Multilayer Neural Network based on Multi-Valued Neurons and the Blur Identification Problem

Igor Aizenberg, Member, IEEE, Dmitriy Paliy, and Jaakko T. Astola, Fellow, IEEE

Abstract — A multilayer neural network based on multi-valued neurons (MLMVN) is a neural network with a traditional feedforward architecture. At the same time this network has a number of specific properties and advantages. Its backpropagation learning algorithm does not require differentiability of the activation function. The functionality of MLMVN is higher than the ones of the traditional feedforward neural networks and a variety of kernel-based networks. Its higher flexibility and faster adaptation to the mapping implemented make possible an accomplishment of complex problems using a simpler network. The MLMVN can be used to solve those non-standard recognition and classification problems that cannot be solved using other techniques. In this paper we use the MLMVN as a tool for the blur identification problem. A prior knowledge about the distorting operator and its parameter is of crucial importance in blurred image restoration.

I. INTRODUCTION

A multilayer neural network based on multi-valued neurons (MLMVN) has been introduced in [1] and then investigated and developed further in [2]. This network consists of multi-valued neurons (MVN). That is a neuron with complex-valued weights and an activation function, defined as a function of the argument of a weighted sum. This activation function was proposed in 1971 in the pioneer paper of N. Aizenberg et al. [3].

The multi-valued neuron was introduced in [4]. It is based on the principles of multiple-value logical circuit over the field of the complex numbers formulated in [5] and then developed in [6]. A comprehensive observation of the discrete-valued MVN, its properties and learning is presented in [6]. A continuous-valued MVN and its learning are considered in [1],[2]. The most important properties of MVN are: the complex-valued weights, inputs and output coded by the $\mathbb{A}^n$ roots of unity (a discrete-valued MVN) or lying on the unit circle (a continuous-valued MVN), and the activation function, which maps the complex plane into the unit circle. It is important that MVN learning is reduced to the movement along the unit circle. The MVN learning algorithm is based on a simple linear error correction rule and it does not require differentiability of the activation function.

Different applications of MVN have been considered during recent years, e.g.: MVN as a basic neuron in the cellular neural networks [6], as the basic neuron of the neural-based associative memories [6],[7]-[10], as the basic neuron in a variety of pattern recognition systems [10]-[12], and as a basic neuron of the MLMVN [1],[2]. MLMVN outperforms a classical multilayer feedforward network and different kernel-based networks in the terms of learning speed, network complexity, and classification/prediction rate tested for such popular benchmarks problems as the parity $n$, the two spirals, the sonar, and the Mackey-Glass time series prediction [1],[2]. These properties of MLMVN show that it is more flexible and adapts faster in comparison with other solutions. In this paper we apply MLMVN to identify blur and its parameters, which is a key problem in image restoration.

Usually blur refers to the low-pass distortions introduced into an image. It can be caused, e.g., by the relative motion between the camera and the original scene, by the optical system which is out of focus, by atmospheric turbulence (optical satellite imaging), aberrations in the optical system, etc. [13]. Any type of blur, which is spatially invariant, can be expressed by the convolution kernel in the integral equation [14],[15]. Hence, restoration (deblurring) of a blurred image is an ill-posed inverse problem [16], and regularization is commonly used when solving this problem [16].

There is a variety of sophisticated and efficient deblurring techniques such as deconvolution based on the Wiener filter [13],[17], nonparametric image deblurring using local polynomial approximation with spatially-adaptive scale selection based on the intersection of confidence intervals rule [17], Fourier-wavelet regularized deconvolution [18], expectation-maximization algorithm for wavelet-based image deconvolution [19], etc. All these techniques assume a prior knowledge of the blurring kernel, characterized by the point spread function (PSF), and its parameter.

When the blurring operator is unknown, the image restoration becomes a blind deconvolution problem [20]-[22]. Most of the methods to solve it are iterative, and, therefore, they are computationally costly. Due to the presence of noise they suffer from the stability and convergence problems [23].

