ARCHIVES OF ACOUSTICS
22, 1, 27–50 (1997)

APPLICATION OF ARTIFICIAL NEURAL NETWORKS TO THE RECOGNITION OF MUSICAL SOUNDS

B. KOSTEK and R. KRÓLIKOWSKI

Sound Engineering Department,
Faculty of Electronics, Telecommunications and Informatics,
Gdańsk University of Technology
(80-952 Gdańsk, ul. Narutowicza 11/12)

The aim of the presented work was to train a neural network in order to recognize a class of a chosen musical instrument. As problems related to analysis of sounds are related to human subjective perception abilities, then it seems that such tools of analyses as neural nets should be used for recognition processes. On the other hand, an artificial neural network should not be trained directly with subsequent samples of a sound, thus the feature extraction procedure is needed at first. As, there is no consensus regarding the selection of parameters for feature vector extraction, thus the experiment aimed to check whether calculated parameters are sufficient for creating a set of sound patterns used for neural network training. Some neural nets were investigated in the experiment, they were trained with so-called ELEVEN and FOURTEEN vector types. After the learning procedure was executed, other examples of the previously created database (but not seen by the neural network) were presented to neural nets. Results show that NNs (neural networks) are able to generalize information included in feature vectors. Therefore, when presenting data to NN inputs, there is no problem with variation of parameters within data, and consequently with data clustering, because a NN has the ability to generalize information during the learning phase. In the paper, an analysis of experimental results will be carried on, and conclusions derived from the performed tests will be presented.

1. Introduction

Despite the development of contemporary computer systems and growth in their processing power, there still exists a certain class of problems that have not been assigned the solving method. This class includes, among others, musical instrument sound recognition tasks. From the viewpoint of sound engineering the effectiveness of sound recognition is still imperfect even in well designed systems based on learning algorithms. Neural networks are one of the most frequently used learning algorithms. Systems that are based on these algorithms have become especially significant in the process of recognition of images, speech, also applications of musical sounds classifications have appeared [1, 2, 3]. It is the latter field that belongs to the most interesting aspects of musical acoustics.

One of the main advantages of artificial neural networks is the ability to generalize, that is the ability to correctly classify when the network comes across a new object at the input. The neural network processes the input object using the knowledge acquired

during the training phase. Therefore a neural network is applicable to solving all types of classification tasks and to recognition, that is difficult to be described algorithmically. The effectiveness of processing of input objects depends on the quality of the training phase progress. If the network is imprecisely taught, it may fail to learn how to generalize, hence its effectiveness in the recognition phase will be small. On the other hand, if the network gets taught too excessively it will lose the ability to generalize and will process correctly only elements of the training set. It is difficult to determine the moment at which the training of the network should be terminated as it depends on the character of the training set, the selected structure of the network, initial conditions, parameters of the network and the selected training method.

Artificial neural networks have found extensive application in many fields. Despite, however, a big number of various methods of training and the structures of the networks, the most often used are multi-layer networks of the *feedforward* type that are taught using the error back-propagation method (EBP) [4, 5].

The goal of this research work was to design a neural network for the purpose of classifying musical instruments patterns, and then to determine the effectiveness of its processing. The basis for the below research was a parametrized basis of musical sounds developed at the Sound Engineering Department of the Gdańsk University of Technology. In the article, out of necessity, the issues related to searching for parameters that would best represent distinctive features of sound of various classes of musical instruments were narrowed down to a presentation of parameters that were examined previously [6, 7, 8].

2. Architecture of the neural network – descripiton of the algorithm

2.1. Model of the artificial neuron

Each artificial neural cell consists of a processing element with $n + 1$ synaptic input connections attached to it and with a single output coming out of it. Additionally, one connection, a so-called threshold, is distinguished. Its input is permanently fed by value of -1.

The output signal of the neuron is given by the following relation:

$$o = f(\mathbf{w}^T\mathbf{x}), \tag{2.1}$$

where $\mathbf{w}$ is the vector of weights and $\mathbf{x}$ is the input vector. Because of the presence of the threshold, they are augmented by w_{n+1} and -1, respectively, and are defined as:

$$\mathbf{w} = \left[w_1\ w_2\ \ldots\ w_n\ w_{n+1}\right]^T, \qquad \mathbf{x} = \left[x_1\ x_2\ \ldots\ x_n\ -1\right]^T. \tag{2.2}$$

Function f in the formula (2.1) is referred to as a neuron activation function. Its domain is represented by the set of activation values expressed as [4]:

$$\text{net} = \mathbf{w}^T\mathbf{x}. \tag{2.3}$$

Since the error back-propagation method using the delta learning rule requires a differentiable function, that is why the commonly used activation functions are of sigmoidal

type (unipolar, bipolar, hyperbolic tangent, etc.) [4]. The unipolar continuous activation function defined in (2.4) is presented in Fig. 1.

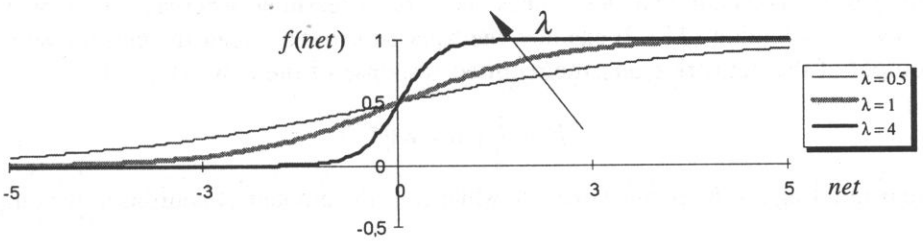

Fig. 1. Diagram of the unipolar function.

$$f(\text{net}) = \frac{1}{1 + \exp(-\lambda \cdot \text{net})}, \tag{2.4}$$

where net is given by the expression (2.2), whereas $\lambda > 0$ is proportional to the network gain and defines the steepness of the activation function. The function defined in formula (2.4) is very convenient to use since its derivative is expressed using simple expressions. Assuming the coefficient $\lambda = 1$, the derivative $f'(\text{net})$ adopts the form [4]:

$$f'(\text{net}) = f(1 - f). \tag{2.5}$$

2.2. Structure of the two-layer network of the feedforward type

The structure of the two-layer network of the *feedforward* type is one of the most commonly used structures. The consecutive layers are the input layer **x**, hidden layer **y** and output layer **o**. The number of neurons for the consecutive layers is respectively: $\mathbf{x}-I, \mathbf{y}-J, \mathbf{o}-K$. The input and hidden layers may have additionally one dummy neuron each. The output value of the neuron is constant and equals to -1, whereas the value of the weight may change. The dummy neuron is therefore an equivalent of the threshold synapse for all neurons in the next layer. Matrices $\mathbf{V}(J + 1 \times I + 1)$, $\mathbf{W}(K \times J + 1)$ are sets of synaptic weights respectively: from the input layer toward the hidden one and from the hidden toward the output layer.

2.3. Delta learning rule as the basis of the EBP algorithm

The error back-propagation method is based on the delta learning rule [4] which determines how the vector of weights should be updated in the successive step of the training. The increment of the vector of weights in the step $s + 1$ is expressed in the following way:

$$\mathbf{w}^{s+1} = \mathbf{w}^{s} + \Delta\mathbf{w}^{s+1}, \tag{2.6}$$

where s signifies the number of the training step.

In the course of training the increment in the weight vector $\Delta\mathbf{w}$ requires a change in the direction of the negative gradient in the error space. This is the general concept of

the delta learning rule [4]:

$$\Delta \mathbf{w}^{s+1} = -\eta \nabla E(\mathbf{w}^s), \tag{2.7}$$

where η is the constant that determines the rate of learning, whereas the error E is given by the definition (2.8). It signifies the squared error between the current value at the output of the network **o** and the required response of the network **d** [4]:

$$E = \frac{1}{2} \| \mathbf{d} - \mathbf{o} \|^2, \tag{2.8}$$

where **o** and **d** signify K-element vectors, while K is the number of neurons in the output layer.

2.4. Algorithm of the error back-propagation method

In order to simplify the notation of the error back-propagation method [4], it was adopted that the layers: output and hidden ones were extended by a neuron with a constant value at the input equal to -1. Therefore it was assumed that the number of neurons in the input layer equals I, in the hidden J, and the output one K.

