Blur Identification by Multilayer Neural Network Based on Multivalued Neurons

Igor Aizenberg, *Senior Member, IEEE*, Dmitriy V. Paliy, Jacek M. Zurada, *Fellow, IEEE*, and Jaakko T. Astola, *Fellow, IEEE*

***Abstract*—A multilayer neural network based on multivalued neurons (MLMVN) is a neural network with a traditional feedforward architecture. At the same time, this network has a number of specific different features. Its backpropagation learning algorithm is derivative-free. The functionality of MLMVN is superior to that of the traditional feedforward neural networks and of a variety kernel-based networks. Its higher flexibility and faster adaptation to the target mapping enables to model complex problems using simpler networks. In this paper, the MLMVN is used to identify both type and parameters of the point spread function, whose precise identification is of crucial importance for the image deblurring. The simulation results show the high efficiency of the proposed approach. It is confirmed that the MLMVN is a powerful tool for solving classification problems, especially multiclass ones.**

***Index Terms*—Blind deconvolution, complex-valued neuron, derivative-free learning, feedforward network, multivalued neuron.**

I. Introduction

A MULTILAYER neural network based on multivalued neurons (MLMVN) has been introduced in [1] and then investigated and developed further in [2]. This network consists of multivalued neurons (MVN), having 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 seminal paper of N. Aizenberg *et al.* [3].

The multivalued neuron was introduced in [4]. It is based on the principles of multivalued threshold logic over the field of complex numbers formulated in [5] and then developed in [6]. A comprehensive study of the discrete-valued MVN, its properties, and learning is presented in [6]. A continuous-valued MVN and its learning are considered in [1] and [2]. In this paper, we consider the continuous-valued MVN (further called MVN) only.

The most important properties of MVN are the complex-valued weights, inputs and output lying on the unit circle, and the activation function that maps the complex plane into the unit circle. Furthermore, MVN learning is reduced to the movement along the unit circle. This 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 introduced in recent years, e.g., cellular neural networks [6], neural associative memories [6]–[10], variety of pattern recognition systems [10]–[12], and the MLMVN [1], [2]. It has been shown that the MLMVN outperforms a classical multilayer feedforward network and several kernel-based networks in terms of learning speed and network complexity. It also offers better classification/prediction rate tested for such popular benchmark problems including the parity n, the two spirals, the sonar, and the Mackey–Glass time series prediction [1], [2]. These properties of MLMVN show that it can be more flexible and adapt faster in comparison with other networks. In this paper, we apply MLMVN to identify blur and its parameters, which is a crucial 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, deblurring (restoration) of a blurred image is an ill-posed inverse problem, and regularization is commonly used when solving this problem [17].

There exist a variety of sophisticated and efficient deblurring techniques such as deconvolution based on the Wiener filter [13], [18], nonparametric image deblurring using local polynomial approximation with spatially adaptive scale selection based on the intersection of confidence intervals rule [18], Fourier-wavelet regularized deconvolution [19], expectation–maximization algorithm for wavelet-based image deconvolution [20], etc. All these techniques assume a prior knowledge of the blurring kernel or point spread function (PSF) and its parameter.

When the blurring operator is unknown, the image restoration becomes a blind deconvolution problem [21]–[23]. Most of the methods to solve it are iterative, and, therefore, computationally costly. Due to the presence of noise, they suffer from stability and convergence problems [24].

The most popular approaches to blind deconvolution considered in the scientific literature can be divided in two classes: a multichannel deconvolution [24]–[26], and a single-channel one [27]–[32].

The multichannel blind deconvolution assumes that several observations of a single scene are available for restoration. The problem is to restore the true scene from these noisy,

Manuscript received December 15, 2006; revised June 9, 2007; accepted September 24, 2007. This work was supported in part by the Academy of Finland, Project 213462, Finnish Centre of Excellence program (2006-2011) and by the Finnish Funding Agency for Technology and Innovation (Tekes).

I. Aizenberg is with Texas A&M University—Texarkana, Texarkana, TX 75505 USA (e-mail: igor.aizenberg@tamut.edu).

D. V. Paliy and J. T. Astola are with Tampere International Center for Signal Processing, Tampere University of Technology, 33101 Tampere, Finland (e-mail: dmitriy.paliy@tut.fi; jaakko.astola@tut.fi).

J. M. Zurada is with Computational Intelligence Laboratory, University of Louisvillle, Louisville, KY 40292 USA (e-mail: jacek.zurada@louisville.edu).

Digital Object Identifier 10.1109/TNN.2007.914158

differently blurred data having no preliminary knowledge about the distortion (smoothing) operators [25]. Multichannel deblurring of spatially misaligned images has been considered in [24]. Restoration from two observations where one of them is very noisy (for instance, taken by the camera phone with short exposition time), and one is less noisy but blurred (taken with long exposition time) was proposed in [26].

The single-channel blind deconvolution usually assumes preliminary knowledge on the model of blur. For instance, it can be defocus model, Gaussian model, motion model, etc. After this, having one observation of a scene and knowing the type of distortion operator, the problem is to estimate the parameters of this model that can be described mathematically (e.g., variance for the Gaussian blur, extent for motion blur, etc.). For instance, restoration from data destroyed with motion blur was considered in [28]. The parameter for Gaussian model of blur is determined efficiently by wavelet decomposition in [29]. Single-channel blind deconvolution within Bayesian framework is considered in [30]. The known size of PSF support may simplify the problem significantly [31]. Parametric solution where a prior imposed on PSF takes into account multiple classes is proposed in [32].

In this paper, we propose to identify a blur model and its parameters from the finite number of multiple models and the corresponding parameters using an observed blurred image. The proposed identification technique is based on the use of the MLMVN as an identifier (classifier). The proposed solution is a one step process, which is significantly different from typical single-channel blind deconvolution.

Recently, applications of neural networks in image restoration became very popular. In [33], a neural network is used for segmentation. The authors propose an approach to find and classify areas in one photo image that are blurred and in-focus. In [34], a support vector machine (SVM)-based method is used for blind superresolution image restoration.

The original solutions of blur identification problem based on the use of MVN-based neural networks were proposed in [12] and [35]. Two different single-layer MVN-based networks have been used to identify blur and its parameter in [35]. The results were good, but this approach had some disadvantages. For instance, the networks have a 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 image size). Thus, the learning process was heavy.

Significant improvement was proposed in [36] as compared to [35]. A single neural network (the discrete-valued MLMVN) with the original backpropagation training scheme was used to identify both smoothing operator and its parameter on a single observed noisy image. However, the quantization is used to compute the inputs and outputs for all neurons of the discrete-valued MLMVN. The introduced quantization errors affect the results.

In this paper, we propose to use the continuous-valued MLMVN to identify the blur model and its parameters avoiding the disadvantages mentioned previously. Preliminary results obtained using this approach [37] show that the modification of the MLMVN results in significant improvement of the functionality.

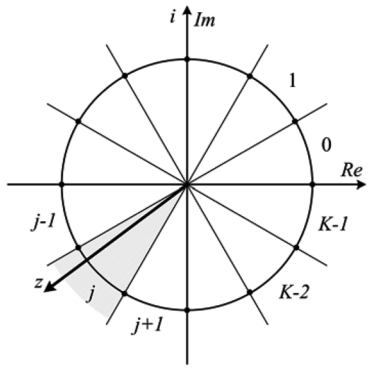


Fig. 1. Geometrical interpretation of the MVN activation function.

The structure of this paper is as follows. In Section II, we introduce MVN and MLMVN and their training algorithms. The problem of image deconvolution is described in Section III. The use of MLMVN for blur identification is described in Section IV, and simulation results are presented in Section V. The exact derivation of the error backpropagation algorithm for the MLMVN is presented in Appendix I. Numerical example that illustrates training of the MLMVN is presented in Appendix II.

II. Multilayer Neural Network Based on MVNs

A. MVN With Discrete Activation Function

An MVN was introduced in [4] as a neural element based on the principles of multivalued threshold logic over the field of complex numbers proposed in [5]. It was thoroughly analyzed in [6], where its theory, basic properties, and learning were presented.

A single discrete-valued MVN performs a mapping between n inputs and a single output. This mapping is described by a multivalued (K-valued) function of n variables $f(x_1,\dots,x_n)$ with $n+1$ complex-valued weights as parameters

$$f(x_1,\dots,x_n) = P(w_0 + w_1x_1 + \cdots + w_nx_n) \qquad (1)$$

where $X=(x_1,\dots,x_n)$ is an input vector (pattern vector) and $W=(w_0,w_1,\dots,w_n)$ is a weighted vector. The function and variables are the Kth roots of unity $\varepsilon^j = \exp(i2\pi j/K), j = 0,\dots,K-1$, where i is an imaginary unity. P is the activation function of the neuron

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

where $j = 0,\dots,K-1$ are the values of K-valued logic, $z = w_0 + w_1x_1 + \cdots + w_nx_n$ is a weighted sum, and $\arg z$ 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 Kth roots of unity.

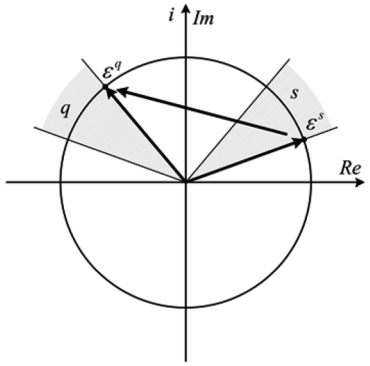


Fig. 2. Geometrical interpretation of the MVN learning rule.

