPROP)* is a local adaptive learning scheme, performing supervised batch learning in feed-forward neural networks. M. Riedmiller introduced it in 1993. For a detailed discussion see [3,4,5,6].

The basic principle of *RPROP* is to eliminate the harmful influence of the size of the partial derivative on the weight step. As a consequence, only the sign of the derivative is considered to indicate the direction of the weight update. To achieve this, we introduce for each weight  $w_{ij}$  its individual *update-value*  $\Delta_j(t)$ , which solely determines the size of the weight-update.

It is introduced a second learning rule, which determines the evolution of the update-value  $\Delta_j(t)$ . This estimation is based on the observed behavior of the partial derivative during two successive weight-steps:

$$\Delta_{ij}(t) = \begin{cases} \eta^+ \cdot \Delta_{ij}(t-1), & \text{if } \frac{\partial E}{\partial w_{ij}}(t) \cdot \frac{\partial E}{\partial w_{ij}}(t-1) > 0 \\ \eta^- \cdot \Delta_{ij}(t-1), & \text{if } \frac{\partial E}{\partial w_{ij}}(t) \cdot \frac{\partial E}{\partial w_{ij}}(t-1) < 0 \\ \Delta_{ij}(t-1), & \text{else} \end{cases}$$

where  $0 < \eta^- < 1 < \eta^+$ .

In words, the adaptation rule works as follows. Every time the partial derivative of the corresponding weight  $w_{ij}$  changes its sign, which indicates that the last update was too big and the algorithm has jumped over a local minimum, the update-value  $\Delta_{ij}(t)$  is decreased by the factor  $\eta^-$ . If the derivative retains its sign, the update-value is slightly increased in order to accelerate convergence in shallow regions.

Once the update-value for each weight is adapted, the weight-update itself follows a very simple rule: if the derivative is positive (increasing error), the weight is decreased by its update-value, if the derivative is negative, the update-value is added:

$$\Delta w_{ij}(t) = \begin{cases} -\Delta_{ij}(t), & \text{if } \frac{\partial E}{\partial w_{ij}}(t) > 0 \\ \Delta_{ij}(t), & \text{if } \frac{\partial E}{\partial w_{ij}}(t) < 0 \\ 0, & \text{else} \end{cases}$$

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t).$$

However, there is one exception. If the partial derivative changes sign, that is the previous step was too large and the minimum was missed, the previous weight-update is reverted:

$$\Delta w_{ij}(t) = -\Delta w_{ij}(t-1),$$

$$\text{if } \frac{\partial E}{\partial w_{ij}}(t) \cdot \frac{\partial E}{\partial w_{ij}}(t-1) < 0$$

Due to that ‘backtracking’ weight-step, the derivative is supposed to change its sign once again in the following step. In order to avoid a double punishment of the update-value, there should be no adaptation of the update-value in the succeeding step. In practice this can be done by setting  $\frac{\partial E}{\partial w_{ij}}(t-1) = 0$  in the  $\Delta_{ij}$  update-rule above.

The partial derivative of the total error is given by

$$\frac{\partial E}{\partial w_{ij}}(t) = \frac{1}{2} \sum_{p=1}^P \frac{\partial E_p}{\partial w_{ij}}(t).$$

Hence, the partial derivatives of the errors must be accumulated for all  $P$  training patterns. This means that the weights are updated only after the presentation of all training patterns.

In [10] it is introduced a *weight-decay* parameter  $\alpha$ . This parameter determines the relationship of two goals, namely to reduce the output error (the standard goal) and to reduce the size of the weights (to improve generalization). The composite error function is:

$$E = \frac{1}{2} \sum_{p=1}^P \sum_{j=1}^{N_o} (d_{pj} - a_{pj})^2 + \frac{1}{10^\alpha} \cdot \sum_{i,j} w_{ij}^2 .$$

Note that the *weight-decay* parameter  $\alpha$  denotes the exponent, to allow comfortable input of very small values. Hence, a choice of  $\alpha=4$  corresponds to a ratio of weight decay term to output error of  $1:10000$ .

We shall use this version of *RPROP* in this paper.