The original solution of blur identification problem that is based on the use of MVN-based neural networks was proposed in [12] and [24]. Any blur specifically distorts the
Fourier amplitude spectrum of an image. Quarter of the amplitude spectrum coefficients have been used as the features. Two different single-layer MVN-based networks have been used to identify blur and its parameter (e.g., variation for the Gaussian blur, extent for motion blur, etc.). The results were good, but this approach had some disadvantages. For instance, the networks used have specific architecture with no universal learning algorithm, thus each neuron was trained separately. Another disadvantage is the use of too many spectral coefficients as features (quarter of them). Thus the learning process was heavy.

In this paper, a single MLMVN is applied to solve both the blur and its parameter identification problems in order to overcome the disadvantages mentioned above. The MLMVN and its backpropagation learning algorithm are described in Section II. The image restoration and the blur identification problems are briefly discussed in Section III. Finally, the simulation aspects and results are presented in Section IV.

II. MULTILAYER NEURAL NETWORK BASED ON MULTI-VALUED NEURONS

A. Multi-Valued Neuron

MVN was introduced in [4],[5] as a neural element based on the principles of multiple-valued threshold logic over the field of complex numbers, and then deeply considered in [6], where its theory, basic properties, and learning are comprehensively observed. A single discrete-valued MVN performs a mapping between $n$ inputs and a single output. This mapping is described by a multiple-valued ($K$-valued) function of $n$ variables $f(x_1, ..., x_n)$ with $n+1$ complex-valued weights as parameters:

$$ f(x_1, ..., x_n) = P(w_0 + w_1 x_1 + ... + w_n x_n), \quad (1) $$

where $X = (x_1, ..., x_n)$ is a vector of inputs (a pattern vector) and $W = (w_0, w_1, ..., w_n)$ is a weighting vector. The function and variables are the $K$th roots of unity: $e^{i(pi/j)/K}, \quad j = 0, ..., K-1$, where $i$ is an imaginary unity. $P$ is the activation function of the neuron:

$$ P(z) = \text{exp}(i2\pi j/K), \quad \text{if} \quad 2\pi j/K \leq \text{arg} \ z < 2\pi (j+1)/K, \quad (2) $$

where $j=0, ..., K-1$ are the values of $K$-valued logic, $z = w_0 + w_1 x_1 + ... + w_n x_n$ is a weighted sum, $\text{arg} \ z$ is the argument of the complex number $z$. Fig. 1 illustrates the idea behind (2). Function (2) divides a complex plane onto $K$ equal sectors and maps the whole complex plane into a subset of points belonging to the unit circle. This is a set of $K$th roots of unity.

The MVN learning is reduced to the movement along the unit circle. This movement does not require a derivative of the activation function, because it is impossible to move in the incorrect direction. Any direction along the circle always leads to the target. The shortest way of this movement is completely determined by an error that is a difference between the desired and actual outputs. Let $\epsilon^d$ be a desired output of the neuron (see Fig. 2) and $\epsilon^s = P(z)$ be an actual output of the neuron. The most efficient MVN learning algorithm is based on the error correction learning rule [6]:

$$ W_{r+1} = W_r + C_r \left( (\epsilon^d - \epsilon^s) \ X \right), \quad (3) $$

where $X$ is an input vector, $n$ is a number of neuron’s inputs, $\overline{X}$ is a vector with the components complex conjugated to the components of vector $X$, $r$ is the number of iteration, $W_r$ is a current weighting vector, $W_{r+1}$ is a weighting vector after correction, $C_r$ is a learning rate.

The convergence of the learning process based on the rule (3) is proven in [6]. The rule (3) ensures such a correction of the weights that the weighted sum is moving from the sector $s$ to the sector $q$ (see Fig. 2). The direction of this movement is completely determined by the error $\delta = \epsilon^d - \epsilon^s$. The correction of the weights according to (3) changes the weighted sum exactly on the value $\delta$.

The activation function (2) is discrete. It has been recently proposed in [1],[2], to modify the function (2) in order to generalize it for the continuous case in the following way. If $K \rightarrow \infty$ in (2) then the angle value of the sector (see Fig. 1) tends to zero. Hence, the function (2) is transformed in this case as follows:
$$P(z) = \exp(i \arg z) = e^{i \arg z} = \frac{z}{|z|},$$  \hspace{1cm} (4)

where Arg \( z \) is a main value of the argument of the complex number \( z \) and \(|z|\) is its modulo.