There are two MVN learning algorithms. Both algorithms are analyzed in [6], as well as their convergence is proven there. Both algorithms are derivative-free. One of these algorithms is based on the error-correction learning rule. The latter is computationally more efficient and exactly this algorithm will be used in this paper. The error-correction learning rule is known form the classical perceptron theory [38]–[41]. In the perceptron error-correction learning, the direction, in which the "incorrect" actual weighted sum must move, as a result of the learning step, is determined by the difference between the desired output and the actual output. Because in the classical perceptron theory [38]–[41] only Boolean mappings between inputs and output are considered, this difference can be equal either to $2 = 1 - (-1)$ or to $-2 = -1 - 1$ (usually, the Boolean alphabet $\{1, -1\}$ is used instead of $\{0, 1\}$).

The perceptron is a binary neuron, while the MVN is a multivalued neuron. However, the error-correction learning rule works for the MVN as well. The MVN error-correction learning is based on the generalization of the perceptron error-correction learning. Evidently, the MVN learning is reduced to the movement along the unit circle: if the actual output is "incorrect," we have to move the weighted sum in the direction of the desired output. The right way of this movement is completely determined by an error that is the difference between the desired and actual outputs [6].

Let ε^q be a desired output of the neuron (see Fig. 2) and $\varepsilon^s = P(z)$ be an actual output of the neuron. The most efficient MVN learning algorithm is based on the error-correction learning rule [6]

$$W_{r+1} = W_r + \frac{C_r}{(n+1)}(\varepsilon^q - \varepsilon^s)\bar{X} \tag{3}$$

where X is an input vector, n is the number of neuron's inputs, $\bar{X}$ is a vector with the components complex conjugated to the components of vector X, r is the index of iteration, W_r is the current weighted vector, W_{r+1} is a weighted vector after correction, and 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 $\delta = \varepsilon^q - \varepsilon^s$. The correction of weights according to (3) changes the weighted sum exactly by the value δ, so by the value of the error

$$\begin{aligned}\tilde{z} &= \tilde{w}_0 + \tilde{w}_1 x_1 + \cdots + \tilde{w}_n x_n \\ &= w_0 + \frac{\delta}{(n+1)} + \left(w_1 + \frac{\delta}{(n+1)}\bar{x}_1\right)x_1 + \cdots \\ &\quad + \left(w_n + \frac{\delta}{(n+1)}\bar{x}_n\right)x_n \\ &= w_0 + \frac{\delta}{(n+1)} + w_1 x_1 + \frac{\delta}{(n+1)} + \cdots \\ &\quad + w_n x_n + \frac{\delta}{(n+1)} \\ &= w_0 + w_1 x_1 + \cdots + w_n x_n + \delta = z + \delta.\end{aligned}$$

It is worth to note that the factor $1/(n+1)$ in (3) is an important constant part of the learning rate. The use of this factor ensures distribution of the error among all the neuron's inputs. Because all of them are equitable, it is natural to consider that the error is uniformly distributed among all the $n+1$ synaptic weights. Without this factor the weighted sum would be changed by $\delta(n+1)$ instead of δ. This would lead to the permanent jumping over the desired output, without the convergence of the learning algorithm [2], [6].

B. MVN With Continuous Activation Function

The activation function (2) is discrete. As proposed in [1] and [2], (2) can be modified in order to generalize it for the continuous case in the following way: if $K \to \infty$ in (2), then the angle value of the sector (see Fig. 1) approaches zero. Hence, the function (2) can be defined as follows:

$$P(z) = \exp(i(\arg z)) = e^{i\operatorname{Arg} z} = \frac{z}{|z|} \tag{4}$$

where $\operatorname{Arg} z$ is the 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]:

$$\begin{aligned}W_{r+1} &= W_r + \frac{C_r}{(n+1)}(\varepsilon^q - e^{i\operatorname{Arg} z})\bar{X} \\ &= W_r + \frac{C_r}{(n+1)}\left(\varepsilon^q - \frac{z}{|z|}\right)\bar{X}.\end{aligned} \tag{5}$$

It is worth to consider the following modification of (3) and (5):

$$W_{r+1} = W_r + \frac{C_r}{(n+1)}\tilde{\delta}\bar{X} \tag{6}$$

where $\tilde{\delta}$ is obtained from $\delta = \varepsilon^q - z/|z| = T - z/|z|$ using a normalization by the factor $1/|z|$

$$\tilde{\delta} = \frac{1}{|z|}\delta = \frac{1}{|z|}\left(T - \frac{z}{|z|}\right). \tag{7}$$

$1/|z|$ becomes a variable and self-adaptive part of the learning rate (thus C_r becomes a constant part of the learning rate). By using (6) and (7) instead of (5) a space for the possible values of the weighted sum can be reduced to the respectively narrow ring, which includes the unit circle. This approach prevents small changes either of the weights or the inputs causing significant change of z.

C. MVN-Based Multilayer Feedforward Neural Network

A multilayer feedforward neural network [44] (MLF; it is also often referred to as a "multilayer perceptron") and backpropagation learning algorithm for it are well established. MLF learning is based on the algorithm of error backpropagation. The error is sequentially backpropagated from the "rightmost" layers to the "leftmost" ones. A crucial property of the MLF backpropagation is that the error of each neuron of the network is proportional to the derivative of the activation function.

As proposed in [1] and [2], MLMVN network has at least two principal advantages in comparison with an MLF: higher functionality (i.e., an MLMVN with the smaller number of neurons outperforms an MLF with the larger number of neurons) and simplicity of learning.

As mentioned previously, for a single MVN, the differentiability of the MVN activation function is not required for its learning. The MVN learning is reduced to the movement along the unit circle and it is based on the error-correction rule. Hence, the correction of weights is completely determined by the neuron's error. The same property holds not only for the single MVN, but for an MVN-based feedforward neural 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_{km} be a desired output of the kth neuron from the mth (output) layer; and Y_{km} be an actual output of the kth neuron from the mth (output) layer. Then, the global error of the network taken from the kth neuron of the mth (output) layer is calculated as follows:

$$\delta^*_{km} = T_{km} - Y_{km}. \tag{8}$$

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

$$E_s = \sum_k (\delta^*_{km})^2 \tag{9}$$

where δ^*_{km} is a global error of the kth neuron of the mth (output) layer and E_s is the square error of the network for the sth pattern. 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 E for the network with N training patterns is defined as follows:

$$E = \frac{1}{N}\sum_{s=1}^{N} E_s. \tag{10}$$

The backpropagation of the global errors δ^*_{km} through the network is used (from the mth (output) layer to the $(m-1)$th one, from the $(m-1)$th one to the $(m-2)$th one, ..., from the second one to the first one) in order to express the error of each neuron $\delta_{kj}, j = 1, \ldots, m$ by means of the global errors δ^*_{km} of the entire network.

The MLF backpropagation learning algorithm [38], [39], [43] cannot be applied to MLMVN because it strongly depends on the derivative of the activation function. Following the backpropagation learning algorithm for the MLMVN proposed in [1] and [2], the errors of all the neurons 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 notation: let w_i^{kj} be the weight corresponding to the ith input of the kjth neuron (kth neuron of the jth layer), Y_{ij} be the actual output of the ith neuron from the jth layer ($j = 1, \ldots, m$), and N_j be the number of the neurons in the jth layer. It means that the neurons from the $(j+1)$th layer have exactly N_j inputs. Let $x_1, \ldots, x_n$ be the network inputs. The backpropagation learning algorithm for MLMVN with a single hidden layer and a single output neuron was first presented in [2].

The global errors of the entire network are determined by (8). We have to distinguish the global errors of the network δ^*_{km} from the local errors δ_{km} of the particular output layer neurons. It is essential 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 distributed among these neurons. Hence, the local errors are represented in the following way. The errors of the mth (output) layer neurons

$$\delta_{km} = \frac{1}{s_m}\delta^*_{km} \tag{11}$$

where km specifies the kth neuron of the mth layer, $s_m = N_{m-1}+1$, i.e., the number of all neurons on the preceding layer (layer $m-1$ which the error is propagated back to) incremented by 1.

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

$$\delta_{kj} = \frac{1}{s_j}\sum_{i=1}^{N_{j+1}} \delta_{ij+1}\left(w_k^{ij+1}\right)^{-1} \tag{12}$$

where kj specifies the kth neuron of the jth layer ($j = 1, \ldots, m-1$); and $s_j = N_{j-1}+1, j = 2, \ldots, m$ is the number of all neurons in the layer $j-1$ (the preceding layer j which error is backpropagated to) incremented by 1. Note that $s_1 = 1$ because there is no preceding layer for the first hidden layer, that is $N_{1-1} = N_0 = 0$. The derivation of the error backpropagation rule (11) and (12) is presented in the Appendix I.

Thus, (11) and (12) determine the backpropagation of error for the MLMVN. It is worth to stress two distinctions from the classical error backpropagation for MLF: 1) equations (11) and (12) do not contain a derivative of the activation function; and 2) 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 requires that $\Delta W = \left(C_m/(n+1)\right)\left(\varepsilon^q - z/|z|\right)\overline{X}$. This expression contains the factor $1/(n+1)$ in order to distribute the correction uniformly among all the n+1 weights $w_0, w_1, ..., w_n$. Since all the inputs are equitable, the correction ΔW has to be distributed 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 $\hat{\delta}$, then this $\hat{\delta}$ must have followed by a factor $1/s_j$. This ensures distribution of the particular neuron error among all the neurons contributing to this error.