Due to the above assumptions there are two matrices of weights $\mathbf{V}(J \times I)$, $\mathbf{W}(K \times J)$, whose values change in the course of the training phase:

$$\mathbf{V} = \begin{bmatrix} v_{11} & v_{12} & \dots & v_{1I} \\ v_{21} & v_{22} & \dots & v_{2I} \\ \dots & \dots & \dots & \dots \\ v_{JI} & v_{J2} & \dots & v_{JJ} \end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1J} \\ w_{21} & w_{22} & \dots & w_{2J} \\ \dots & \dots & \dots & \dots \\ w_{K1} & w_{K2} & \dots & w_{KJ} \end{bmatrix} \tag{2.9}$$

and 3 vectors $\mathbf{x}(I \times 1)$, $\mathbf{y}(J \times 1)$, $\mathbf{o}(K \times 1)$ denoted as the outputs of the particular layers:

$$\mathbf{x} = \begin{bmatrix} x_1 \, x_2 \, \dots \, x_{I-1} \, -1 \end{bmatrix}^T, \quad \mathbf{y} = \begin{bmatrix} y_1 \, y_2 \, \dots \, y_J \end{bmatrix}^T, \quad \mathbf{o} = \begin{bmatrix} o_1 \, o_2 \, \dots \, o_K \end{bmatrix}^T. \tag{2.10}$$

Considering the simplicity of the matrix notation of the EBP method, vectors: $\mathbf{f}'_o$, $\mathbf{f}'_y$ are defined. Their elements are derivatives of neuron activation function defined in the formula (2.5) and refer to neural nodes in the output layer and in the hidden one, respectively: layers **o** and **y**. The vectors: $\mathbf{f}'_o$, $\mathbf{f}'_y$ are as follows:

$$\begin{aligned} \mathbf{f}'_o &= \left[f'_1(\text{net}_1) \; f'_2(\text{net}_2) \; \dots \; f'_K(\text{net}_K) \right]^T, \\ \mathbf{f}'_y &= \left[f'_1(\text{net}_1) \; f'_2(\text{net}_2) \; \dots \; f'_J(\text{net}_J) \right]^T. \end{aligned} \tag{2.11}$$

Moreover, let linear diagonal operator $\mathbf{\Phi}[\mathbf{q}]$ and nonlinear one $\mathbf{\Gamma}[\mathbf{q}]$ be defined as below:

$$\mathbf{\Phi}[\mathbf{q}] = \begin{bmatrix} q_1 & 0 & \dots & 0 \\ 0 & q_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & q_Q \end{bmatrix}, \quad \mathbf{\Gamma}[\mathbf{q}] = \begin{bmatrix} f_1(q_1) & 0 & \dots & 0 \\ 0 & f_2(q_2) & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & f_q(q_Q) \end{bmatrix}, \tag{2.12}$$

where $f_1, f_2, \dots, f_Q$ – neuron activation functions as defined in the formula (2.4).

If the layers responses are as follows:

$$\mathbf{y} = \Gamma[\mathbf{V}\mathbf{x}], \qquad \mathbf{o} = \Gamma[\mathbf{W}\mathbf{y}] \tag{2.13}$$

then error signal terms for respective layers are defined:

$$\begin{aligned} \boldsymbol{\delta}_o &= -\nabla E(\mathbf{o}), \qquad \text{for the output layer } \mathbf{o}, \\ \boldsymbol{\delta}_y &= -\nabla E(\mathbf{y}), \qquad \text{for the hidden layer } \mathbf{y}. \end{aligned} \tag{2.14}$$

After having performed required calculations, the vectors for the error signal terms are expressed as follows:

$$\begin{aligned} \boldsymbol{\delta}_o &= \boldsymbol{\Phi}[\mathbf{d} - \mathbf{o}] \cdot \mathbf{f}'_o, \\ \boldsymbol{\delta}_y &= \mathbf{w}_j^T \cdot \boldsymbol{\delta}_o \cdot \mathbf{f}'_y, \end{aligned} \tag{2.15}$$

According to the delta learning rule (2.6)–(2.7), the update of weights **V**, **W** in the $k+1$-th step is calculated as in the formulae:

$$\begin{aligned} \mathbf{V}^{k+1} &= \mathbf{V}^k + \eta \boldsymbol{\delta}_y \mathbf{x}^T, \\ \mathbf{W}^{k+1} &= \mathbf{W}^k + \eta \boldsymbol{\delta}_o \mathbf{y}^T, \end{aligned} \tag{2.16}$$

In order to accelerate the convergence of the EBP training process, a momentum method is often applied by supplementing the current weight adjustment with a fraction of the most recent weight adjustment [4]. The momentum term (MT) in the $k+1$-th iteration is expressed by the relationship:

$$\mathrm{MT}^{k+1} = \alpha \cdot \Delta \mathbf{w}^k, \tag{2.17}$$

where α – user-defined positive momentum constant, typically from the range 0.1 to 0.8, $\Delta \mathbf{w}^k$ – increment of weights in the k-th step.

And thus, the final equations for the updates of matrices **V**, **W** with the momentum terms are computed as below:

$$\begin{aligned} \mathbf{V}^{k+1} &= \mathbf{V}^k + \eta \boldsymbol{\delta}_y \mathbf{x}^T + \alpha \cdot \Delta \mathbf{v}^k, \\ \mathbf{W}^{k+1} &= \mathbf{W}^k + \eta \boldsymbol{\delta}_o \mathbf{y}^T + \alpha \cdot \Delta \mathbf{w}^k. \end{aligned} \tag{2.18}$$

The dataflow of the EBP algorithm is diagrammed in Fig. 2.

The more detailed description of the algorithm from Fig. 2 is presented below:

Step 1 – weights of matrices **V**, **W** are initialized at small random values, which is recommended in the literature [1, 4]. In the majority of cases the variability of the weights values should range from -1 to 1.

Step 2 – cumulative cycle error E is set for 0. The goal of the training is to adjust the weights of the neural network in such a way that the value of the cumulative cycle error drops below the arbitrarily set value $E_{\max}$. Therefore parameter E is increased by the value computed using the expression (Eq. (2.8)) for each pattern from the training set.

Step 3 – an element is selected from the training set. It is recommended for vector **x** to be selected at random. At the same time the vector of required responses of network **d** gets updated.