The function (4) maps the complex plane into a whole unit circle, while the function (2) maps a complex plane just into a discrete subset of the points belonging to the unit circle. Thus the activation function (4) determines a continuous-valued MVN. The learning rule (3) is modified for the continuous-valued case in the following way [1],[2]:

$$\delta = e^s - \frac{z}{|z|} = T - \frac{z}{|z|},$$ \hspace{1cm} (6)

using a normalization by the factor \( 1/|z| \) :

$$\tilde{\delta} = \frac{1}{|z|} \delta = \frac{1}{|z|} \left( T - \frac{z}{|z|} \right),$$ \hspace{1cm} (7)

Learning according to the rule (6)-(7) makes possible squeezing a space for the possible values of the weighted sum. Thus this space can be reduced to the respectively narrow ring, which includes the unit circle inside, by using (6),(7) instead of (5). This approach can be useful in order to exclude a situation when small changes either in the weights or the inputs lead to a significant change of \( z \).

B. MVN-based Multilayer Feedforward Neural Network

A multilayer feedforward neural network (MLF, it is also often referred as a "multilayer perceptron" - MLP) and the backpropagation learning algorithm for it are well studied. A multilayer architecture of the network with a feedforward dataflow through nodes that requires full connection between consecutive layers and an idea of a backpropagation learning algorithm was proposed in [25] by D. E. Rumelhart and J. L. McClelland. It is well known fact that MLF is based traditionally on the neurons with a sigmoid activation function. MLF learning is based on the algorithm of error backpropagation. The error is being sequentially distributed form the "right hand" layers to the "left hand" ones. A crucial point of the backpropagation is that the error of each neuron of the network is proportional to the derivative of the activation function.

On the other hand, it is possible to use different neurons as the basic ones for a network with the feedforward architecture. A multilayer feedforward neural network based on multi-valued neurons (MLMVN) has been recently proposed in [1],[2]. This network has at least two principal advantages in comparison with an MLF: higher functionality and simplicity of learning, i.e. an MLMVN with the smaller number of neurons outperforms an MLF with the larger number of neurons [1],[2].

As it is mentioned above for a single neuron, the differentiability of the MVN activation function is not required for its learning. Since the MVN learning is reduced to the movement along the unit circle, the correction of weights is completely determined by the neuron’s error. The same property holds not only for a single MVN, but for any MVN-based network.

MLMVN is a multilayer neural network with standard feedforward architecture, where the outputs of neurons from the preceding layer are connected with the corresponding inputs of neurons from the following layer.

The network contains one input layer, \( m \)-1 hidden layers and one output layer. Let us use here the following notations. Let \( T_{in} \) be a desired output of the \( k^{th} \) neuron from the \( m \)-th (output) layer; \( Y_{bm} \) be an actual output of the \( k^{th} \) neuron from the \( m \)-th (output) layer. Then the global error of the network taken from the \( k^{th} \) neuron of the \( m \)-th (output) layer is calculated as follows:

$$\delta_{km} = T_{in} - Y_{km}.$$  \hspace{1cm} (8)

The learning algorithm for the classical feedforward network is derived from the consideration that the global error of the network expressed in terms of the mean squared error (MSE) is minimized.

The square error functional for the \( s \)-th pattern \( X_s = (x_1,\ldots,x_n) \) is as follows:

$$E_s = \sum_k (\delta_{km}^s)^2(W),$$ \hspace{1cm} (9)

where \( \delta_{km}^s \) is a global error of the \( k^{th} \) neuron of the \( m \)-th (output) layer, \( E_s \) is a square error of the network for the \( s \)-th pattern, and \( W \) denotes all the weighting vectors of all the neurons of the network. It is fundamental that the error depends not only on the weights of the neurons from the output layer but on all neurons of the network.

The mean square error functional for the network is defined as follows:

$$E = \frac{1}{N} \sum_{s=1}^{N} E_s,$$ \hspace{1cm} (10)

where \( E \) is a mean square error of the whole network and \( N \) is a total number of patterns in the training set.