The weights for neurons of the network are corrected after calculation of all errors. This can be done by using the learning rule (3) or (5) depending on the discrete- (2) or continuous-valued (4) model. Let $\overline{\tilde{Y}}_{ij}$ be the complex conjugated output of the i^{th} neuron from the j^{th} layer after weights update in the current training step. Hence, the following correction rules are used for the weights [1], [2]:

$$\begin{aligned} \tilde{w}_i^{km} &= w_i^{km} + \frac{C_{km}}{(N_{m-1}+1)}\delta_{km}\overline{\tilde{Y}}_{im-1},\ i=1,...,n, \\ \tilde{w}_0^{km} &= w_0^{km} + \frac{C_{km}}{(N_{m-1}+1)}\delta_{km}, \end{aligned} \tag{13}$$

for the neurons from the m^{th} (output) layer (k^{th} neuron of the m^{th} layer),

$$\begin{aligned} \tilde{w}_i^{kj} &= w_i^{kj} + \frac{C_{kj}}{(N_{j-1}+1)\,|\,z_{kj}\,|}\delta_{kj}\overline{\tilde{Y}}_{ij-1},\ i=1,...,n, \\ \tilde{w}_0^{kj} &= w_0^{kj} + \frac{C_{kj}}{(N_{j-1}+1)\,|\,z_{kj}\,|}\delta_{kj}, \end{aligned} \tag{14}$$

for the neurons from the 2nd till m-1st layer (k^{th} neuron of the j^{th} layer (j=2, …, m-1), and

$$\begin{aligned} \tilde{w}_i^{k1} &= w_i^{k1} + \frac{C_{k1}}{(n+1)\,|\,z_{k1}\,|}\delta_{k1}\overline{x}_i,\ i=1,...,n, \\ \tilde{w}_0^{k1} &= w_0^{k1} + \frac{C_{k1}}{(n+1)\,|\,z_{k1}\,|}\delta_{k1}, \end{aligned} \tag{15}$$

for the neurons of the 1st hidden layer, where C_{kj} is a constant part of the learning rate.

The factor $1/|z_{kj}|$ 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 determines the learning process for the output neurons. For the output neurons, the errors calculated according to (8) are immediately known, while for all the hidden neurons the errors are obtained according to the heuristic rule. The use of $1/|z_{kj}|$ 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 increase of the number of iterations needed for weights adjustment.

It should be mentioned that it is possible to set $C_{kj}=1$ in (14),(15) , and then learning rate contains only a variable self-adaptive 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:

$$E = \frac{1}{N}\sum_{s=1}^{N}\sum_{k}(\delta_{kms}^{*})^2 = \frac{1}{N}\sum_{s=1}^{N}E_s \le \lambda\,, \tag{16}$$

where λ determines the precision of learning. In particular, in the case when $\lambda = 0$, (16) is transformed to $\delta_{kms}^{*} = 0$ for all k and all s.

The convergence proof of the learning algorithm (13)-(15) is based in [2]. While the MLF learning algorithm is usually considered as an optimization problem of finding a minimum of the error functional [38],[39], the convergence of the MLMVN training algorithm follows from the same considerations as of the learning algorithm with the rule (5) for the single neuron. The convergence of the learning algorithm with the rule (5) means that the following condition must be satisfied for the error:

$$\left|\arg\left(\varepsilon^q\right) - \arg\left(e^{iArg\ z}\right)\right| \le \lambda\,. \tag{17}$$

However, in this case the learning rule (5) is reduced to the learning rule (3) for the discrete case, because for the argument of the neuron's output we obtain the following condition: $\arg\left(\varepsilon^q\right) - \lambda \le \arg\left(e^{iArg\ z}\right) \le \arg\left(\varepsilon^q\right) + \lambda$. This means that the continuous-valued learning is reduced to the discrete-valued learning in a π/λ - valued logic. Indeed, it follows from the last considerations, that we produce exactly π/λ sectors on the complex plane (see Fig. 1) and $K = \pi/\lambda$ in (2). But for the discrete-valued case, the convergence of the learning algorithm is proven for any K in [6].

A numerical example that illustrates training of the MLMVN is presented in Appendix II.

III. Image Deblurring Problem

Mathematically, a variety of image capturing principles can be modelled by the Fredholm integral of the first kind in $\mathbb{R}^2$ space $z(t) = \int_X v(t,l)q(l)dl$, where $t, l \in X \subset \mathbb{R}^2$, v is a point-spread function (PSF) of a system, q 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 is given in the following form:

$$z(t) = (v * q)(t) + \varepsilon(t), \tag{18}$$

where "$*$" denotes the convolution, t is defined on the regular $L_1 \times L_2$ lattice, $t \in X = \{(t_1,t_2);\ t_i = 0,1,...,L_i - 1,\quad i = 1,2\}$, and $\varepsilon(t)$ is the noise. It is assumed that the noise is white Gaussian with zero-mean and variance σ^2, $\varepsilon(t) \sim N(0,\sigma^2)$. In the 2D frequency domain the model (18) takes the form:

$$Z(\omega) = V(\omega)Q(\omega) + \varepsilon(\omega), \tag{19}$$

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,

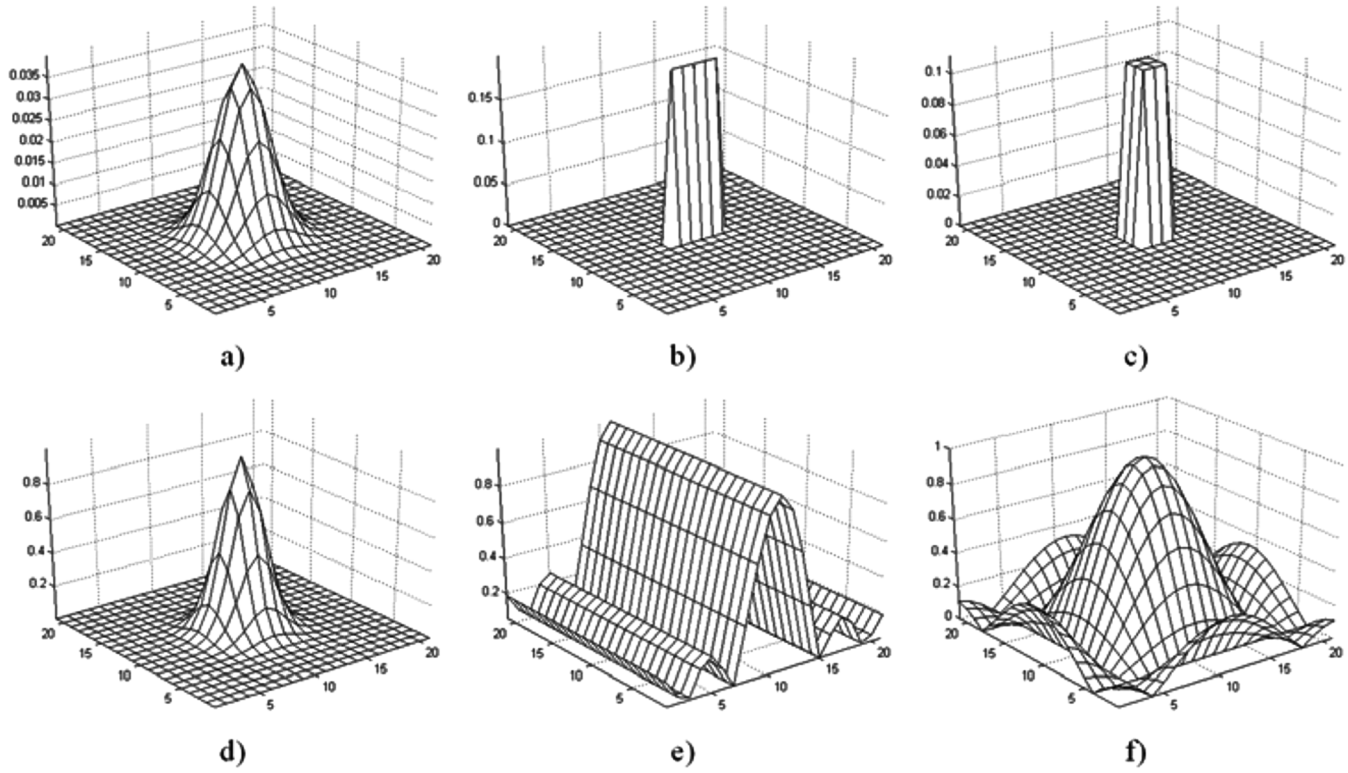


Fig. 3. Types of PSF used: (a) Gaussian PSF with $\tau = 2$ and size 21×21; (b) linear uniform motion blur of the length 5; (c) boxcar blur of the size 3×3; (d) frequency characteristics of (a); (e) frequency characteristics of (b); (f) frequency characteristics of (c).

$V(\omega) = F\{v(t)\}, Q(\omega) = F\{q(t)\}, \varepsilon(\omega) = F\{\varepsilon(t)\}$, $\omega \in W$, and $W = \{(\omega_1, \omega_2), \omega_i = 2\pi k_i / L_i, k_i = 0, 1, \ldots, L_i - 1, i = 1, 2\}$ is the normalized 2-D frequency.