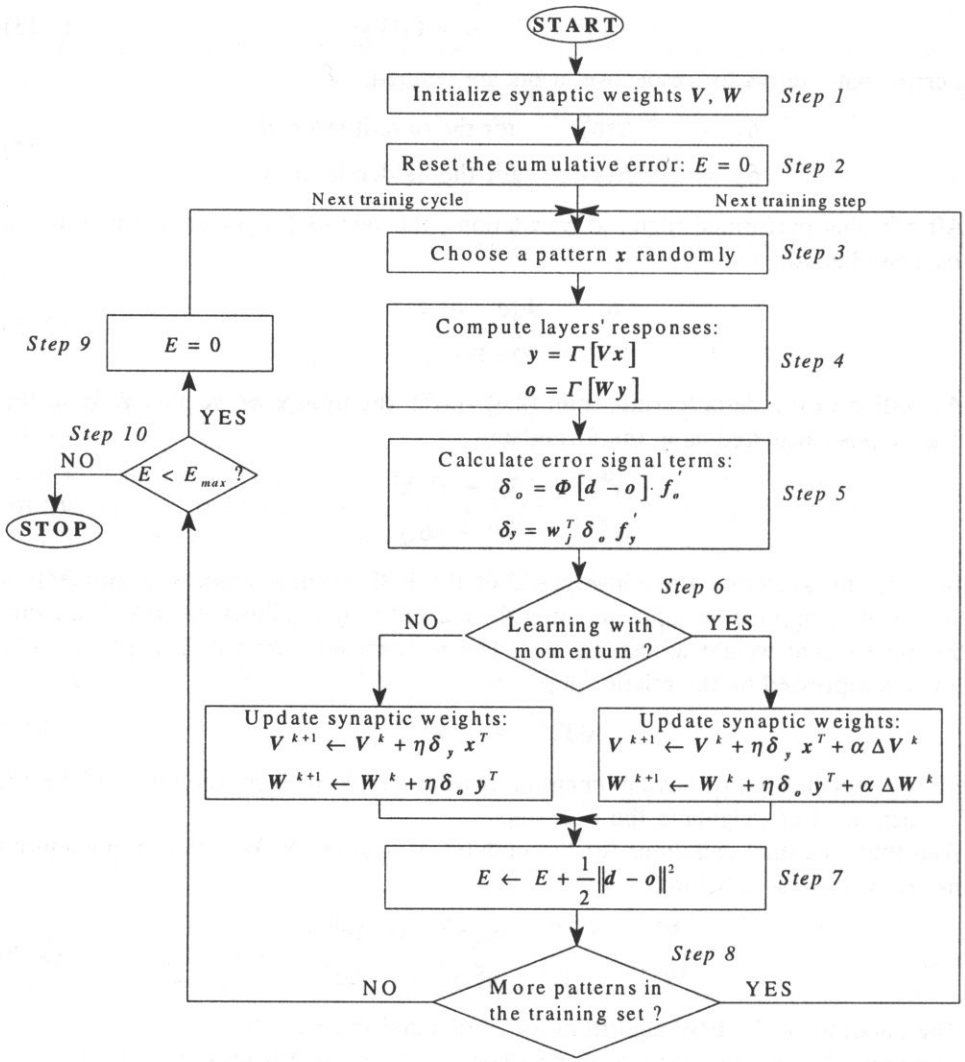

Fig. 2. Algorithm of the EBP method for a two-layer neural network.

Step 4 – responses of the particular layers are computed: **y**, **o**.

Step 5 – error signal terms are computed for the consecutive layers: δ_o, δ_y according to the equation (2.15).

Step 6 – if training process is performed with the momentum correction, matrices of weights **V**, **W** are updated based on the formula (2.16), otherwise on the equation (2.18).

Step 7 – error of the network is determined for the given pattern, whereas this value is added to the value of the cumulative error E.

Step 8 – if it is not the last pattern in the training set, then a consecutive object is selected at random and the processing goes back to *Step 3*. At the same time the object that was used is removed from the training set and does not take part further within one cycle of training.

Step 9 – in the contrary case if it is the last element in the training set, cumulative error E is compared to the condition of stop providing an arbitrarily assigned threshold value E_{max}. If the neural network processes all patterns in the training set with a satisfactory error ($E < E_{max}0$), the algorithm stops.

Step 10 – in the contrary case ($E \geq E_{max}$), one cycle of training is completed. The value of parameter E is reset to 0, the training set is reconstructed and another training cycle begins.

3. Experimental phase – training of the network

The goal of the experiments was to study the possibility of identifying selected classes of instruments by a neural network in order to verify the effectiveness of the extracted parameters of sounds. In the study a database, containing vectors of parameters of musical sounds that was prepared at the Sound Engineering Department of the Gdańsk University of Technology was used [6, 7]. The basis for the created database was digital recordings of musical instruments on CD records, released by McGill University Master Samples (MUMS) and partly the SHARC basis developed at the Sussex University [6]. However, as the SHARC database contains only FFT information of musical instrument sounds, therefore additional spectral parameters, such as brightness, energy of even and odd harmonics, the Tristimulus [9] parameters etc were calculated [6, 7, 8]. Additionally, some time-related parameters were extracted from sound patterns and added to the database.

The sounds of the instruments were represented by vectors of parameters. The training of the network and its testing was carried out on the basis of 2 types of vectors of parameters, called respectively: ELEVEN and FOURTEEN. The first one contained 11 parameters of steady-state and an initial transient, the latter additionally encompassed 3 parameters of steady-state on the basis of the SHARC database [6]. Since a stereo sound constituted the basis for calculating the parameters of musical sounds, so the same parameters were calculated separately for the left channel and the right one. Below formats of vectors and mathematical relations on the basis of which the parameters were calculated are shown.

Table 1. Formats of feature vectors, namely ELEVEN and FOURTEEN.

ELEVEN:

F	T_2	T_3	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8

FOURTEEN:

F	T_2	T_3	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	B	Od	Ev

where [8]

F – normalized frequency,

T_1 – energy of the first harmonic (the first modified Tristimulus parameter):

$$T_1 = A_1^2 \Big/ \sum_{n=1}^{N} A_n^2 , \tag{3.1}$$

where A_n – amplitude of the n-th harmonic, N – number of all available harmonics;

T_2 – the second modified Tristimulus parameter:

$$T_2 = \sum_{n=2}^{4} A_n^2 \Big/ \sum_{n=1}^{N} A_n^2 . \tag{3.2}$$

T_3 – the third modified Tristimulus parameter:

$$T_3 = \sum_{n=5}^{N} A_n^2 \Big/ \sum_{n=1}^{N} A_n^2 , \tag{3.3}$$

P_1 – rising time of the first harmonic expressed in periods of the signal,

P_2 – T_1 at the and of the attack divided by T_1 for the steady-state (so-called overshoot) (see Eq. (3.1)),

P_3 – rising time of the second, the third and the fourth harmonic expressed in periods of the signal,

P_4 – T_2 at the and of the attack divided by T_2 for the steady-state (see Eq. (3.2)),

P_5 – rising time of the remaining harmonics expressed in periods of the signal,

P_6 – T_3 at the and of the attack divided by T_3 for the steady-state (see Eq. (3.3)),

P_7 – delay of the second, the third and the fourth harmonic in relation to the first harmonic during the attack,

P_8 – delay of the remaining harmonics in relation to the first harmonic during the attack,

B – brightness,

$$B = \sum_{n=1}^{N} n \cdot A_n \Big/ \sum_{n=1}^{N} A_n , \tag{3.4}$$

Od – contents of odd harmonics without the first one in spectrum:

$$Od = \sqrt{\sum_{k=2}^{L} A_{2k-1}^2} \Big/ \sqrt{\sum_{n=1}^{N} A_n^2}, \tag{3.5}$$

where $L = \text{Entier}(N/2 + 1)$;

Ev – contents of even harmonics in spectrum:

$$Ev = \sqrt{\sum_{k=1}^{M} A_{2k}^2} \Big/ \sqrt{\sum_{n=1}^{N} A_n^2} , \tag{3.6}$$

where $M = \text{Entier}(N/2)$;

These parameters were normalized, i.e.:

$$T_1 + T_2 + T_3 = 1. \tag{3.7}$$

$$T_1 + Ev^2 + Od^2 = 1. \tag{3.8}$$

A multi-layer neural network of the *feedforward* type was used in the experiments. The number of neurons in the initial layer was equal to the number of elements of the parameters vector, hence it was respectively: 11 or 14. In turn, each neuron in the output layer was matched to a different class of the instrument and so their number was equal to the number of classes of instruments used in the experiment. The given network contained a hidden layer, too. The number of hidden neurons was arbitrarily adopted at 15. The first stage of the experiments encompassed the phase of training of the neural network. The training of the neural network was carried out using the error back-propagation method several times: 2 for the vector type ELEVEN, 3 – for the vector type FOURTEEN for the same training set. Each time different initial conditions were adopted as well as training parameters: the training process constant (η) and the momentum term (α) were changed dynamically in the course of the training. They were used later to evaluate the progress of the training process. Additionally the number of iterations was observed necessary to make the value of the cumulative error drop below the assumed threshold value.

To train the neural network, parameters vectors of 4 classes of instruments were selected: BASS TROMBONE, TROMBONE, ENGLISH HORN, CONTRA-BASSOON. In general, 2 types of training sets were formed: one encompassed all parameters vectors for the given channel (type ALL), while the other one contained about 70% of all vectors for the given (left or right) channel (type 70_PC). The vectors that were included in the set type 70_PC were chosen at random, however, it was attempted to maintain a uniform distribution. To make the results comparable, sets of type 70_PC consisted of vectors of the same indexes in the database, irrespective of the type of parameters vectors (ELEVEN or Fourteen). Below in Table 2 the number of parameters vectors for the given class of instruments in the training set type 70_PC in regard to the size of the class is shown. Additionally, this relation is presented in per cent, and also indexes of vectors that were excluded from the set type 70_PC are mentioned.

Table 2. Representation of the training set type 70_PC.