The backpropagation of the global errors \( \delta_{km}^s \) through the network is used (from the \( m \)-th (output) layer to the \( m-1 \)-th one, from the \( m-1 \)-th one to the \( m-2 \)-th one, ..., from the 2nd one to the 1st one) in order to express the error of each neuron \( \delta_{jm}, j = 1,\ldots,m \) by means of the global errors \( \delta_{km}^s \) of the entire network.

The MLF backpropagation learning algorithm [26] cannot be applied for the case of MLMVN because it strongly depends on the derivative of the activation function. Following the backpropagation learning algorithm for the MLMVN proposed in [1],[2], the errors of all the neurons

475
from MLMVN are determined by the global errors of the network (8). The MLMVN learning is based on the minimization of the error functional (10).

Let us use the following notations. Let \( w_{ij} \) be the weight corresponding to the \( i \)th input of the \( j \)th neuron (\( k \)th neuron of the \( j \)th level), \( Y_j \) be the actual output of the \( j \)th neuron from the \( j \)th layer (\( j=1,\ldots,m \)), and \( N_j \) be the number of the neurons in the \( j \)th layer. It means that the neurons from the \( j+1 \)th layer have exactly \( N_j \) inputs. Let \( x_1,\ldots,x_m \) be the network inputs. The backpropagation learning algorithm for MLMVN was presented in [2].

The global errors of the entire network are determined by (8). We have to distinguish the global error of the network \( \delta_{mN} \) from the local errors \( \delta_{jm} \) of the particular output layer neurons. It is important to remember that the global error of the network consists not only of the output neurons errors, but of the local errors of the output neurons and hidden neurons. It means that in order to obtain the local errors for all neurons, the global error must be shared among these neurons. Hence, the local errors are represented in the following way. The errors of the \( m \)th (output) layer neurons are:

\[
\delta_{km} = \frac{1}{s_m} \sum_{i=1}^{N_m} \delta_{mi}(w_{ji}^{m+1})^{-1},
\]

where \( km \) specifies the \( k \)th neuron of the \( m \)th layer; \( s_m = N_{m-1} + 1 \), i.e. the number of all neurons on the previous layer (layer \( m-1 \) which the error is backpropagated to) incremented by 1.

The errors of the hidden layers neurons are computed as follows:

\[
\delta_{ji} = \frac{1}{s_j} \sum_{i=1}^{N_j} \delta_{ji+1}(w_{ji}^{j+1})^{-1},
\]

where \( kj \) specifies the \( k \)th neuron of the \( j \)th layer (\( j=1,\ldots,m-1 \)); \( s_j = N_{j-1} + 1 \), \( j = 2,\ldots,m \), \( s_1 = 1 \) is the number of all neurons on the layer \( j-1 \) (the previous layer \( j \) which error is backpropagated to) incremented by 1. Thus, the equations (11),(12) determine the error backpropagation for MLMVN. It is worth to stress on its two distinctions from the classical error backpropagation for MLF: 1) the equations (11),(12) do not contain a derivative of the activation function; 2) for MLMVN the errors of the neurons at a preceding layer are based on the inversely weighted summations of the next layer errors, while for MLF they are based on the weighted summations of the next layer errors.

Let us clarify a role of the factor \( 1/s_j \) in (11) and (12). The learning rule (5) for a single MVN determines that

\[
\Delta W = \frac{C_m}{(n+1)} \left( e^q \cdot \frac{x}{|z|} \right) \overline{X}.
\]

This expression contains the factor \( 1/(n+1) \) in order to share a contribution of the correction uniformly among all the \( n+1 \) weights \( w_0, w_1, \ldots, w_n \). Since all the inputs are equitable, it is natural that during the correction procedure \( \Delta W \) has to be shared among all the weights uniformly. We have to take into account the same property in a case of having not a single neuron but a feedforward network. It has to be used, in order to implement properly a backpropagation of the error through the network. It means, that if the error of a neuron on the layer \( j \) is equal to \( \delta_j \), then this \( \delta_j \) must contain a factor \( 1/s_j \). This ensures sharing of the particular neuron error among all the neurons on which this error depends. It should be remembered that for the 1st hidden layer the parameter \( s_j = 1 \) because there is no previous hidden layer, and there are no neurons the weight may be shared with.