The removal of the degradation caused by a PSF is an inverse problem, widely referred to as a deconvolution. Usually this problem is ill-posed and it 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 such problems refers to the methods of Lagrange multipliers and the Tikhonov regularization [17]. The regularized inverse filter can be obtained as a solution of the least square problem with a penalty term

$$J = \|Z - VQ\|_2^2 + \alpha \|Q\|_2^2 \tag{20}$$

where $\alpha \geq 0$ is a regularization parameter and $\|\cdot\|_2$ denotes l^2-norm. Here, the first term $\|Z - VQ\|_2^2$ gives the fidelity to the available data Z and the second term bounds the power of this estimate by means of the regularization parameter α. In (20), and further, we omit the argument ω in the Fourier transform variables. We obtain the solution in the following form by minimizing (20):

$$\hat{Q} = \frac{\overline{V}}{|V|^2 + \alpha} Z \quad \hat{q}_\alpha(x) = F^{-1}\{\hat{Q}\} \tag{21}$$

where $\hat{Q}$ is an estimate of Q, $\widehat{q}_\alpha$ is an estimate of the true image q, and $\overline{V}$ denotes complex-conjugate value of V.

In this paper, we consider Gaussian, motion, and rectangular (boxcar) blurs. We aim to identify both the blur, which is characterized by PSF, and its parameter using a single neural network. Let us consider all these models and how they depend on the corresponding parameters.

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 [16]. For a variety of devices, such as photo or video camera, microscope, telescope, etc., PSFs are often approximated by the Gaussian function

$$v(t) = \frac{1}{2\pi\tau^2} \exp\left(-\frac{t_1^2 + t_2^2}{\tau^2}\right) \tag{22}$$

where τ^2 is a parameter of the PSF (the variance) [see Fig. 3(a)]. Its Fourier transform V is also a Gaussian function and its absolute values $|V|$ are shown in Fig. 3(b).

Another source of blur is a uniform linear motion which occurs while taking a picture of a moving object relatively to the camera

$$v(t) = \begin{cases} \dfrac{1}{h}, & \sqrt{t_1^2 + t_2^2} < h/2, t_1 \cos\phi = t_2 \sin\phi \\ 0, & \text{otherwise} \end{cases} \tag{23}$$

where h is a parameter which depends on the velocity of the moving object and describes the length of motion in pixels, and ϕ is the angle between the motion orientation and the horizontal axis. Any uniform function like (23) is characterized by the number of slopes in the frequency domain [Fig. 3(b) and (e)].

The uniform rectangular blur is described by the following function [Fig. 3(c)]:

$$v(t) = \begin{cases} \dfrac{1}{h^2}, & |t_1| < \dfrac{h}{2}, |t_2| < \dfrac{h}{2} \\ 0, & \text{otherwise} \end{cases} \tag{24}$$

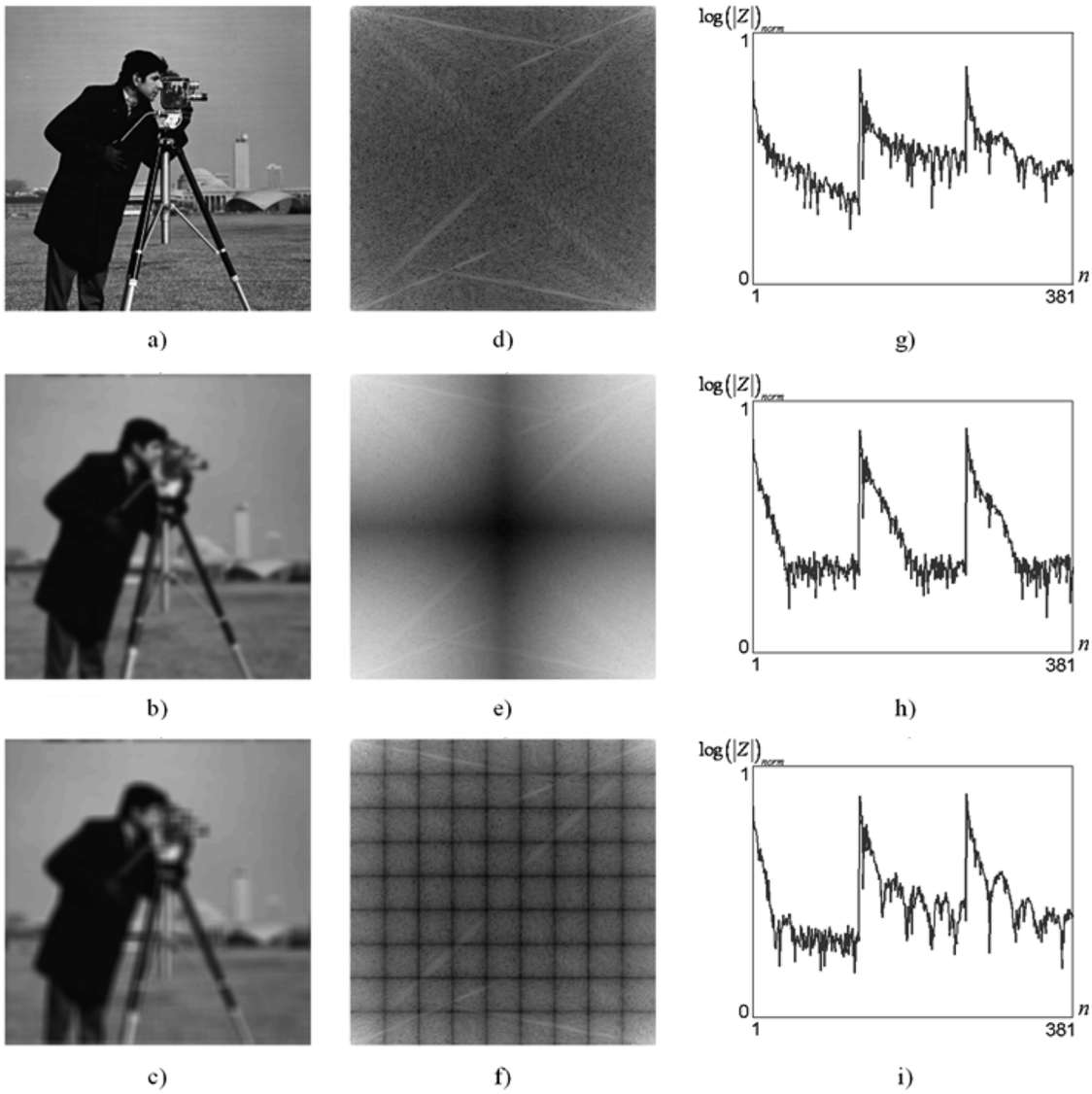


Fig. 4. True test *Cameraman* image (a) blurred by (b) Gaussian blur with $\tau = 2$; (c) boxcar blur of the size 9×9 $\log |Z|$ of the true test *Cameraman* image (d) blurred by (e) Gaussian blur with $\tau = 2$; (f) rectangular blur of the size 9×9. The normalized $\log |Z|$ values used as arguments to generate training vectors in (26) and (27) obtained from the true test *Cameraman* image (g) blurred by (h) Gaussian blur with $\tau = 2$; i) boxcar blur of the size 9×9.

where parameter h defines the size of smoothing area. The frequency characteristic of (24) is shown in Fig. 3(f).

In order to solve (21) one should know the PSF V. In this paper, we propose to use a neural network to recognize type and parameter of V from the noisy observation Z.

IV. Neural Network Simulation

A. Training Set

The observed image $z(t)$ is modeled as the output of a linear shift-invariant system (18), which is characterized by the PSF v. The PSF v, e.g., (22)–(24) that are used in this paper, has its own specific frequency characteristics. Hence, it is natural to use its spectral coefficients as features for both training and testing sets. Since originally the observation is not v (v is not known) but z, we use spectral coefficients of z as input training (and testing) vectors in order to identify the PSF v. Because this model in the frequency domain is the product of the true object function Q and the PSF V, we state the problem as recognition of the shape of V and its parameter from the power-spectral density (PSD) of the observation Z, i.e., from $|Z|^2 = Z \cdot \overline{Z}$, which in terms of statistical expectation can be rewritten as

$$E\{|Z|^2\} = E\{|QV + \varepsilon|^2\} = |Q|^2|V|^2 + \sigma^2 \qquad (25)$$

where σ^2 is the variance of noise in (19).

We use log values of $|Z|$ for pattern vectors X in (1). Examples of $\log |Z|$ values are shown in Fig. 4. The distortions of PSD for the test image Cameraman [Fig. 4(a)] that are typical for each type of blur [Fig. 4(b) and (c)] are clearly visible in Fig. 4(e) and (f).

For the sake of simplicity (but without loss of generality), we consider a square image $z(t)$, i.e., $L = L_1 = L_2$ in (18) and (19). In order to obtain the training vector $X = (x_1, \ldots, x_n)$ in

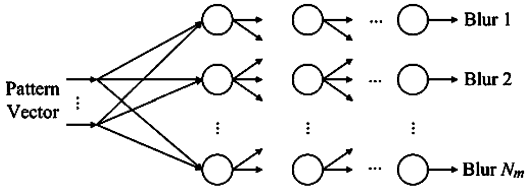


Fig. 5. Structure of the feedforward neural network used for both blur model and its parameter identification.

(20) as an input data of the network, and taking into account that the PSF v is symmetrical, PSD of $z(t)$ (25) is used as follows:

$$x_j = \exp\left(2\pi i \frac{\log(|Z(\omega_{k_1,k_2})|) - \log(|Z_{\min}|)}{\log(|Z_{\max}|) - \log(|Z_{\min}|)}\right) \tag{26}$$

where (27), shown at the bottom of the page, holds, and $Z_{\max} = \max_{k_1,k_2}(Z(\omega_{k_1,k_2}))$ and $Z_{\min} = \min_{k_1,k_2}(Z(\omega_{k_1,k_2}))$. Eventually, the length of the pattern vector is $n = 3L/2 - 3$.