Instrument	Size of the class 70_PC	Size of the class 70_PC in [%]	Vectors excluded from the set type 70_PC
BASS TROMBONE	18/25	72%	2, 7, 10, 14, 18, 21, 23
TROMBONE	22/32	68.75%	1, 4, 7, 10, 15, 18, 22, 26, 29, 30
ENGLISH HORN	21/30	70%	3, 5, 8, 12, 16, 19, 22, 27, 30
CONTRA-BASSOON	22/32	68.75%	3, 6, 8, 11, 14, 19, 22, 25, 28, 31

The training set type ALL included 119 parameters vectors, while the set type 70_PC encompassed 83 vectors (69.75%).

For the selected phases of the training process, graphic presentations were made of dynamic changes of the parameters of training: η and α, respectively. Additionally,

the relation between the number of iterations and the maximum admissible value of cumulative cycle error is shown. The values of this error are at the same time the condition of discontinuing the process of training. The diagram of the above relation illustrates an increase in the number of required iterations along with an increase in accuracy of the training. Data matching this relation are presented in appropriate tables.

3.1. Training of the network on the basis of vectors type ELEVEN

In the case of training using vectors type ELEVEN, one training set was formed with parameters for the left channel. The set was type 70_PC. The training was continued up to the moment when the value of the cumulative error in the EBP method dropped below 0.005. This value was adopted arbitrarily in order to observe a possible case of network over-training. The diagram of the research is presented graphically in Fig. 3. The adopted descriptions have the following respective meanings: 70_PC – type of the training set, while variables range_V and range_W give information on the range of values of elements of matrix **V** and **W**. In the first case (**1**), matrices of network weights were initiated at random, however, these values did not exceed the range of $(-0.2, 0.2)$, while in the second case (**2**) this range decreased to values within $(-0.1, 0.1)$.

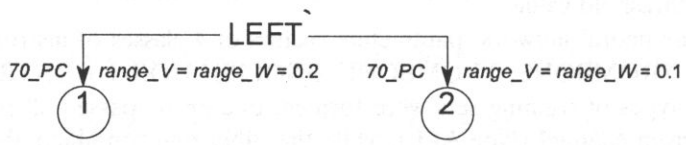

Fig. 3. Diagram of the training phase for the parameters of the left channel.

3.1.1. Training process – LEFT.1_70PC (ELEVEN). For this type of the training set (LEFT.1_70PC), the following initial conditions were adopted: unipolar activation function of the neuron, random initialization of values of elements of matrices **V** and **W** ranging from -0.2 to 0.2, training with the momentum method applied, $\eta = 0.6$, $\alpha = 0.4$. In Fig. 4 the dynamic change of training parameters is shown. Additionally, in Table 3 the convergence of the training phase is presented.

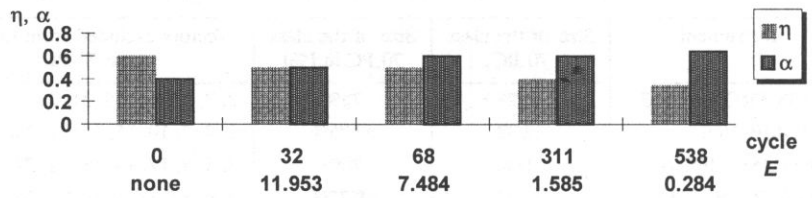

Fig. 4. Dynamic change of training parameters.

Table 3 lets one draw a conclusion that the network did not have any problems with training parameter vectors. Up till the value of the cumulative error 0.02 the training process proceeded quickly and relatively uniformly. Then the progress of training was

Table 3. Convergence of the training phase.

E_{max}	Number of iteration
0.05	714
0.04	783
0.03	897
0.02	1137
0.01	1902
0.005	3430

significantly curbed. It is worth noticing that all dynamic changes (see Fig. 4) of training parameters took place before the accuracy of training increased below 0.1.

3.1.2. Training process – LEFT.2_70PC (ELEVEN). For this type of the training set (LEFT.2_70PC), the same initial conditions were adopted as in the case of the training set type (LEFT.1_70PC). However, random values of elements of matrices **V** and **W** ranged from −0.1 to 0.1. In Fig. 5 the convergence of the training phase is presented. On the basis of diagram in Fig. 5 it is possible to see that up to the value of 0.05 of the cumulative error, the network training proceeded uniformly. Further growth in the required changes of values of weights should be small (because the maximum admissible error was small). It is also worth noticing that the last dynamic change of training parameters took place at error 0.1.

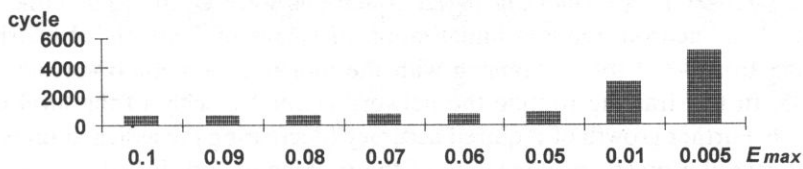

Fig. 5. Convergence of the training parameters.

3.2. Network training on the basis of vectors type FOURTEEN

For the purpose of training on the basis of vectors type FOURTEEN, 4 training sets were formed, 2 for each channel: the network was trained on all available vectors (100%) – type ALL and on about 70% (69.75%) of vectors from the database – type 70_PC. The training was proceeding up to the moment when the value of the cumulative error dropped below 0.01. This value was selected to be arbitrarily small so that a possible state of network over-training could be achieved. Twelve network training processes were conducted, 6 for each channel. A diagram of the training phase is presented in Fig. 6. The descriptors are analog to those that were presented in Fig. 3.

The ranges of random initialization of weights of matrices **V** and **W** for experiments marked as (1) and (2) are the same as for training sets type ELEVEN, as the objective was to observe the effects of initial conditions in the training process for both types of training sets. While training diagrams, marked as (3) (for both channels), have an

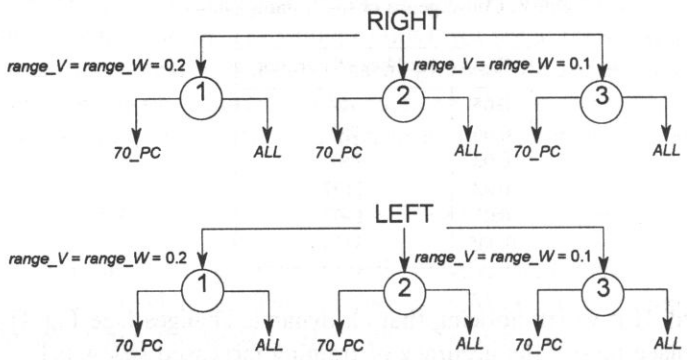

Fig. 6. Diagram of the training phase conducted for parameters of both channels: Left and Right.

identical range of random weight initialization as diagram (2). However, these routines differ from one another because each time the values of the weights during random initialization are different. The purpose of such diagrams (2) and (3) is to compare the process of training convergence within the same type of a training set.

The network training according to adopted training routines is shown in detail on the basis of the left channel.

3.2.1. Training process – LEFT.1_70PC (FOURTEEN). For this type of the training set (LEFT.1_70PC), the following initial conditions were adopted: unipolar activation function of the neuron, random initialization of values of elements of matrices **V** and **W** ranging from -0.2 to 0.2, training with the momentum method applied, $\eta = 0.05$, $\alpha = 0.45$. In this training routine the network learned quickly to the level of error at $0.07 - 0.06$. Further growth of required accuracy (decreasing the assigned threshold value of error) caused a drastic prolongation of the training period. It is due to a small value of the training coefficient. It is worth emphasising that in the proximity of the error value at 0.02 and 0.01 the term η was increased many times which caused the previously mentioned high error oscillations and finally attainment of required accuracy.

3.2.2. Training process – LEFT.1 ALL (FOURTEEN). The initial conditions were the same as in the case of training set (LEFT.1_70PC), however values were selected differently, namely: $\eta = 0.01$, $\alpha = 0.4$. The network training was proceeding very slowly. Starting from a maximum admissible error of the network at the level of 0.05, increasing the accuracy by 0.01 required additional 10 000 – 15 000 iterations. When the error generated by the network was equal to 0.0195, then the value η was increased from 0.004 to 0.03. Next η was being decreased gradually which caused the error to drop in consecutive iterations. This was happening at the expense of the speed of convergence. The end effect was a case when the training process stopped at a certain flat local minimum, which was observed in the proximity of error at 0.0101 – 0.01. That was why η was increased by 100 which caused rapid oscillations in the proximity of 0.01 and the result was that the training terminated with the final error below 0.01. The values of the η were adjusted at a relatively low level so as not to reduce the magnitude of the oscillations that arose.