## 2.2 Parameters

In order to reduce the number of freely adjustable parameters, often leading to a tedious search in parameter space, the increase and decrease factors  $\eta^+$  and  $\eta^-$  are set to fixed values:  $\eta^- = 0.5$  and  $\eta^+ = 1.2$ . See [5] for considerations which led to these values.

At the beginning of the algorithm, all update-values  $\Delta_j$  are set to an initial value  $\Delta_0$ . A good choice may be  $\Delta_0 = 0.1$ . However, the choice of this parameter is not critical at all, for it is adapted as learning proceeds.

In order to prevent the weights from becoming too large, the maximum weight-step determined by the size of the update-value is limited. The upper bound is set by the second parameter of *RPROP*,  $\Delta_{\max}$ . The default upper bound is set somewhat arbitrarily to  $\Delta_{\max} = 50.0$ . Usually, the convergence is rather insensitive to this parameter as well. The minimum step size is constantly fixed to  $\Delta_{\min} = 1e^{-6}$ .

The third parameter of the algorithm is the weight-decay parameter  $\alpha$ .

## 3. Experimental model

A time series is a sequence of time-ordered data values that are measurements of some physical process. A time series forecasting problem can be easily mapped to a neural network. The number of input units corresponds to the number of input data terms. The number of output units represents the forecast horizon. One-step-ahead forecast can be performed by a neural network with one output unit, and  $k$ -step-ahead forecast can be mapped to a neural network with  $k$  output units [8].

This paper presents an application of feed-forward neural networks and *Resilient Backpropagation (RPROP)* to forecast the average monthly liquid flow measured at Borzii Vineți hydrometric station from the Jiu River. The data are from the period 1950-1994 [12]. A forecasting is correct if its value presents a deviation of at most  $\pm 10\%$  from the measured value.

We used in our experiments the program *SNNS (Stuttgart Neural Network Simulator) 4.0*, a software simulator for neural networks on Unix workstations developed at the Institute for Parallel and Distributed High Performance Systems (Institut für Parallele und Verteilte Höchstleistungsrechner, IPVR) at the University of Stuttgart since 1989 [10].

In this paper, we used the batch simulator version *snsbat*, which we improved in order to choose from an input file the network structure, the values of the parameters and the number of runs for each such choice and also to display the results in the desired appearance.

All the experiments were carried out on a Pentium at 133 MHz and with 24MB RAM, on Linux operating system.

We tested different neural network structures with one hidden layer by varying the number of hidden units and we used different values for the parameters appearing in the algorithm *RPROP*. For each neural network structure and parameter setting, 20 experiments were run with different random initial weights, uniformly distributed over the interval  $[-1,1]$ . The reported forecast errors are the averages of the 20 runs. The data series were normalized to the range  $[0,1]$  before feeding into the neural networks:

$$data_{i,j} = \frac{largest\_data - data}{largest\_data - smallest\_data}$$

Forecasts from the neural network outputs were transformed to the original data scale before the percentage of good forecasts was reported.

The parameters used by the algorithm are:

- $\Delta_0$  We set this parameter to 0.1.
- $\Delta_{max}$  We also set this parameter to 50.0.
- $\alpha$  -the *weight-decay* parameter. We used in the experiments the following values for  $\alpha$ : 0, 1, 3, 5, 10, 50.
- $Nh$  -the number of units from the hidden layer.

We consider 4 time series, with the data from 1950-1990+1991, 1951-1991+1992, 1952-1992+1993, 1953-1993+1994, respectively. The notation 1950-1990+1991 means that the data from 1950-1990 were used as training patterns and the data from 1991 were used as testing patterns; similarly for the other three time series. For each of the 4 time series, the monthly liquid flows from the last year were used to test the forecasting performance of the models.

We have two approaches, differing in the number of previous months used for prediction:

### **I 12+1**

We predict the liquid flow from one month using the liquid flows from 12 previous months. Simulating the flow capacity using previous 12 months is based on the fact that in the flow capacity evolution there is a certain annual cycle, imposed mainly by the climatic factors. Hence, a term of the time series consists in 12+1 values. Each of the 4 time series had 492 terms. The first 480 terms were used for training and the last 12 terms were used for testing.