Some examples of the values $\log|Z|$ normalized to be in the range $[0, 1[$ as follows:

$$\log(|Z|)_{\text{norm}} = \frac{\log(|Z(\omega_{k_1,k_2})|) - \log(|Z_{\min}|)}{\log(|Z_{\max}|) - \log(|Z_{\min}|)}$$

are shown in Fig. 4(g)–(i). They are used in (26) and (27) to produce the input training vectors X.

B. Neural Network Structure

In the following, we consider the complex multiclass identification problem where every class (blur model) also has a parameter to be identified. The number of neurons at the output layer N_m (see Section II-C) equals to the number of classes to be identified, and Ψ_i is a number of parameter's values for the ith class, $i = 1, \ldots, N_m$. Each output neuron has to classify simultaneously blur, a parameter of the corresponding type of blur, and to reject other blurs (as well as an unblurred image).

The MLMVN that we use here contains five neurons in the first hidden layer and 35 ones in the second hidden layer; this structure of the network has been selected experimentally. The training algorithm for smaller number of hidden neurons either requires more time because its convergence requires more iterations, or such a network yields poorer results. On the other hand, further increasing of the amount of hidden neurons does not lead to better results than for the selected network. Additionally, its training also requires more time because it is necessary to adjust the weights for more neurons. Therefore, the structure of network is $5 \rightarrow 35 \rightarrow N_m$ (Fig. 5). However, the structure may vary in other applications or with larger number of either models or parameters for the models.

The output level neuron has a specific structure (see Fig. 6). The range of values $[\exp(i \cdot 0) = 1, \exp(i \cdot 2\pi) = 1[$ of the activation function (4) is divided onto $\Psi_i + 1$ intervals, each used for identification of the blur parameter value, and one more interval is used to reject other blurs and unblurred images.

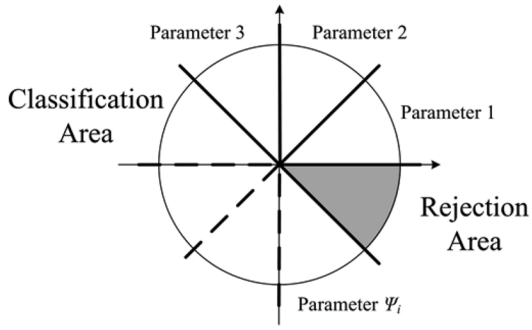


Fig. 6. Structure of the neural element on the output layer of MLMVN.

V. Results

A. Comparative Performance Analysis

Let us first test performance of the MLMVN solving the problem of identification of the Gaussian PSF parameter. Thus, the structure of MLMVN is $5 \rightarrow 35 \rightarrow 1$. The Gaussian blur is considered with $\tau \in \{1, 1.33, 1.66, 2, 2.33, 2.66, 3\}$ in (22), i.e., $\Psi_1 = 7$. The level of noise in (18) is selected to satisfy the blurred signal-to-noise ratio (BSNR) [18], [19] to be equal to 40 dB.

For a comparison, we have chosen well-known feedforward neural network with backpropagation training algorithm and the popular SVM approach. Fletcher–Reeves conjugate gradient [42] and scaled conjugate gradient [43] backpropagation training algorithms were used for training. They are implemented in MATLAB neural networks toolbox. For SVM modeling, we used MATLAB code available in [45].

In order to draw objective comparison, both standard feedforward neural networks used have the same structure as MLMVN, i.e., $5 \rightarrow 35 \rightarrow 1$. The input and hidden level neurons have tan–sigmoid activation function and the output layer neuron has linear activation function. For classification, the range of values $[-1, 1]$ of the activation function is divided onto eight intervals [seven intervals are reserved for the possible values of the blur parameter τ from (22), and one interval is reserved for rejection, similarly to MLMVN].

SVM-based classification was performed following the well-developed strategy to use a combination of several binary SVM classifiers to solve a given multiclass problem [46]. We used $\Psi_1 = 7$ SVMs, where each performs binary decision—positive and negative. Any positive answer corresponds to the correct classification of the parameter τ and the negative one corresponds to any rejected case.

$$\begin{cases} j = 1, \ldots, L/2 - 1, & \text{for } k_1 = k_2, \quad k_2 = 1, \ldots, L/2 - 1 \\ j = L/2, \ldots, L - 2, & \text{for } k_1 = 1, \quad k_2 = 1, \ldots, L/2 - 1 \\ j = L - 1, \ldots, 3L/2 - 3, & \text{for } k_2 = 1, \quad k_1 = 1, \ldots, L/2 - 1 \end{cases} \tag{27}$$

TABLE I
COMPARISON OF SPEED OF CONVERGENCE (ITER.) AND CLASSIFICATION RATES (CR) FOR GAUSSIAN BLUR PARAMETER IDENTIFICATION

	Network								
	Fletcher-Reeves Conjugate Gradient [42]			Scaled Conjugate Gradient [43]		SVM-based [45]		Continuous-valued MLMVN	
Runs	Iter.	CR	Train. Er.*	Iter.	CR	Iter.	CR	Iter.	CR
1	342	96.3 %	1 %	616	94.5 %	-	-	609	**99.5 %**
2	323	74.5 %	20 %	663	88.5 %	-	-	478	**98.7 %**
3	89	96.8 %	7 %	382	93.0 %	-	-	260	**98.5 %**
4	139	87.3 %	10.8 %	1329	98.5 %	-	-	126	**99.0 %**
5	515	90.8 %	7.1 %	893	89.25 %	-	-	423	**100. %**
Mean	**281.6**	89.1 %	9.2 %	776.6	92.75 %	-	97.5 %	379.2	**99.14 %**
Med	**323**	90.8 %	7.1 %	663	93.0 %	-	97.5 %	423	**99.0 %**
Std	± 171.2	± 9.1 %	± 7.0 %	± 358.15	± 4.08 %	-	± **0 %**	± 188.95	± 0.61 %

*The Training Error (column "Train. Er.") is provided only for Fletcher- Reeves Conjugate Gradient technique since all others are trained till 0 error.

The backpropagation training algorithm (11)–(15) is used to train MLMVN. The performance is evaluated in terms of speed of convergence and classification rate (CR). Speed of convergence is the number of iterations needed to train a network until MSE reaches the acceptable level, which is not necessarily 0. For instance, Fletcher–Reeves conjugate gradient [42] training algorithm stops when the gradient is 0, but it does not mean that the error is equal to 0.

We have used a database which consists of 150 grayscale images with sizes 256×256 to generate the training and testing sets. One hundred images are used to generate the training set and 50 other images are used to generate the testing set. The images with no blur and no noise are also included in both training and testing sets. As a result, $800 = 100 \cdot (7+1)$ pattern vectors X of the length 381 [see (26) and (27)] were used for training and $400 = 50 \cdot (7+1)$ for testing.

The trained network is used to perform classification on the testing set. The classification rate is computed as the number of correct classifications in terms of percentage

$$\mathrm{CR} = 100 \frac{N_{\mathrm{correct}}}{N_{\mathrm{total}}} (\%)$$

where N_{total} is a total number of pattern vectors X in the testing set, and N_{correct} is a number pattern vectors correctly classified by the trained network.

The initial weights for all the neural networks are randomly selected from the interval $[0, 1]$ (for the complex-valued weights, both their real and imaginary parts are selected in the same way). The numerical results are presented for all the networks for five independent runs in Table I. The best ones are highlighted by the bold face. The mean ("Mean" row), median ("Med" row), and standard deviation ("Std" row) values for the number of iterations and classification rate are given.

The SVM-based classification does not exploit random initialization and, therefore, only the result for the classification rate is presented.

It is clearly seen that the MLMVN provides CR that is very close to 100% and significantly outperforms a standard feedforward network. A standard MLF can be trained faster using the Fletcher–Reeves conjugate gradient method, but this training algorithm cannot ensure the zero error, and CR for the network trained using this method is the lowest, respectively. The MLMVN also outperforms the MLF trained using the scaled conjugate gradient algorithm with the zero error both in the terms of CR and the number of training epochs.

The SVM-based method provides CR $= 97.5\%$ that is closer to the MLMVN result CR $= 99.14\%$. However, the total number of support vectors in the SVM ensemble is significantly larger than the total number of the weights for all the neurons in the MLMVN. In fact, seven SVMs $(\Psi_1 = 7)$ use $(508 + 508 + 478 + 469 + 496 + 496 + 504)381 = 1317879$ support vectors in total, while the MLMVN uses $(381 + 1) \cdot 5 + 35 \cdot (5 + 1) + 1 \cdot (35 + 1) = 2156$ complex-valued weights in total.

It is worth to note that the MLMVN and SVM also show the comparable classification rates solving such benchmarks problems as "two spirals" and "sonar" [2], but the complexity of the MLMVN is drastically lower.

B. Multiple Blur Identification

For multiple models of blur, we provide the following two experiments. In the first experiment (Experiment 1), we consider six types of blur $(N_m = 6)$ with the following parameters:

- Gaussian blur is assumed with $\tau \in \{1, 1.33, 1.66, 2, 2.33, 2.66, 3\}$ in (22), the same as in Section V-A, $\Psi_1 = 7$;
- linear uniform horizontal $\phi = 0$ motion blur of the lengths 3, 5, 7, 9 in (23), $\Psi_2 = 4$;
- the data corrupted by the linear uniform vertical $\phi = 90$ motion blur of the length 3, 5, 7, 9 in (23), $\Psi_3 = 4$;
- the linear uniform diagonal motion from South–West to North–East blur [$\phi = 45$ in (23)] of the lengths 3, 5, 7, 9 in (23), $\Psi_4 = 4$;
- the linear uniform diagonal motion from South–East to North–West blur $(\phi = 135)$ of the lengths 3, 5, 7, 9 in (23), $\Psi_5 = 4$;
- rectangular has sizes 3×3, 5×5, 7×7, and 9×9 in (24), $\Psi_6 = 4$.