3.2.3. Training process – LEFT.2_70PC (FOURTEEN). The following initial conditions were adopted in the case of the LEFT.2_70PC training set, namely: unipolar activation function of the neuron, random initialization of values of elements of weight matrices covering the range $(-0.1, 0.1)$, training with the momentum term, $\eta = 0.05$, $\alpha = 0.4$. In Fig. 7 the dynamic change of training parameters is presented. Additionally, in Fig. 8 the convergence of the training process is shown.

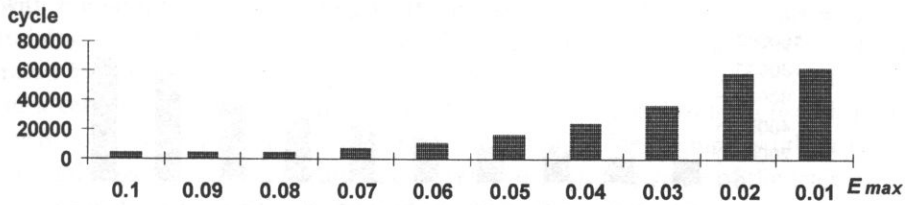

Fig. 7. Convergence of the training process.

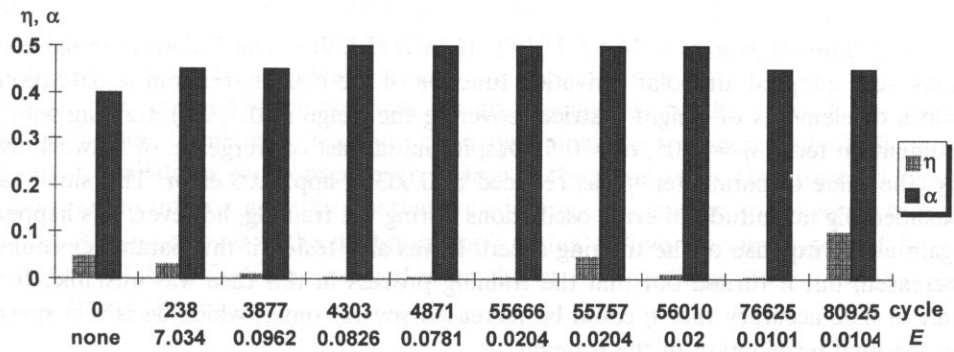

Fig. 8. Dynamic change of training parameters.

The data from Fig. 7 and 8 show clearly that the training process was sharply stopped because the value of admissible error was decreased below 0.05. Initially the training proceeding very rapidly and attained the assigned boundary error of 0.1 – 0.7 within only several thousand of iterations. As the accuracy of training was being increased, the number of necessary iterations was growing. It was due to the fact that the speed of training η was very low (~ 0.005) and at the same time the momentum term α was reaching a high value (~ 0.5). It can be observed that close to the value of the error at 0.02 the value of η was increased ten times to evoke higher error oscillations. The result was such as that after about 250 iterations the accuracy of the training dropped below 0.02. On the other hand, close to the value of 0.01 (0.0101) the speed of training was reduced twice ($0.01 \rightarrow 0.005$) in order to reduce the error generated, to go below the boundary value of 0.01. However, it did not succeed, the error generated increased and only by evoking higher error oscillations (η was increased twenty times) was the training terminated.

3.2.4. Training process – LEFT.2_ALL (FOURTEEN). In the training routine used next – LEFT.2_ALL the training was proceeding quickly and without interferences. The

following initial conditions were adopted: unipolar activation function of the neuron, random initialization of values of elements of weight matrices covering the range $(-0.1, 0.1)$, training with the momentum term, $\eta = 0.03$, $\alpha = 0.45$. Less than 10000 iterations were needed to obtain training accuracy below 0.03. What could be observed was that the training was paused at the admissible error of 0.02 and 0.01. However, such regularity was also present in other training routines.

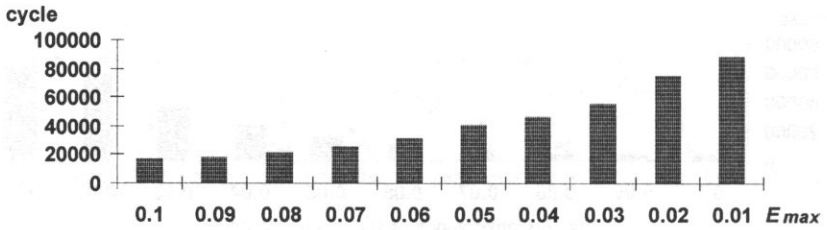

Fig. 9. Convergence of the training process.

3.2.5. Training process – LEFT.3_70PC (FOURTEEN). The following initial conditions were adopted: unipolar activation function of the neuron, random initialization of values of elements of weight matrices covering the range $(-0.1, 0.1)$, training with the momentum term, $\eta = 0.05$, $\alpha = 0.5$. Despite initial fast convergence of network training, the value of parameter η was reduced to 0.003 at app. 0.08 error. This small value excluded big magnitude of error oscillations during the training, however, this happened again at the expense of the training speed. It was also tested if this parameter could be increased, but it turned out that the training process in this case was unstable. It was only at 0.02 accuracy that η could be increased several times, which decisively speeded up the final termination of the training.

3.2.6. Training process – LEFT.3_ALL (FOURTEEN). In the case of the LEFT.3_ALL training set the following initial conditions were adopted: unipolar activation function of the neuron, random initialization of values of elements of weight matrices ranging from -0.1 to 0.1, training with the momentum term, $\eta = 0.05$, $\alpha = 0.45$. The training process was the longest of the experiments conducted. From the beginning the process was slowly converging and the assigned values $\eta(\sim 0.02)$ turned out to be too high. It is worth observing that the attempt to increase η from 0.002 to 0.003 (for iteration equal to 23865) did not succeed and therefore the network was learning with a training coefficient at 0.001. The changes introduced pertained only to the momentum term, responsible for damping of undesirable error oscillations (because as it would turn out – the network was in a very unstable state and had great difficulty in learning). Increasing the value of η happened close to the threshold error of value at 0.03. By adopting this value of η it was possible to speed up the training process.

3.3. Recapitulation of the training phase

It is possible to observe a huge difference in the training process between vectors type ELEVEN and FOURTEEN. In the first case the training was proceeding quickly,

uniformly and it reached the error value 0.01 after a relatively small number of iterations (3000 – 5000). Besides, dynamic changes of training parameters were not required often. Also the values of these parameters could be adjusted at a relatively high level, particularly the training speed term $\eta(\sim 0.3)$.

Training of vectors type FOURTEEN proceeded differently both for the left and right channel. The conclusions below pertain to network training for both channels in line with the adopted routines (Fig. 6). It can be observed that a quick convergence of the training process would take place at the beginning of the training until the error generated by the network was bigger from about 0.07 – 0.05. Next, the process would stop in a certain area of error space and within several or a dozen thousand iterations there was no improvement in the quality of the training. The training would usually end after several dozen thousand iterations (60 000 – 90 000). The value of the training speed term η had to be very small which additionally prolonged the training process, and on the average it was about 0.001 – 0.005. In some cases, increasing the value would cause a drastic increase in the error. At the same time quite frequently the dynamic parameters of training had to be changed. For this type of network, occasionally a training technique was applied which was to gradually reduce η in proximity of a certain boundary value of the error, the result of which was that the error generated by the network diminished below this value. Sometimes in similar cases the value of η was increased which caused high error oscillations and consequently a drop in the error below the threshold value.

What seems interesting is the fact that the addition of 3 elements to the vector brought about such a big change in the quality of the training. Network accuracy of 0.01 for the vector type FOURTEEN was reached after almost 10 – 20 times more iterations than for type ELEVEN.

4. Testing phase of the neural network

After the training phase was over, 14 neural networks were available: 2 were trained with vectors type ELEVEN and 12 with vectors type FOURTEEN. In the testing phase the purpose was to test the effectiveness relation of identifying new objects by the network as it relates to network training accuracy. The neuron whose output value was the highest showed the winning class and such was the classification done by the network. The number of correct and wrong responses expressed in per cent (pos/neg [%]) and in numbers (pos/neg) is presented in the recognition effectiveness table. This record encompasses recognition both in the particular classes as well as the total effectiveness of the network. Besides, the numbers of indexes (in the set type ALL) of these vectors are given, that were wrongly classified.