The weights for all neurons of the network are corrected after calculation of the errors. In order to do this, we can use either the learning rule (3) or (5) depending on the discrete- (2) or continuous-valued (4) model. Hence, the following correction rules are used for the weights following [1],[2]:

\[
\tilde{w}_{ji}^{m+1} = w_{ji}^{m+1} + \frac{C_m}{(N_m + 1)} \delta_{mi} Y_{m-1}, \quad i = 1,\ldots,n,
\]

(13)

for the neurons from the \( m \)th (output) layer (\( k \)th neuron of the \( m \)th layer),

\[
\tilde{w}_{ji}^{m+1} = w_{ji}^{m+1} + \frac{C_j}{(N_j + 1)} \left| z_{ji} \right| \delta_{ji} Y_{j-1}, \quad i = 1,\ldots,n,
\]

(14)

for the neurons from the \( j \)th till \( m-1 \)th layer (\( k \)th neuron of the \( j \)th layer (\( j=2,\ldots,m-1 \)), and

\[
\tilde{w}_{ji}^{k+1} = w_{ji}^{k+1} + \frac{C_{j-1}}{(n+1)} \left| z_{ji} \right| \delta_{ki} X_i, \quad i = 1,\ldots,n,
\]

(15)

for the neurons of the 1st hidden layer, where \( C_j \) is a constant part of the learning rate. The factor \( 1/|z_{ji}| \) is a variable and self-adaptive part of the learning rate that was introduced in the modified learning rule (6),(7). It is used in (14) and (15) that determine the learning process for the hidden neurons. However, it is absent in (13) that determine the learning process for the output neurons. For the output neurons the errors calculated according to (8) are known, while for all the hidden neurons the errors are obtained according to the heuristic rule. The use of \( 1/|z_{ji}| \) in (14) and (15) is important in order to avoid situations when the weighted sum for the hidden layers neurons may become not a smooth function with dramatically high jumps. This can cause a drastic increasing of the number of iterations for the weights adjustment.
It should be mentioned that it is possible to set $C_{ij} = 1$ in (14), (15), and then learning rate contains only a variable part. The learning rate in this form is used in simulations in Section IV.

In general, the learning process should continue until the following condition is satisfied:

$$E = \frac{1}{N} \sum_{i=1}^{N} (\delta_{\text{learn}})^2(W) = \frac{1}{N} \sum_{i=1}^{N} E_i \leq \lambda,$$

(16)

where $\lambda$ determines the precision of learning. In particular, in the case when $\lambda = 0$ the equation (16) is transformed to $\forall k, \forall s \delta_{\text{learn}} = 0$.

III. IMAGE RESTORATION

A. Deconvolution problem

Mathematically, a variety of image capturing principles can be modelled by the Fredholm integral of the first kind in $\mathbb{R}^2$ space $z(t) = \int_X v(t, l)y(l)dl$, where $t, l \in X \subset \mathbb{R}^2$, $v$ is a point-spread function (PSF) of a system, $y$ is an image intensity function, and $z(t)$ is an observed image [15]. A natural simplification is that the PSF $v$ is shift-invariant which leads to a convolution operation in the observation model. We assume that the convolution is discrete and noise is present. Hence, the observed image $z$ given in the following form:

$$z(t) = (v \otimes y)(t) + \epsilon(t),$$

(17)

where " $\otimes$ " denotes the convolution, $t$ is defined on the regular $L_1 \times L_2$ lattice, $i \in X = \{t_i, t_j\}; t_i = 0, 1, ..., L_i - 1, i = 1, 2\}$, and $\epsilon(t)$ is a noise. It is assumed that the noise is white Gaussian with zero-mean and variance $\sigma^2$, $\epsilon(t) \sim N(0, \sigma^2)$. In the 2D frequency domain the model (17) takes the form:

$$Z(\omega) = V(\omega)Y(\omega) + \epsilon(\omega),$$

(18)

where $Z(\omega) = F[z(t)]$ is a representation of a signal $z$ in a Fourier domain and $F[\cdot]$ is a discrete Fourier transform, $V(\omega) = F[v(t)], Y(\omega) = F[y(t)], \epsilon(\omega) = F[\epsilon(t)]$, and $\omega \in W, W = \{(\omega_i, \omega_i) ; \omega = 2\pi k_i / L_i, k_i = 0, 1, ..., L_i - 1, i = 1, 2\}$ is the normalized 2D frequency. Equation (18) means that the convolution (17) is circular.

The removal of the degradation caused by a PSF is an inverse problem, widely referred to as deconvolution. Usually this problem is ill-posed which results in the instability of a solution, i.e. it is highly sensitive to the noise. The stability can be provided by constraints imposed on the solution. A general approach to this kind of problems refers to the methods of Lagrange multipliers and the Tikhonov regularization [16]. The regularized inverse filter can be obtained as a solution of the least square problem with a penalty term:

$$J = \|Z - VY\|_F^2 + \alpha \|Y\|_F^2,$$

(19)

where $\alpha \geq 0$ is a regularization parameter and $\|\cdot\|_F$ denotes $F$-norm. Here, the first term $\|Z - VY\|_F^2$ gives the fidelity to the available data $Z$ and the second term bounds the power of this estimate by means of the regularization parameter $\alpha$.

In (19), and further, we omit the argument $\omega$ in the Fourier transform variables. We obtain the solution in the following form by minimizing (19):

$$\hat{Y} = \frac{\overline{V}}{|V|^2 + \alpha} Z, \hat{y}_\alpha(x) = F^{-1}\{\hat{Y}\},$$

(20)

where $\hat{Y}$ is an estimate of $Y$, and $\overline{V}$ denotes complex-conjugate value of $V$.

B. Blur Model

In this paper we consider Gaussian, motion and rectangular (boxcar) blurs. We aim to identify both blur, which is characterized by PSF, and its parameter using a single network.

The PSF $v$ describes how the point source of light is spread over the image plane. It is one of the main characteristics of the optical system. For a variety devices, like photo or video camera, microscope, telescope, etc., PSFs are often approximated by the Gaussian function:

$$v(t) = \frac{1}{2\pi\tau^2} \exp\left(-\frac{t^2 + t_2^2}{2\tau^2}\right)$$

(21)

where $\tau$ is a parameter of the PSF, usually referred to as the variance (Fig. 4a). Its Fourier transform $V$ is also a Gaussian function and its absolute values $|V|$ are shown in Fig. 4d.

Another example of blur is a uniform linear motion which happens while taking a picture of a moving object relatively to the camera:

$$v(t) = \begin{cases} \frac{1}{h^2} \sqrt{t_1^2 + t_2^2} < h/2, t_1 \cos \phi = t_2 \sin \phi, \\ 0, \quad \text{otherwise}, \end{cases}$$

(22)

where $h$ is a parameter which depends on the velocity of the moving object and describes the length of motion in pixels, and $\phi$ is the angle between the motion orientation the horizontal axis. Any uniform function like (22) is characterized by the number of slopes in the frequency domain (Fig. 4b,c).

The uniform rectangular blur is described by the following function (Fig. 4c):

$$v(t) = \begin{cases} \frac{1}{h^2}, & |t_1| < \frac{h}{2}, |t_2| < \frac{h}{2}, \\ 0, \quad \text{otherwise}, \end{cases}$$

(23)

where parameter $h$ defines the size of smoothing area. The frequency characteristics of (23) are shown in Fig. 4f.
values are shown in Fig. 5. The pattern vector is MLMVN.

Structure of the neural element on the output layer of Fig. 3.