The feed-forward neural networks used in the experiments had 12 input units, one hidden layer and one output unit.

The next tables present the percentages of correct predictions obtained after 3000 epochs.

### **1950-1990+1991**

	$\alpha=0$	$\alpha=1$	$\alpha=3$	$\alpha=5$	$\alpha=10$	$\alpha=50$
Nh=0	0.00%	16.00%	0.00%	0.00%	0.00%	0.00%
Nh=1	0.00%	25.00%	0.00%	0.00%	0.00%	0.00%
Nh=3	13.80%	16.00%	19.65%	18.35%	15.50%	16.30%
Nh=6	12.15%	16.00%	17.95%	13.40%	16.00%	18.00%
Nh=9	15.05%	16.00%	20.50%	18.85%	18.15%	16.40%

### 1951-1991+1992

	$\alpha=0$	$\alpha=1$	$\alpha=3$	$\alpha=5$	$\alpha=10$	$\alpha=50$
Nh=0	16.00%	16.00%	16.00%	16.00%	16.00%	16.00%
Nh=1	25.00%	16.00%	25.00%	25.00%	25.00%	25.00%
Nh=3	12.25%	16.00%	24.95%	14.15%	14.60%	12.15%
Nh=6	13.45%	16.00%	18.40%	13.50%	12.60%	11.45%
Nh=9	12.65%	16.00%	17.55%	9.30%	16.70%	10.00%

### 1952-1992+1993

	$\alpha=0$	$\alpha=1$	$\alpha=3$	$\alpha=5$	$\alpha=10$	$\alpha=50$
Nh=0	16.00%	16.00%	16.00%	16.00%	16.00%	16.00%
Nh=1	16.00%	16.00%	8.00%	16.00%	16.00%	16.00%
Nh=3	10.05%	16.00%	14.05%	10.20%	10.15%	9.30%
Nh=6	6.85%	16.00%	12.95%	13.05%	10.10%	11.05%
Nh=9	8.50%	16.00%	10.95%	8.85%	10.15%	10.15%

### 1953-1993+1994

	$\alpha=0$	$\alpha=1$	$\alpha=3$	$\alpha=5$	$\alpha=10$	$\alpha=50$
Nh=0	25.00%	33.00%	25.00%	25.00%	25.00%	25.00%
Nh=1	8.00%	33.00%	0.00%	8.00%	8.00%	8.00%
Nh=3	15.00%	33.00%	14.61%	17.95%	13.80%	16.20%
Nh=6	9.75%	33.00%	13.85%	15.90%	13.40%	15.10%
Nh=9	11.00%	33.00%	8.80%	13.05%	13.90%	11.85%

### II 3+1

We predict the liquid flow from one month using the liquid flows from 3 previous months. Simulation of the flow capacity by the analysis of the last 3 monthly successive flows was used, because there is certain dependency of the flows in the soil, and the quantity of water accumulated or streamed down in the previous months. In this case, a term of the time series consists in 3+1 values. Each of the time series had 501 terms: the first 489 for training and the last 12 terms for testing. The feed-forward neural networks had 3 input units, one hidden layer and one output unit.

The next tables present the percentages of correct predictions obtained after 5000 epochs.

### 1950-1990+1991

	$\alpha=0$	$\alpha=1$	$\alpha=3$	$\alpha=5$	$\alpha=10$	$\alpha=50$
Nh=0	16.00%	8.00%	16.00%	16.00%	16.00%	16.00%
Nh=1	0.00%	7.20%	0.00%	0.00%	0.00%	0.00%
Nh=2	22.90%	8.00%	17.80%	20.75%	27.95%	22.45%
Nh=3	16.75%	6.80%	16.00%	15.75%	18.00%	16.80%

### 1951-1991+1992

	$\alpha=0$	$\alpha=1$	$\alpha=3$	$\alpha=5$	$\alpha=10$	$\alpha=50$
Nh=0	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Nh=1	16.00%	0.00%	16.00%	16.00%	16.00%	16.00%
Nh=2	15.20%	0.00%	8.00%	14.45%	15.20%	16.45%
Nh=3	3.60%	0.00%	0.00%	4.00%	4.80%	3.60%