It is worth observing that the effectiveness of the network does not determine the quality of the trained network. Good quality can be understood as a feature of the network that causes that k-th neuron at the output will generate a high value in relation to the values of outputs of the remaining neurons (e.g. 0.8 to $\sim$ 0.005) for a given vector at the input, being a member of k-th class. In the case when the values at the remaining outputs are substantial in relation to k-th output, one can speak about a decreased quality of the trained network. In order to designate the recognition quality, all outputs

of neurons were observed when the vectors from the k-th class were being presented. The values of outputs of neurons were treated as deviations from the expected value of 0. Variance can then be a measure of the quality of the trained network. The bigger the variance calculated for particular outputs of neurons is, the stronger the classifications for particular classes are. This parameter is computed on the basis of the following formula:

$$\mathrm{Var}_k = \frac{1}{N_k} \sum_{i=1}^{N_k} o_i^2, \tag{4.1}$$

where Var_k – value of variance for k-th neuron, N_k – number of parameter vectors, members of the k-th class and not present in the training set, o_i – output of the k-th neuron for k-th parameter vector.

In the tables presented in the next paragraph the values of variances for the particular neurons are listed when the network was activated with vectors belonging to the consecutive classes. Distribution of variance of network outputs is given in relation to training accuracy. The purpose for that presentation was to check until what value of the threshold error the network could be taught to obtain the best quality of recognition for the given testing set. On the basis of the exemplary tests, the procedure of effectiveness verification of the given network will be shown.

4.1. Testing a network trained for parameters of the left channel (ELEVEN).

4.1.1. Network test – LEFT_30PC. The test was conducted on a set type LEFT_30PC. The tested network was type LEFT1_70PC. The recognition effectiveness for the chosen testing set is presented in Table 4.

Table 4. Recognition effectiveness.

$E_{\max}$	BASS TROMBONE		TROMBONE		ENGLISH HORN		CONTRA-BASSOON		Total score	
	pos/neg [%]	pos/ neg	pos/neg [%]	pos/ neg	pos/neg [%]	pos/ neg	pos/neg [%]	pos/ neg	pos/neg [%]	pos/ neg
0.1	100 / 0	7 / 0	90 / 10	9 / 1	100 / 0	9 / 0	70 / 30	7 / 3	88.89 / 11.11	32 / 4
0.09	100 / 0	7 / 0	90 / 10	9 / 1	100 / 0	9 / 0	70 / 30	7 / 3	88.89 / 11.11	32 / 4
0.07	100 / 0	7 / 0	90 / 10	9 / 1	100 / 0	9 / 0	70 / 30	7 / 3	88.89 / 11.11	32 / 4
0.05	100 / 0	7 / 0	90 / 10	9 / 1	100 / 0	9 / 0	80 / 20	8 / 2	91.67 / 8.33	33 / 3
0.01	100 / 0	7 / 0	90 / 10	9 / 1	100 / 0	9 / 0	80 / 20	8 / 2	91.67 / 8.33	33 / 3
0.005	100 / 0	7 / 0	90 / 10	9 / 1	100 / 0	9 / 0	80 / 20	8 / 2	91.67 / 8.33	33 / 3

Indexes of wrongly classified vectors are presented for this type of the testing set:
TROMBONE:
— 10; $E_{\max} = (0.1 - 0.005)$

CONTRA-BASSOON:

— 8, 19, 28; $E_{max} = (0.1 - 0.07)$

— 8, 19; $E_{max} = (0.06 - 0.005)$.

As is seen all vectors representing objects from BASS TROMBONE and ENGLISH HORN classes were recognized correctly.

For the purpose of presenting the values of variances, two classes of instruments were selected, namely: BASS TROMBONE (Tab. 5) and TROMBONE (Tab. 6). The first of these instruments was identified with better effectiveness than the other one.

Table 5. Variances in outputs of neurons upon presentation of vectors of the class BASS TROMBONE.

E_{max}	BASS TROMBONE	TROMBONE	ENGLISH HORN	CONTRA-BASSOON
0.1	0.9185582	0.0005653	0.0001944	0.0008058
0.09	0.9185272	0.0005191	0.0001844	0.0008058
0.07	0.9180164	0.0003908	0.0001581	0.0007576
0.05	0.9174483	0.0003162	0.0001296	0.0006543
0.01	0.9187201	0.0001693	0.0000340	0.0001537
0.005	0.9163189	0.0001222	0.0000174	0.0000873

Table 6. Variances in outputs of neurons upon presentation of vectors of the class TROMBONE.

E_{max}	BASS TROMBONE	TROMBONE	ENGLISH HORN	CONTRA-BASSOON
0.1	0.0000060	0.8361880	0.0005759	0.1158222
0.09	0.0000054	0.8387308	0.0005390	0.1175816
0.08	0.0000045	0.8412930	0.0004921	0.1194312
0.07	0.0000043	0.8392953	0.0004291	0.1249455
0.06	0.0000041	0.8412675	0.0003830	0.1277956
0.05	0.0000034	0.8437424	0.0003230	0.1324253
0.01	0.0000013	0.8663360	0.0000854	0.1481672
0.005	0.0000007	0.8721313	0.0000494	0.1511610

The recognition effectiveness increase with changing the accuracy of the network from 0.07 to 0.06. Despite a further increase in accuracy up to the value $E_{max} = 0.005$, the effectiveness remained at the same level. It is to say that the best recognition results were obtained at $0.06 - 0.05$ of the cumulative error E_{max}.

4.1.2. Network test – LEFT_30PC. The test was conducted also on a set type LEFT_30PC, but in this case the tested network was type LEFT2_70PC. The recognition effectiveness for the chosen testing set is presented in Table 7.

Indexes of wrongly classified vectors are presented for this type of the testing set:

TROMBONE:

— 10, 18; $E_{max} = (0.05 - 0.005)$

CONTRA-BASSOON:

— 8; $E_{max} = (0.05 - 0.005)$.

Table 7. Recognition effectiveness.

E_{max}	BASS TROMBONE		TROMBONE		ENGLISH HORN		CONTRA-BASSOON		Total score	
	pos/neg [%]	pos/ neg	pos/neg [%]	pos/ neg	pos/neg [%]	pos/ neg	pos/neg [%]	pos/ neg	pos/neg [%]	pos/ neg
0.05	100 / 0	7 / 0	80 / 20	8 / 2	100 / 0	9 / 0	90 / 10	9 / 1	91.67 / 8.33	33 / 3
0.04	100 / 0	7 / 0	80 / 20	8 / 2	100 / 0	9 / 0	90 / 10	9 / 1	91.67 / 8.33	33 / 3
0.03	100 / 0	7 / 0	80 / 20	8 / 2	100 / 0	9 / 0	90 / 10	9 / 1	91.67 / 8.33	33 / 3
0.02	100 / 0	7 / 0	80 / 20	8 / 2	100 / 0	9 / 0	90 / 10	9 / 1	91.67 / 8.33	33 / 3
0.01	100 / 0	7 / 0	80 / 20	8 / 2	100 / 0	9 / 0	90 / 10	9 / 1	91.67 / 8.33	33 / 3
0.005	100 / 0	7 / 0	80 / 20	8 / 2	100 / 0	9 / 0	90 / 10	9 / 1	91.67 / 8.33	33 / 3

The per cent of correctly qualified vectors was equal to 91.67% for all values of the cumulative error. It is to say also that the recognition effectiveness increase with the growing accuracy of the network.

4.2. Testing a network trained for parameters of the left channel (FOURTEEN)

4.2.1. Network test – LEFT_30PC. The test was conducted on a set type LEFT_30PC. The tested network was type LEFT1_70PC. The recognition effectiveness for the chosen testing set is presented in Table 8.

Table 8. Recognition effectiveness.