Because we consider six types of blur $(N_m = 6)$, the output layer contains six neurons. As mentioned previously, the structure of the MLMVN is two hidden layers with five and 35 neurons, respectively, and the output layer. Therefore, the structure of network is $5 \rightarrow 35 \rightarrow 6$.

Each neural element of the output layer has to classify a parameter of the corresponding type of blur, and reject other blurs, as well as the unblurred image. For instance, the first neuron is

TABLE II
CR FOR BLUR IDENTIFICATION

Blur	[12],[33]	Discrete-valued MLMVN [36]	SVM-based classification	Continuous-valued MLMVN	
				Exp. 1	Exp. 2*
No blur	-	**100. %**	**100. %**	96.0 %	90.0 %
Gaussian	93.5 %	98.7 %	**99.4 %**	99.0 %	85.0 %
Rectangular	95.6 %	97.9 %	96.4 %	**98.0 %**	-
Motion Horizontal	98.1 %	97.8 %	96.4 %	**98.5 %**	-
Motion Vertical	-	97.2 %	96.4 %	**98.3 %**	-
Motion North-East Diagonal	-	-	96.5 %	**97.9 %**	-
Motion North-West Diagonal	-	-	96.5 %	**97.2 %**	-

* Experiment 1 and Experiment 2 are not comparable to each other because they simulate the different problems (see Section V.B)

used to identify the Gaussian blur and to reject the non-Gaussian ones. If the weighted sum for the first neuron at the output (third) layer hits the jth interval, $j \in \{1, \ldots, 7\}$, then the input vector X corresponds to the Gaussian blur and its parameter is τ_j.

We have used the same initial database of 150 different grayscale images with sizes 256×256, which has been used to generate the training and testing sets in Section V-A. As well as previously, 100 images are used to generate the training set and 50 other images are used to generate the testing set. Because we consider here six types of blur, five of them with the four parameter values and one with the seven parameter values, then along with the class of clean images, we have $5 \cdot 4 + 7 + 1 = 28$ classes. Eventually, the training set consists of $2800 = 28 \cdot 100$ pattern vectors, and the testing set consists of $1400 = 28 \cdot 50$ pattern vectors. The level of noise in (18) is selected satisfying BSNR to be equal to 40 dB.

The trained network is used to perform the classification on the testing set. The CR is used as an objective criterion of classification. The results are presented in Table II (Experiment 1 column). The first row corresponds to the recognition of the original nonblurred images. All output layer neurons should classify them as they are not distorted by any of the considered types of blur.

Other rows present the results for blurred images classification and identification a parameter of a blurring function. The results for six types of blur are better or comparable with those presented in [12], [35], and [36]. The best ones are highlighted by the bold font. It was succeeded for the first time to classify six blurs (compared to three in [12] and [35] and four in [36]). It is important to mention that 16 384 spectral coefficients have been used in [12], and [35] as the features and the network inputs, respectively, while here we use only 381 spectral coefficients as features and network inputs.

The SVM-based classification is designed in a similar manner as in Section V-A. We have used a combination of $\sum_{i=1}^{N_m} \Psi_i = 7 + 4 + 4 + 4 + 4 + 4 = 27$ binary decision SVM classifiers. The results are presented in Table II as well. It is seen that they are similar to those provided in Table I, that is, they are comparable to the results for MLMVN. However, the complexity of the SVM ensemble, which is used for solving this multiclass problem, is significantly higher than the one of the MLMVN. Each SVM uses approximately $2500 \cdot 381 = 952\,500$ support vectors, so there are about $952\,500 \cdot 27 = 25\,717\,500$ support vectors in total, while the MLMVN uses $(381 + 1) \cdot 5 + 35 \cdot (5 + 1) + 6 \cdot (35 + 1) = 2336$ weights in total. Because the standard MLF shows significantly worse results in comparison to the MLMVN for a simpler problem in Section V-A (Table I), we do not provide here results for the multiple-blur identification with MLF.

TABLE III
COMPARISON OF SPEED OF CONVERGENCE (ITER.) AND CLASSIFICATION RATES (CR) FOR GAUSSIAN BLUR PARAMETER IDENTIFICATION OF A STANDARD REAL-VALUED MLF WITH EXTENDED TOPOLOGY $10 \rightarrow 71 \rightarrow 1$

Runs	Network				
	Fletcher-Reeves Conjugate Gradient [42]			Scaled Conjugate Gradient [43]	
	Iter.	CR	Train. Er.	Iter.	CR
1	347	88.7 %	7.1 %	413	90.8 %
2	258	93.2 %	2.1 %	535	88.5 %
3	144	89.7 %	3.0 %	602	92.3 %
4	120	96.5 %	2.9 %	341	98.3 %
5	183	91.7 %	4.5 %	378	92.0 %
Mean	**210.4**	92.0 %	3.8 %	453.8	92.4 %
Med	**183**	91.7 %	2.9 %	413	92.0 %
Std	± 92.5	± 3.06 %	± 2.3 %	± 110.3	± 3.6 %

In order to avoid possible speculations on the complexity of complex-valued and real-valued neural networks, the results of Gaussian blur identification for a standard real-valued MLF with extended to $10 \rightarrow 71 \rightarrow 1$ topology using the Fletcher–Reeves conjugate gradient [42] and the scaled conjugate gradient methods [43] are shown in Table III. Simulations are performed similarly to those given in Table I. It is seen that the improvement of the results is not significant and the CR for the twice larger MLF is still lower than for the MLMVN.

The results of using the MLMVN classification for image reconstruction are shown in Fig. 7 for the test Cameraman image. The adaptive deconvolution technique proposed in [18] has been used after the blur and its parameter identified.[1] The image was blurred by the Gaussian PSF (22) with $\tau = 2$. It is seen that if classified PSF coincides with the true PSF then the value of improved signal-to-noise ratio (ISNR) [18] criterion is 3.88 dB. If the image is blurred using $\tau = 1.835$ or $\tau = 2.165$ then the network classifies them as blurred with $\tau = 2$ and reconstruction is applied using the recognized value. Then, the error of reconstruction is approximately 0.6 dB below the accurate value.

In order to reduce this error we propose to consider Experiment 2. We are targeting here classification of a single Gaussian

[1] MATLAB code for this technique is available following the link http://www.cs.tut.fi/~lasip/

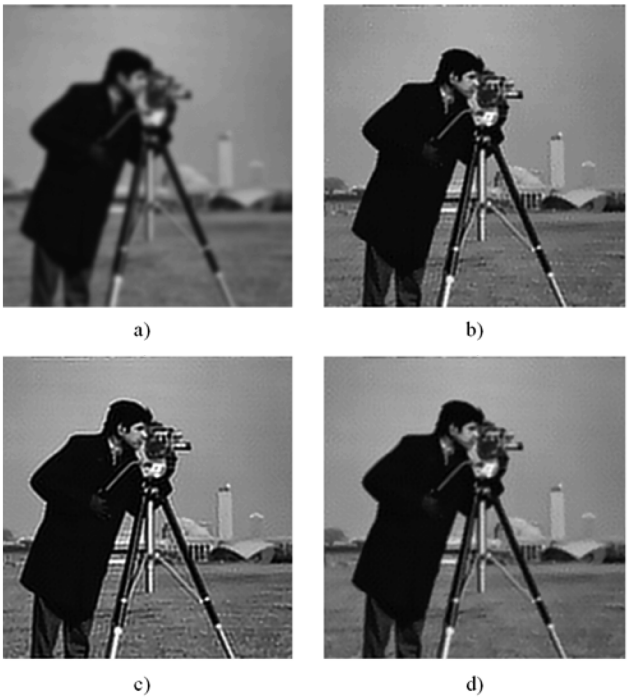


Fig. 7. (a) Test noisy blurred *Cameraman* image with Gaussian PSF $\tau = 2$ (b) reconstructed using the regularization technique [18] after the blur and its parameter has been identified as Gaussian PSF with $\tau = 2$ (ISNR = 3.88 dB); (c) the original Cameraman image blurred by the Gaussian PSF with $\tau = 1.835$ [this blurred image does not differ visually from the one in Fig. 7(a)] and then reconstructed using the regularization technique [18] after the blur and its parameter has been identified as Gaussian PSF with $\tau = 2$ (ISNR = 3.20 dB); (d) the original Cameraman image blurred by Gaussian PSF with $\tau = 2.165$ [this blurred image does not differ visually from the one in Fig. 7(a)] and then reconstructed using the regularization technique [18] after the blur and its parameter has been identified as Gaussian PSF with $\tau = 2$ (ISNR = 3.22 dB).

blur type, but with much higher precision. The grid of the blur's parameters is finer with significantly larger number of them in the same interval $\tau \in \{1 + 0.15\Delta : \Delta = 0, 1, \ldots, 14\}$ in (22), which makes the problem of classification more challenging. The output layer of the network contains in this case a single neuron, and the network structure is $5 \rightarrow 35 \rightarrow 1$.

The results of classification are presented in Table II (Experiment 2 column). It is evident that the error of classification is formally higher. Nevertheless, it is very important that the reconstruction error for the similar experiment as shown in Fig. 7 does not exceed 0.1 dB, which is a minor value in practice. During the reconstruction simulation, we used the images that have been blurred with $\tau = 2 - 0.15/2 = 1.925$ and $\tau = 2 + 0.15/2 = 2.075$, while the reconstruction has been done as for $\tau = 2$.