### 1952-1992+1993

	$\alpha=0$	$\alpha=1$	$\alpha=3$	$\alpha=5$	$\alpha=10$	$\alpha=50$
Nh=0	8.00%	8.00%	8.00%	8.00%	8.00%	8.00%
Nh=1	8.00%	8.40%	8.00%	8.00%	8.00%	8.00%
Nh=2	8.00%	8.00%	16.00%	8.85%	8.00%	8.85%
Nh=3	6.85%	8.00%	8.00%	6.00%	7.25%	6.00%

### 1953-1993+1994

	$\alpha=0$	$\alpha=1$	$\alpha=3$	$\alpha=5$	$\alpha=10$	$\alpha=50$
Nh=0	33.00%	25.00%	33.00%	33.00%	33.00%	33.00%
Nh=1	8.00%	25.00%	8.00%	8.00%	8.00%	8.00%
Nh=2	8.00%	25.00%	8.00%	7.60%	8.00%	8.00%
Nh=3	3.60%	25.00%	8.00%	8.40%	4.00%	5.20%

We notice that in both cases (**12+1**, **3+1**), the best percentages of good predictions were obtained for the series 1953-1993+1994. We think that this is due to the fact that 1994 is a relative normal hydrological year; in 1994, the average was  $19.2 \text{ m}^3/\text{sec}$ , smaller only with  $11.1\%$  than the average of 1950-1994 -  $21.6 \text{ m}^3/\text{sec}$ . The smaller values from 1991, 1992 and 1993 are due to hydrological anomalies. Thus, in 1991, the average flow ( $27.5 \text{ m}^3/\text{sec}$ ) was higher with  $27.3\%$  than the average of 1950-1994, in 1992, the average flow ( $15.5 \text{ m}^3/\text{sec}$ ) was smaller with  $29.2\%$  than the average of 1950-1994, in 1993 the average flow ( $14.9 \text{ m}^3/\text{sec}$ ) was smaller with  $31.0\%$  than the average of 1950-1994;

In the next two tables we present an average of the results obtained for the 4 series in both cases, **12+1**, and **3+1**. We think that these results are the most relevant for analyzing the performances of the forecasting models, since, as we saw, there are great differences from year to year. An average of the results obtained for the 4 years (1991, 1992, 1993, and 1994) is more concluding.

### **12+1**

	$\alpha=0$	$\alpha=1$	$\alpha=3$	$\alpha=5$	$\alpha=10$	$\alpha=50$
Nh=0	14.25%	20.25%	14.25%	14.25%	14.25%	14.25%
Nh=1	12.25%	22.50%	8.25%	12.25%	12.25%	12.25%
Nh=3	12.77%	20.25%	18.31%	15.16%	13.51%	13.48%
Nh=6	10.55%	20.25%	15.79%	13.96%	13.02%	13.90%
Nh=9	11.80%	20.25%	14.45%	12.51%	14.72%	12.10%

### **3+1**

	$\alpha=0$	$\alpha=1$	$\alpha=3$	$\alpha=5$	$\alpha=10$	$\alpha=50$
Nh=0	14.25%	10.25%	14.25%	14.25%	14.25%	14.25%
Nh=1	8.00%	10.15%	8.00%	8.00%	8.00%	8.00%
Nh=2	13.52%	10.25%	12.45%	12.91%	14.79%	13.94%
Nh=3	7.70%	9.95%	10.00%	8.54%	8.51%	7.90%

## **4. Conclusions**

The best percentages of correct predictions obtained are:

- 22.50% with **12+1**,  $Nh = 1$ ,  $\alpha = 1$  ;
- 20.25% with **12+1**,  $Nh = 0, 3, 6, 9$ ,  $\alpha = 1$  ;
- 18.31% with **12+1**,  $Nh = 3$ ,  $\alpha = 3$  ;
- 15.79% with **12+1**,  $Nh = 6$ ,  $\alpha = 3$  ;

Since we used 12 terms to test the forecasting performances of the networks, a percentage of 22.50% means that 2.7 (average of 20 runs) from the 12 predictions were correct. We notice that the best results were obtained in the case **12+1**. The better results which were obtained by means of 12 months' series in comparison of 3 months' series are the result of better homogeneousness of physical-geographical factors during 12 months than 3 months.