IV. SIMULATIONS

A. Training Set Formation

The observed image \( z(t) \) is modeled as the output of a linear shift-invariant system (17) which is characterized by the PSF \( v \). Since in the frequency domain this model is a product of the true object function \( Y \) and \( v \), we state the problem as a recognition of the shape of \( V \) and its parameters from the power-spectral density (PSD) of the observation \( Z \), i.e. from \( |Z|^2 = Z \cdot \bar{Z} \). In terms of statistical expectation we can rewrite that as following:

\[
E \left[ |Z|^2 \right] = E \left[ |Y^2 + n|^2 \right] = |Y|^2 |V|^2 + \sigma^2
\]

(24)

where \( \sigma^2 \) is the variance of noise in (18).

Examples of \( \log |Z| \) values are shown in Fig. 5. The distortions of PSD for the test image (Fig. 5a) that are typical for each type of blur (Fig. 5b-e) are clearly visible in Fig. 5g-j.

For the sake of simplicity we consider the image \( z(t) \) with the equal sizes, i.e. \( L = L_s = L_a \) in (17),(18). In order to obtain the training vector \( X = (x_1, ..., x_s) \) in (15) as an input data for the network, and taking into account that the PSF \( v \) is symmetrical, PSD of \( z(t) \) (24) is used as follows:

\[
x_j = \exp \left( 2\pi i \cdot (K - 1) \right) \frac{\log \left[ Z \left( \omega_{k_1, k_2} \right) \right] - \log \left[ \min_k \left( Z \left( \omega_{k, k_s} \right) \right) \right]}{\log \left[ \max_k \left( Z \left( \omega_{k, k_s} \right) \right) \right] - \log \left[ \min_k \left( Z \left( \omega_{k, k_s} \right) \right) \right]},
\]

(25)

where

\[
\begin{align*}
&j = 1, ..., \frac{L}{2} - 1, \quad \text{for} \ k_1 = k_2, \ k_1 = 1, ..., L / 2 - 1, \\
&j = \frac{L}{2}, ..., L - 2, \quad \text{for} \ k_1 = 1, \ k_1 = 1, ..., L / 2 - 1, \\
&j = L - 1, ..., \frac{3L}{2} - 3, \quad \text{for} \ k_1 = 1, \ k_1 = 1, ..., L / 2 - 1,
\end{align*}
\]

(26)

and \( Z_{max} = \max_{k_1, k_2} \left( Z \left( \omega_{k_1, k_2} \right) \right), \ Z_{min} = \min_{k_1, k_2} \left( \min_{k_1} \left( Z \left( \omega_{k_1, k_2} \right) \right) \right) \), and \( K \) is a number of sectors in (2). Eventually, the length of the pattern vector is \( n = 3L / 2 - 3 \).

Some examples of vectors of PSD log values multiplied by \( K \)-1 used in (25),(26) to obtain the input training vector \( X \) are shown in Fig. 6.

B. Neural Network Structure

In this paper we use the discrete-valued MLMVN whose neurons have the activation function (2) with \( K = 2048 \). The MLMVN has two hidden layers consisting of 5 and 30 neurons, respectively, and the output layer which consists of the same number of neurons as the number of classes, i.e. types of blur. Since we consider four types of blur (Gaussian, rectangular, linear uniform horizontal, \( \phi = 0 \) in (22), and vertical, \( \phi = 90 \) in (22), motion) the output layer contains four neurons. Therefore, the structure of network is \( 5 \rightarrow 30 \rightarrow 4 \).

Each neural element of the output layer is responsible for recognition of one type of blur and its parameter. The MVN activation function (2) (see Fig. 1) for the output layer neurons has a specific form. The upper semi-plane of the complex plane is used to classify one particular blur and its parameters and the lower semi-plane is used to reject other blurs (Fig. 3).

We consider each blur with four parameters. The Gaussian blur is considered with \( \tau \in \{1, 1.5, 2, 2.5\} \) in (21); the linear uniform horizontal motion blur of the lengths 3, 5, 7, 9, in (22); the data corrupted by the linear uniform vertical motion blur of the length 3, 5, 7, 9, in (22); rectangular has sizes \( 3 \times 3 \), \( 5 \times 5 \), \( 7 \times 7 \), \( 9 \times 9 \), in (23). Hence, the upper semi-plane is divided onto nonoverlapping groups of sectors, each group for the corresponding blur parameter value (Fig. 3). For instance, the upper semi-plane for the first neuron is used to identify the Gaussian blur and lower is used to reject the non Gaussian one. If the weighted sum for the 1st neuron at the output 3rd layer hits \( j \)-th group, \( j \in \{1, ..., 4\} \), in the upper semi-plane then the input vector \( X = (x_1, ..., x_s) \) corresponds to the Gaussian blur and its parameter is \( \tau_j \).