It is worth to note that time spent on the network training for Experiment 1 was about 24 h on a computer with Pentium 4 3.2-GHz central processing unit (CPU) and 45 min for Experiment 2.

VI. Conclusion

In this paper, we have proposed a novel technique for blur identification using a single observed image. The technique employs a continuous-valued feedforward multilayer neural network with multivalued neurons (MLMVN), which is trained for a database of images. Then, this network is used to identify both the type and the parameters of the blur. This identification procedure is computationally fast and cheap. The error backpropagation algorithm for the MLMVN is derived for the most general case of an arbitrary number of output layer neurons. It is confirmed that the MLMVN is a powerful tool for solving the classification problems, especially the multiclass ones. The obtained results show the high efficiency of the proposed approach. It is shown by simulations that this network can be used as an efficient estimator of PSF, whose precise identification is of crucial importance for image deblurring.

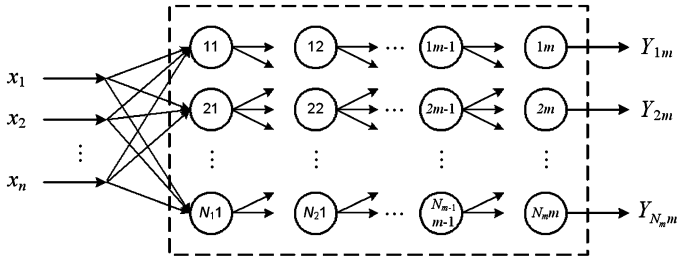


Fig. 8. Feedforward neural network.

Appendix I

A backpropagation training algorithm for the MLMVN is analogous to training of a single MVN. First, this algorithm for the MLMVN with a single output neuron and a single hidden layer has been derived in [2]. In this paper, we would like to present the derivation of the MLMVN error backpropagation for the most general case of the network with $m-1$ hidden layers and one output layer with an arbitrary number of neurons in each.

The MVN learning is reduced to the movement along the unit circle and it is based on the error-correction learning rule. Thus, the correction of the weights is completely determined by the neuron's error. The same property is true not only for the single MVN, but also for the whole MLMVN. The errors of all neurons from the MLMVN are completely determined by the global errors of the network (8). Let the MLMVN contains m layers of neurons: $m-1$ hidden layers and one output layer, the mth one (see Fig. 8).

The errors of the network that are taken from the N_m output neurons are equal to $\delta^*_{km} = T_{km} - Y_{km}, k = 1, \ldots, N_m$, where N_m is the number of neurons in the mth layer (the output one), T_{km} is a desired output of the kmth output neuron, and Y_{km} is its actual output. It is crucial to understand how the errors for each particular neuron are calculated, propagating the errors δ^*_{km} back from the mth layer to the first layer.

Let $(w_0^{km}, w_1^{km}, \ldots, w_{N_{m-1}}^{km})$ be an initial weighting vector of the kmth output neuron, $Y_{i,m-1}$ be an initial output of the neuron $i, m-1$ from the last hidden layer $(i = 1, \ldots, N_{m-1})$, z_{km} be the weighted sum on the kmth output neuron before the correction of the weights, δ_{im-1} be the error of the neuron $i, m-1$ from the hidden layer $(i = 1, \ldots, N_{m-1})$, and δ_{km} be the error of the kmth output neuron. Without loss of generality, let us suppose that all neurons from the first hidden layer until the $(m-1)$th layer are already trained, and we have to train the neurons from the output layer. The learning rule (5) is used for the correction of the weights. Let us suppose that the errors

for all the neurons of the network are already known. Then, the weights for the kmth output neuron are corrected as in (13) and the weighted sum is derived as in (28), shown at the bottom of the page, where $\bar{Y}$ is the complex number complex conjugated to Y.

Taking into account that $\forall i = 1, \ldots, N_{m-1}(Y_{i,m-1} + \delta_{i,m-1})$ is an output of the neuron, this complex number is always lying on the unit circle and, therefore, its absolute value is always equal to 1. This means that $\forall i = 1, \ldots, N_{m-1}, \overline{(Y_{i,m-1} + \delta_{i,m-1})}(Y_{i,m-1} + \delta_{i,m-1}) = 1$.

Then, we obtain the weighted sum for kmth neuron as in (29), shown at the bottom of the page.

Thus, finally

$$\tilde{z}_{km} = z_{km} + \delta_{km} + \sum_{i=1}^{N_{m-1}} w_i^{km}\delta_{i,m-1}. \tag{30}$$

To ensure the weighted sum of the kmth output neuron is exactly equal to $z_{km} + \delta_{km}^*$ after the correction procedure, it follows from (30) that we need to satisfy

$$\delta_{km} + \sum_{i=1}^{N_{m-1}} w_i^{km}\delta_{i,m-1} = \delta_{km}^*. \tag{31}$$

Equation (31) as a formal equation has an infinite number of solutions, while we have to find among them a single solution which will correctly represent the local errors of each neuron through the global error of the network and through the errors of the neurons of the preceding layer.

Let us come back to the learning rule (5) for the single MVN. In this rule, $\Delta W = (C_r/(n+1))(\varepsilon^q - z/|z|)\bar{X}$. It contains a factor $1/(n+1)$ in order to distribute the error $\varepsilon^q - z/|z|$ uniformly among all $n+1$ weights $w_0, w_1, \ldots, w_n$. As it was mentioned previously, if we omitted this factor, then the corrected weighted sum would not be equal to $z + \delta$ as expected, but to $z + (n+1)\delta$. Because all the inputs are equitable, both the error and ΔW are distributed among the weights uniformly during the correction procedure. However, a feedforward network does not have to differ from a single neuron at this point because the network in general performs in the same way as the single neuron. This means that the global error of the network consists not only of the output neurons error, but of the local errors of the output neurons and hidden neurons connected to the output neurons. It means that to obtain the local errors for all neurons, the global error must be distributed among these neurons. We can assume without loss of generality that this distribution is uniform.

Thus, we obtain from (31) the following:

$$\delta_{km} = \frac{1}{(N_{m-1}+1)}\delta_{km}^* \tag{32}$$

and

$$w_i^{km}\delta_{i,m-1} = \frac{1}{(N_{m-1}+1)}\delta_{km}^*, \qquad i = 1, \ldots, N_{m-1}. \tag{33}$$

$$\begin{aligned}
\tilde{z}_{km} &= \left(w_0^{km} + \frac{1}{(N_{m-1}+1)}\delta_{km}\right) + \left(w_1^{km} + \frac{1}{(N_{m-1}+1)}\delta_{km}\overline{(Y_{1,m-1}+\delta_{1,m-1})}\right)(Y_{1,m-1}+\delta_{1,m-1}) + \cdots \\
&\quad + \left(w_n^{km} + \frac{1}{(N_{m-1}+1)}\delta_{km}\overline{(Y_{N_{m-1},m-1}+\delta_{N_{m-1},m-1})}\right)(Y_{N_{m-1},m-1}+\delta_{N_{m-1},m-1}) \\
&= \left(w_0^{km} + \frac{1}{(N_{m-1}+1)}\delta_{km}\right) + \sum_{i=1}^{N_{m-1}}\left(w_i^{km} + \frac{1}{(N_{m-1}+1)}\delta_{km}\overline{(Y_{N_{m-1},m-1}+\delta_{i,m-1})}\right)(Y_{i,m-1}+\delta_{i,m-1}) \\
&= \left(w_0^{km} + \frac{1}{(N_{m-1}+1)}\delta_{km}\right) + \sum_{i=1}^{N_{m-1}}\left(w_i^{km}Y_{i,m-1} + w_i^{km}\delta_{i,m-1} + \frac{1}{(N_{m-1}+1)}\delta_{km}\overline{(Y_{i,m-1}+\delta_{i,m-1})}(Y_{i,m-1}+\delta_{i,m-1})\right)
\end{aligned} \tag{28}$$

$$\begin{aligned}
\tilde{z}_{km} &= \left(w_0^{km} + \frac{1}{(N_{m-1}+1)}\delta_{km}\right) + \sum_{i=1}^{N_{m-1}}\left(w_i^{km}Y_{i,m-1} + w_i^{km}\delta_{i,m-1} + \frac{1}{(N_{m-1}+1)}\delta_{km}\overline{(Y_{i,m-1}+\delta_{i,m-1})}(Y_{i,m-1}+\delta_{i,m-1})\right) \\
&= \left(w_0^{km} + \frac{1}{(N_{m-1}+1)}\delta_{km}\right) + \sum_{i=1}^{N_{m-1}}\left(w_i^{km}Y_{i,m-1} + w_i^{km}\delta_{i,m-1} + \frac{1}{(N_{m-1}+1)}\delta_{km}\right) \\
&= w_0^{km} + \frac{1}{(N_{m-1}+1)}\delta_{km} + \sum_{i=1}^{N_{m-1}} w_i^{km}Y_{i,m-1} + \sum_{i=1}^{N_{m-1}} w_i^{km}\delta_{i,m-1} + \frac{N_{m-1}}{(N_{m-1}+1)}\delta_{km} \\
&= \underbrace{w_0^{km} + \sum_{i=1}^{N_{m-1}} w_i^{km}Y_{i,m-1}}_{z_{km}} + \delta_{km} + \sum_{i=1}^{N_{m-1}} w_i^{km}\delta_{i,m-1} \\
&= z_{km} + \delta_{km} + \sum_{i=1}^{N_{m-1}} w_i^{km}\delta_{i,m-1}.
\end{aligned} \tag{29}$$