In previous papers, we used feed-forward neural networks and *SuperSAB* algorithm [1] and recurrent neural networks and *Backpropagation-through-time* algorithm [2] to predict the average monthly liquid flow at Borzii Vineți, with data from 1944-1989. But in these papers we considered only one time series with the data between 1944-1989 and we predicted the liquid flow from one month using the liquid flows from 12 previous months. The last 40 terms [1] and 50 terms [2] of the time series were used to test the forecasting performance of the models.

The best percentage of correct predictions obtained in [1] was 22.50%, as in the present paper, but the comparison is not relevant, since in [1] the reported results are the averages of 5 runs, while in the present paper they are the averages of 20 runs. The best result obtained in [2] is 17.20%, average of 20 runs, worse than the best result obtained in this paper.

We think that the approach from the present paper, considering 4 time series and taking the average of the results obtained for each of the time series and using monthly liquid flows from the last year to test the forecasting performance of the models, is better and more relevant.

In future work, we intend to make a comparison between our approach using neural networks and the classical predictions of these time series, realized using statistical methods, namely Pearson 3 methods. We hope that neural networks can be a promising alternative to classical prognosis and can contribute to hydrological prognosis improvement.

## ACKNOWLEDGEMENTS

This paper is part of INTAS Project no. 397: "Data Mining Technologies and Image Processing: Theory and Applications", Task 4: "The prognosis of harvest on the base of the weather- and geo-monitoring of a region".

## REFERENCES

1. L. Leuștean, Liquid flow time series prediction using feed-forward neural networks and SuperSAB learning algorithm. Conti'2002: 5th International Conference on Technical Informatics, 18-19 October 2002, Timișoara, Romania, Buletinul stiintific al Universitatii "Politehnica" din Timisoara, seria Automatica si Calculatoare, Tomul 47(61), No. 1, 2002, pp. 77-82.

2. L. Leuștean, Liquid flow prediction using recurrent neural networks and Backpropagation-through-time learning algorithm, 6th International Conference on Pattern Recognition and Image Analysis: New Information Technologies (PRIA-6-2002), Velikiy Novgorod, Russian Federation, October 21-26, 2002.
3. M. Riedmiller., Untersuchungen zu Konvergenz und Generalisierungsverhalten überwachter Lernverfahren mit dem SNNS, in A. Zell, editor, SNNS 1999 Workshop Proceedings, Stuttgart, September 1993.
4. M. Riedmiller, Advanced supervised learning in multi-layer perceptrons-from Backpropagation to adaptive learning algorithms, International Journal on Computer Standards and Interfaces, Vol. 16, 1994, pp. 265-278.
5. M. Riedmiller, Rprop-description and implementation details, Technical Report, University of Karlsruhe, January 1994.
6. M. Riedmiller, H. Braun, A direct adaptive method for faster backpropagation learning: The RPROP algorithm, in H. Ruspini, editor, Proceedings of the IEEE International Conference on Neural Networks (ICNN), San Francisco, USA, 1993, pp. 586-591.
7. D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, Parallel Distributed Processing: Explorations in the microstructure of cognition; Vol. 1: Foundations. The MIT Press, Cambridge, Massachusetts, 1986.
8. Z. Tang, P. Fishwick, Feed-forward neural nets as models for time series forecasting, Technical report, University of Florida, Gainesville, 1993.
9. L. Ujvary. Geografia apelor României. Scientific Ed., Bucharest, 1972 (in Romanian).
10. \*\*\*, SNNS-Stuttgart Neural Network Simulator, User Manual, Version 4.0, University of Stuttgart, Institute for Parallel and Distributed High Performance Systems, Technical Report 6/1995.
11. \*\*\*. Monografia hidrologică a bazinului hidrografic al râului Jiu. Studii de hidrologie XV. I. S. C. H., Bucharest, 1966 (in Romanian).
12. \*\*\*. Anualele hidrologice din perioada 1950-1994. I. N. M. H., Bucharest (in Romanian).