E_{max}	BASS TROMBONE		TROMBONE		ENGLISH HORN		CONTRA-BASSOON		Total score	
	pos/neg [%]	pos/ neg	pos/neg [%]	pos/ neg	pos/neg [%]	pos/ neg	pos/neg [%]	pos/ neg	pos/neg [%]	pos/ neg
0.1	100 / 0	7 / 0	70 / 30	7 / 3	100 / 0	9 / 0	20 / 80	2 / 8	66.44 / 30.56	25 / 11
0.09	100 / 0	7 / 0	100 / 0	10 / 0	100 / 0	9 / 0	90 / 10	9 / 1	97.22 / 2.78	35 / 1
0.07	100 / 0	7 / 0	100 / 0	10 / 0	100 / 0	9 / 0	90 / 10	9 / 1	97.22 / 2.78	35 / 1
0.05	100 / 0	7 / 0	100 / 0	10 / 0	100 / 0	9 / 0	90 / 10	9 / 1	97.22 / 2.78	35 / 1
0.03	100 / 0	7 / 0	100 / 0	10 / 0	100 / 0	9 / 0	9 / 10	9 / 1	97.22 / 2.78	35 / 1
0.01	100 / 0	7 / 0	100 / 0	10 / 0	100 / 0	9 / 0	90 / 10	9 / 1	97.22 / 2.78	35 / 1

Below, indexes of wrongly classified vectors are presented for this type of the testing set:

TROMBONE:

— 1, 29, 30; $E_{max} = 0.1$

CONTRA-BASSOON:

— 3, 8, 11, 14, 22, 25, 28, 31; $E_{max} = 0.1$

— 8; $E_{max} = (0.09 - 0.01)$.

For the purpose of presenting the values of variances, two classes of instruments were selected, namely: BASS TROMBONE (Tab. 9) and TROMBONE (Tab. 10). The first of these instruments was identified with much better effectiveness than the other one.

Table 9. Variances in outputs of neurons upon presentation of vectors of the class BASS TROMBONE.

E_{max}	BASS TROMBONE	TROMBONE	ENGLISH HORN	CONTRA-BASSOON
0.1	0.3597468	0.0060959	0.0026676	0.0606067
0.09	0.9647862	0.0027156	0.0000044	0.0003433
0.07	0.9672036	0.0025964	0.0000038	0.0002916
0.05	0.9721547	0.0023566	0.0000030	0.0002217
0.03	0.9778430	0.0022333	0.0000021	0.0001452
0.01	0.9856030	0.0032118	0.0000008	0.0000564

Table 10. Variances in outputs of neurons upon presentation of vectors of the class TROMBONE.

E_{max}	BASS TROMBONE	TROMBONE	ENGLISH HORN	CONTRA-BASSOON
0.1	0.0059164	0.3968045	0.0432925	0.0353321
0.09	0.0000000	0.8179480	0.0014706	0.0222385
0.07	0.0000000	0.8193167	0.0016362	0.0204443
0.05	0.0000000	0.8198924	0.0019054	0.0180781
0.03	0.0000000	0.8206997	0.0022593	0.0148136
0.01	0.0000000	0.8223629	0.0029912	0.0090919

The visible change of recognition effectiveness happened upon changing the accuracy of the network from 0.1 to 0.09. Despite a further increase in accuracy, the effectiveness remained at the same level – 97.22%, that is only one vector was wrongly classified. Upon the presentation of the vectors of the particular classes it can be observed that the quality of identifying new objects was slightly growing together with a reduction in the cumulative error E_{max}: variance in values increased at the output of the neuron which represented the given class, while at the other outputs usually a drop in the variance is observed.

4.2.2. Network test – RIGHT_30PC. The test was conducted on a set type RIGHT_30PC. The tested network was type LEFT1_70PC. The recognition effectiveness for the chosen testing set is presented in Table 11.

Table 11. Recognition effectiveness.

E_{max}	BASS TROMBONE		TROMBONE		ENGLISH HORN		CONTRA-BASSOON		Total score	
	pos/neg [%]	pos/ neg	pos/neg [%]	pos/ neg	pos/neg [%]	pos/ neg	pos/neg [%]	pos/ neg	pos/neg [%]	pos/ neg
0.1	88 / 12	22 / 3	78.12 / 21.88	25 / 7	100 / 0	30 / 0	40.62 / 59.38	13 / 19	75.63 / 24.37	90 / 29
0.09	100 / 0	25 / 0	100 / 0	32 / 0	100 / 0	30 / 0	96.88 / 3.12	31 / 1	99.16 / 0.84	118 / 1
0.07	100 / 0	25 / 0	100 / 0	32 / 0	100 / 0	30 / 0	96.88 / 3.12	31 / 1	99.16 / 0.84	118 / 1
0.05	100 / 0	25 / 0	100 / 0	32 / 0	100 / 0	30 / 0	96.88 / 3.12	31 / 1	99.16 / 0.84	118 / 1
0.03	100 / 0	25 / 0	100 / 0	32 / 0	100 / 0	30 / 0	96.88 / 3.12	31 / 1	99.16 / 0.84	118 / 1
0.01	100 / 0	25 / 0	96.88 / 3.12	31 / 1	100 / 0	30 / 0	96.88 / 3.12	31 / 1	98.32 / 1.68	117 / 2

Below, indexes of wrongly classified vectors are presented for this type of the testing set:

BASS TROMBONE:

— 8, 9, 16; $E_{max} = 0.1$

TROMBONE:

— 1, 2, 3, 4, 9, 29, 30; $E_{max} = 0.1$

CONTRA-BASSOON:

— 1, 3, 10, 11, 14 – 17, 21 – 29, 31, 32; $E_{max} = 0.1$

— 3; $E_{max} = (0.09 - 0.01)$.

The visible change of recognition effectiveness happened upon changing the accuracy of the network from 0.1 to 0.09. In that case, 12 vectors of the CONTRA-BASSOON class, previously wrongly recognized, were classified correctly. It is also possible to observe the case of the network over-training, when the cumulative error E_{max} was reduced from 0.02 to 0.01. In consequence, one vector of the TROMBONE class was wrongly classified.

4.2.3. Network test – RIGHT_ALL. The test was conducted on a set type RIGHT_ALL. The tested network was type LEFT1_ALL. The recognition effectiveness in this case was very good. All vectors from the testing set were recognized correctly by the network.

4.3. Recapitulation of the testing phase

The best scores of recognition effectiveness for the particular training routines of the test were compiled in Table 12 and 13 (separately for sets of type FOURTEEN and ELEVEN). The consecutive columns signify: the test routine (name of the training and testing set), classification effectiveness expressed in per cent and numbers and respective values of E_{max}.

Table 12. Compilation of the best classifications for sets type FOURTEEN.

Test routine	Testing set	Classification effectiveness		E_{max}
		pos/neg [%]	pos/neg	
RIGHT.1_70PC	RIGHT_30PC	**91.67** / 8.33	33 / 3	(0.1 – 0.01)
	LEFT	**95.80** / 4.20	114 / 5	0.08; 0.07; 0.01
RIGHT.1_ALL	LEFT_ALL	**99.16** / 0.84	118 / 1	(0.1 – 0.01)
RIGHT.2_70PC	RIGHT_30PC	**97.22** / 2.78	35 / 1	(0.1 – 0.08); (0.06 – 0.01)
	LEFT_ALL	**98.32** / 1.68	117 / 2	0.09; 0.08; 0.06; (0.04 – 0.01)
RIGHT.2_ALL	LEFT_ALL	**99.16** / 0.84	118 / 1	(0.1 – 0.01)
RIGHT.3_70PC	RIGHT_30PC	**91.67** / 8.33	33 / 3	(0.03 – 0.01)
	LEFT_ALL	**96.64** / 3.36	115 / 4	(0.06 – 0.01)
RIGHT.3_ALL	LEFT_ALL	**98.32** / 1.68	117 / 2	(0.1 – 0.01)
LEFT.1_70PC	LEFT_30PC	**97.22** / 2.78	35 / 1	(0.09 – 0.01)
	RIGHT_ALL	**99.16** / 0.84	118 / 1	(0.09 – 0.02)
LEFT.1_ALL	RIGHT_ALL	**100** / 0	119 / 0	(0.1 – 0.01)
LEFT.2_70PC	LEFT_30PC	**97.22** / 2.78	35 / 1	0.02; 0.01
	RIGHT_ALL	**98.32** / 1.68	117 / 2	0.1; 0.09
LEFT.2_ALL	RIGHT_ALL	**100** / 0	119 / 0	(0.1 – 0.01)
LEFT.3_70PC	LEFT_30PC	**94.44** / 5.56	34 / 2	(0.1 – 0.01)
	RIGHT_ALL	**98.32** / 1.68	117 / 2	(0.1 – 0.07)
LEFT.3_ALL	RIGHT_ALL	**100** / 0	119 / 0	(0.1 – 0.01)

Tables (Table 12 and 13) show that recognition effectiveness during the experiments was very high and was always above 90%. Since the testing was done on sets with 36 and 119 elements, these results can be divided into two classes: for the first type of sets, the score of correct classification ranged from 91.67% to 97.22%, but the number

Table 13. Compilation of the best classifications for sets type ELEVEN.