C. Results

We have used a database which consists of 108 grayscale images with sizes 256×256 to generate the training and testing sets. 72 images are used to generate the training set and 36 for the testing set.

<table>
<thead>
<tr>
<th>Table I. Classification Rate of Blur Identification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blur</td>
</tr>
<tr>
<td>----------------------------</td>
</tr>
<tr>
<td>No blur</td>
</tr>
<tr>
<td>Gaussian</td>
</tr>
<tr>
<td>Rectangular</td>
</tr>
<tr>
<td>Motion horizontal</td>
</tr>
<tr>
<td>Motion vertical</td>
</tr>
</tbody>
</table>

The images with no blur and no noise are also included in both the training and testing set. Eventually, the training set consists of \( 72 \cdot 4 \cdot 4 + 72 = 1224 \) vectors, and the testing set consists of \( 36 \cdot 4 \cdot 4 + 36 = 612 \) vectors. The level of noise is
kept to satisfy blurred signal-to-noise ratio (BSNR) to be 40dB [17],[18].

When the training set is generated, the backpropagation training algorithm (11)-(15) is exploited to train the MLMVN.

The trained network is used to make classification on the testing set. The classification rate is used as an objective criterion of classification. It is computed as a number of correct classifications in terms of percentage (%) for each type of blur.

The results are presented in Table 1. The first row corresponds to the recognition of the original non-blurred images. All the output layer neurons classified them as those that are not distorted by any of the considered types of blur.

The results of blur identification obtained in this paper are better or comparative with those presented in [12],[24]. The best ones are highlighted by the bold font.

It is important to stress that in this paper we use a single neural network with highly efficient learning algorithm, while in [12],[24] two separate networks are used. The number of features used for classification in this paper is 381 while in [12],[24] it is equal to 16384. As a result the MLMVN can be trained in less than one hour on a PC with Pentium IV 3.20 GHz CPU. The software simulator was implemented in Borland Delphi 5.0 environment.

It is worth to note that the larger number of the considered blur types and parameters for each type may lead to the increasing of the number of neurons in MLMVN.

V. CONCLUSIONS

In this paper we propose a novel technique for blur identification using a single observed image. The technique employs a neural network based on the multi-valued neurons (MLMVN) which is trained for a database of images. Then this network is used to identify both type and parameters of the blur. The obtained results show high efficiency of the proposed approach. As a further development of MLMVN we consider the use of MVN with continuous activation function (4).

REFERENCES


Fig. 4. Types of PSF used: a) Gaussian PSF with $\tau = 2$ and size $21 \times 21$; b) Linear uniform motion blur of the length 5; c) Boxcar blur of the size $3 \times 3$; d) frequency characteristics of a); e) frequency characteristics of b); f) frequency characteristics of c).

Fig. 5. True test Cameraman image (a) blurred by: b) Gaussian blur with $\tau = 2$; c) boxcar blur of the size $9 \times 9$; d) horizontal linear uniform motion blur of the length $h = 9$; e) vertical linear uniform motion blur of the length $h = 9$. Logarithm of the PSD of the true test Cameraman image (f) blurred by: g) Gaussian blur with $\tau = 2$; h) rectangular blur of the size $9 \times 9$; i) horizontal linear uniform blur of the length $h = 9$; j) vertical linear uniform blur of the length $h = 9$.

Fig. 6. The normalized multiplied by $K$-1 logarithm values of PSD of $Z$ used as arguments to generate training vectors in (25),(26) obtained from the true test Cameraman image (a) blurred by: b) Gaussian blur with $\tau = 2$; c) boxcar blur of the size $9 \times 9$; d) horizontal linear uniform blur of the length $h = 9$; e) vertical linear uniform blur of the length $h = 9$.