Test routine	Testing set	Classification effectiveness		E_{max}
		pos/neg [%]	pos/neg	
LEFT.1_70PC	LEFT_30PC	**91.67** / 8.33	33 / 3	(0.06 – 0.005)
LEFT.2_70PC	LEFT_30PC	**91.67** / 8.33	33 / 3	(0.05 – 0.005)

of unrecognized vectors was from 1 to 3. And then for the set with 119 vectors the effectiveness of the classification was at the level from 95.80% to 100% where the number of wrongly classified vectors ranged from 5 to 0.

Another fact worth observing is that the change of training vectors from the type ELEVEN into FOURTEEN contributed to an increase in effectiveness of the classification. For vectors type ELEVEN the network attained the recognition score at 91.67%, while extension of the vector up to fourteen parameters caused the score to reach the value of 94.44% – 97.22%.

In all cases a network trained with the set type 70_PC and tested with a set type ALL (with data for the second channel) would give worse results than the network trained on a set type ALL and tested on ALL, too (also with data for the second channel). It is to observe that unrecognized objects amounted to app. 1 – 2%, and in the worst case – app. 4.5%. At the same time it is possible to see that the number of unrecognized objects is bigger if the network was trained in parameter vectors of the right channel (RIGHT) than the left one (LEFT) where the biggest variance amounted to app. 1.5%. One can therefore draw a conclusion that a 30% increase in the training set only relatively increased the effectiveness of classification.

It is visible that a slightly better effectiveness was observed in networks that were trained with vectors type LEFT than RIGHT. On this basis it is possible to conclude that a subspace consisted of vectors type LEFT overlaps to a large extent the subspace spanned by vectors type RIGHT enclosed in the object space, with the first subspace giving better generalization possibilities.

It is possible to see that the highest effectiveness was found in networks whose accuracy was close to 0.03 – 0.01. On the other hand, however, there was also a big number of networks that reached their best recognition at accuracy at 0.1. Tables 12 – 13 do not consider, however, the quality of recognition which was characterized by values of variances at the network output. Analyzing these values of variances it is possible to assume that the networks were best at classifying when the value of the error E_{max} was below 0.05.

Table 14. Numbers of wrongly classified vectors (TROMBONE).

Training set	Testing test	Vector number	E_{max}
LEFT.1_70PC	LEFT_30PC	10	(0.1 – 0.005)
LEFT.2_70PC	LEFT_30PC	10, 18	(0.05 – 0.005)

Additionally, Tables 14 – 15 give numbers of those vectors that were wrongly classified for the particular classes of instruments and test routines limited to training vectors type ELEVEN. Additionally, the tables show the maximum value of error $E_{\max}$ that was achieved in the course of the training. It also needs to be noted that all vectors of classes BASS TROMBONE and ENGLISH HORN were correctly identified.

Table 15. Numbers of wrongly classified vectors (CONTRA-BASSOON)

Training set	Testing test	Vector number	$E_{\max}$
LEFT.1_70PC	LEFT_30PC	8, 19, 28 8, 19	(0.1 – 0.005) (0.06 – 0.005)
LEFT.2_70PC	LEFT_30PC	8	(0.05 – 0.005)

5. Conclusions

The result of the experiments conducted shows a high effectiveness of classification of musical instruments by neural networks. The results obtained show that only in a dozen of experiments (with various initial parameters of the training) per several hundred total some vectors were not correctly identified. Hence the presumption that data in these very vectors may be incorrectly acquired. It is to remember that parametrized signals were sounds recorded in real conditions, i.e. a free way of a musical performance. Therefore phenomena such as musical articulation or differentiated dynamic with all features specific for an individual musician are included in the signal and resulted in signal modulation, amplitude overshoots, etc. That may cause in some cases a certain kind of "non-adaptation" to the engineered algorithms in which only three models of the relation between *Attack-Decay-Sustain* phases in a sound were assumed. What becomes evident is a way of testing the correctness of parametrization, if for a statistically big number of examined networks, the wrongly classified vectors are always the same, then it is these vectors that should be subjected to verification.

What seems interesting is the relation between the results obtained in the testing and the course of the training phase. It was possible to observe that the network trained with 14 element training sets: RIGHT.1_70PC and RIGHT.3_70PC obtained the worst results. The training of these networks was not easy, it required many changes of values of raining parameters – the coefficient of training speed (η) and the momentum term (α). At the same time the process of training did last for very long. On the other hand, when the training phase was short or there were no interferences in the course of the training process, the results obtained by the network in the test were significantly better. Presumably it is due to the fact that the network failed to acquire the ability to correctly classify indefinite cases. High oscillations of error in the course of the training could have caused relatively slight changes in the values of input vectors to bring about a wrong classification. Then the network would lose its ability to generalize. There were also cases (training on some of the LEFT sets) when despite a long training phase, the

caused a gradual increment in the values of weights. The network had a very high grade of generalization which eventually provided effectiveness even at the level of 100%.

The research conducted shows that the neural network performs well the task of identifying classes of musical instruments. The obvious advantage of this type of classifier is the fact that there is no need for quantization of values of parameters included in the vector which describes the musical sound. There is no doubt that a certain disadvantage of this type of testing is a huge amount of work needed to complete the training phase. Further research will focus on testing the effectiveness of a constructed classifier in terms of identifying other musical instruments. For that purpose in the base that was constructed at the Sound Engineering Department, Gdańsk University of Technology [6, 7] sets of feature vectors were included that describe sounds of musical instruments which belong to other groups. The usefulness of an artificial neural network for this type of applications seems all the bigger as the feature vectors included in the database encompass representations of consecutive sounds in the chromatic scale. In this case a high instability of designated parameters is observed, the additional element which affects the lack of stability of parameters is the presence of non-linearity related to differentiated articulations and dynamics of musical sounds. However, in both cases the network ability to generalize allows a correct classification of the objects being under the test.

Acknowledgments

The presented research project was sponsored by the Committee for Scientific Research, Warsaw, Poland Grant No. 8T11C02808.

References

[1] R. TADEUSIEWICZ, *Neural Nets*, [in Polish], Academic Printing Office RM, Warsaw 1993.

[2] J. MOURJOPOULOS and D. TSOUKALAS, *Neural Network Mapping to Subjective Spectra of Music Sounds*, 90th AES Conv., Preprint 3064, Paris 1991, J. Audio Eng. Soc. (Abstr), **39**, 5 (1991).

[3] B. KOSTEK, *Application des reseaux de neurones pour l'analyse de l'articulation musicale*, 3 CFA, 1994, Journal de Physique, **4**, 5 (1994).

[4] J. ZURADA J., *Artificial Neural Systems*, West Publishing Company, St. Paul 1992.

[5] B.K. BOSE, *Expert System, Fuzzy Logic, and Neural Network Applications in Power Electronics and Motion Control*, Proc. IEEE, **82**, 8, 100–114 (1994).

[6] B. KOSTEK, *Feature Extraction Methods for the Intelligent Processing of Musical Signals*, 99th AES Conv., Preprint 4076, New York 1995, J. Audio Eng. Soc. (Abstr), **43**, 12 (1995).

[7] B. KOSTEK and A. WIECZORKOWSKA, *Study of Parameter Relations in Musical Instrument Patterns*, 100th Audio Eng. Soc. Conv., Preprint No. 4173, Copenhagen (1996).

[8] B. KOSTEK and A. WIECZORKOWSKA, *Parametric representation of musical sounds*, Archives of Acoustics, **22**, 1, 3–26 (1997).

[9] H.F. POLLARD and E.V. JANSSON, *A Tristimulus method for the specification of musical timbre*, Acustica, **51** (1982).