However, from (32) $\delta^*_{km} = (N_{m-1}+1)\delta_{km}$ and substituting this expression into (33), we obtain

$$w_i^{km}\delta_{i,m-1} = \delta_{km}, \qquad i = 1,\dots,N_{m-1}. \tag{34}$$

Hence, for the error $\delta_{i,m-1}$ of the neuron $(i, m-1)$, we obtain the following:

$$\delta_{i,m-1} = (w_i^{km})^{-1}\delta_{km}, \qquad i = 1,\dots,N_{m-1}. \tag{35}$$

Let us now substitute $\delta_{i,m-1}, j = 1,\dots,N_{m-1}$ and δ_{km} from (32) and (33), respectively, into (30)

$$\begin{aligned}\tilde{z}_{km} &= z_{km} + \delta_{km} + \sum_{i=1}^{N_{m-1}} w_i^{km}\delta_{i,m-1} \\ &= z_{km} + \delta_{km} + \sum_{i=1}^{N_{m-1}} w_i^{km}(w_i^{km})^{-1}\delta_{km} \\ &= z_{km} + \delta_{km} + N_{m-1}\delta_{km} \\ &= z_{km} + (N_{m-1}+1)\delta_{km} = z_{km} + \delta^*_{km}. \end{aligned}\tag{36}$$

We obtain exactly the result that is our target $\tilde{z}_{km} = z_{km} + \delta^*_{km}$. It is important to mention that (36) corresponds to $\tilde{z} = z+\delta$ for the single MVN.

It follows from (32) that if in general the error of a neuron on the layer j is equal to $\tilde{\delta}$, this $\tilde{\delta}$ must contain a factor equal to $1/s_j$, where $s_j = N_{j-1}+1$ is the number of neurons whose outputs are connected to the inputs of the considered neuron incremented by 1 (the considered neuron itself). This ensures distribution of the error among all the neurons on whose errors the error of the considered neuron depends. In other words, the error of each neuron is distributed uniformly among the neurons connected to it and itself. As it was mentioned previously, for the first hidden layer, $s_1 = 1$ because there is no preceding hidden layer, and there are no neurons, among which the error may be distributed, respectively.

Equation (35) also shows that during a backpropagation procedure the backpropagated error must be multiplied by the inverse values of the corresponding weights. The weights are complex numbers and, therefore, $w^{-1} = \bar{w}/|w|^2$.

Now, it is easy to generalize everything that we considered for the output layer neurons to all the hidden layers with an arbitrary number of neurons in each. We obtain the following. The global errors of the whole network are determined by (8). For the errors of the mth (output) layer neurons, the local errors are determined by (11)

$$\delta_{km} = \frac{1}{s_m}\delta^*_{km}$$

where km specifies a kth neuron of the mth layer and $s_m = N_{m-1}+1$.

For the errors of the hidden layers neurons, the local errors are determined by (12)

$$\delta_{kj} = \frac{1}{s_j}\sum_{i=1}^{N_{j+1}} \delta_{i,j+1}\left(w_k^{i,j+1}\right)^{-1}$$

where $\{kj\}$ specifies a kth neuron of the jth layer $(j = 1,\dots,m-1), s_j = N_{j-1}+1, j = 2,\dots,m$, and $s_1 = 1$.

Thus, (11) and (12) determine the error backpropagation for the MLMVN for the most general case of the network with $m-1$ hidden layers and one output layer with an arbitrary number of neurons in each. In contrast to the error backpropagation for the MLF, the error backpropagation for the MLMVN is derivative-free.

APPENDIX II

Let us illustrate in details, how the presented derivative-free error backpropagation algorithm can be used to train the MLMVN.

Initialization: Suppose we wish to classify three training vectors X_1, X_2, and X_3 of the length 2 to be members of three different classes $\tilde{T}_1, \tilde{T}_2$, and $\tilde{T}_3$

$$\begin{aligned} X_1 &= (\exp(4.23i), \exp(2.10i)) \to \tilde{T}_1 \\ X_2 &= (\exp(5.34i), \exp(1.24i)) \to \tilde{T}_2 \\ X_3 &= (\exp(2.10i), \exp(0i)) \to \tilde{T}_3. \end{aligned}$$

Classes $\tilde{T}_1, \tilde{T}_2$, and $\tilde{T}_3$ are determined in such a way that the argument of the desired output of the network must belong to the interval $[\arg(T_j) - 0.05, \arg(T_j) + 0.05], j = 1,2,3$, where

$$\begin{aligned} T_1 &= \exp(0.76i) \\ T_2 &= \exp(2.56i) \\ T_3 &= \exp(5.35i). \end{aligned}$$

This corresponds to the condition (17) $|\arg(T_j) - \arg(e^{i\mathrm{Arg}\,z})| \le 0.05$, where $e^{i\mathrm{Arg}\,z}$ is the actual output. The mean square error (16) is $E \le 0.05^2 = 0.0025$.

Let the MLMVN have the structure $2 \to 1$, i.e., it has a single hidden layer with two neurons and the output layer containing a single neuron. The initial weights (both real and imaginary parts) of the MLMVN are taken as random numbers from the interval $[0,1]$

Neuron (1,1)	Neuron (1,2)	Neuron (2,1)
$w_0 = 0.23 - 0.38i$	$w_0 = 0.23 - 0.38i$	$w_0 = 0.23 - 0.38i$
$w_1 = 0.19 - 0.46i$	$w_1 = 0.19 - 0.46i$	$w_1 = 0.19 - 0.46i$
$w_2 = 0.36 - 0.33i$	$w_2 = 0.36 - 0.33i$	$w_2 = 0.36 - 0.33i$

Let t be the index of training epoch and j be the index of the training vector, so, initially $t = 1$ and $j = 1$. The MLMVN training is performed as follows.

Iteration t:

1) *Compute the weighting sum* $z = w_0 + w_1x_1 + w_2x_2$ for each neuron from the hidden layer for the first $(j = 1)$ pattern vector $X_j = (x_1, x_2)$, where $x_1 = \exp(4.23i) = -0.46 - 0.88i$ and $x_2 = \exp(2.10i) = -0.51 + 0.85i$

$$\begin{aligned} z_{11} &= (0.23 - 0.38i) + (0.19 - 0.46i)(-0.46 - 0.88i) \\ &\quad + (0.36 - 0.33i)(-0.51 + 0.85i) \\ &= -0.16 + 0.13i. \end{aligned}$$

It is easy to see that $z_{12} = z_{11} = -0.16 + 0.13i$ in this particular case.

2) *Compute the outputs for the hidden layer neurons*: According to (4), the actual outputs are $Y_{11} = z_{11}/|z_{11}| = -0.78 + 0.62i$ and $Y_{12} = z_{12}/|z_{12}| = -0.78 + 0.62i$. They are used as inputs to the output neuron $(2,1)$. Then,

TABLE IV
EXAMPLE OF THE 2 → 1 MLMVN TRAINING

Epoch t	Pattern vector j	Weights $w=(w_0,w_1,w_2)$ for neuron (2,1)	Actual Output Y_{21}	Error (17) (≤ 0.05)	MSE (16) (≤ 0.0025)
1	1	$w=(0.23-0.38i, 0.19-0.46i, 0.36-0.33i)$	0.45+0.89i	0.342	2.4213
	2	$w=(0.26-0.41i, 0.20-0.50i, 0.38-0.36i)$	0.39-0.92i	2.553	
	3	$w=(0.12-0.24i, 0.07-0.67i, 0.18-0.43i)$	-0.19-0.98i	0.829	
2	1	$w=(0.21-0.22i, 0.16-0.63i, 0.27-0.40i)$	0.83+0.55i	0.174	0.1208
	2	$w=(0.20-0.21i, 0.18-0.64i, 0.29-0.41i)$	-1.00+0.00i	0.581	
	3	$\boldsymbol{w=(0.21-0.15i, 0.13-0.68i, 0.30-0.47i)}$	**0.57-0.82i**	**0.030**	
3	1	$w=(0.21-0.15i, 0.13-0.68i, 0.30-0.47i)$	0.57+0.83i	0.209	0.2872
	2	$w=(0.23-0.16i, 0.11-0.69i, 0.28-0.47i)$	-0.10+0.99i	0.888	
	3	$w=(0.15-0.21i, 0.19-0.73i, 0.36-0.41i)$	0.46-0.89i	0.160	
4	1	$w=(0.16-0.20i, 0.21-0.72i, 0.38-0.41i)$	0.19+0.98i	0.619	0.1486
	2	$\boldsymbol{w=(0.22-0.24i, 0.16-0.77i, 0.32-0.44i)}$	**-0.85+0.53i**	**0.024**	
	3	$w=(0.22-0.24i, 0.16-0.77i, 0.32-0.44i)$	0.37-0.93i	0.259	
5	1	$w=(0.25-0.22i, 0.18-0.75i, 0.35-0.44i)$	0.78+0.63i	0.080	0.0049
	2	$\boldsymbol{w=(0.24-0.21i, 0.19-0.75i, 0.35-0.44i)}$	**-0.82+0.57i**	**0.025**	
	3	$w=(0.24-0.21i, 0.19-0.75i, 0.35-0.44i)$	0.53-0.85i	0.080	
6	1	$w=(0.25-0.21i, 0.20-0.74i, 0.36-0.43i)$	0.68+0.73i	0.060	0.0026
	2	$\boldsymbol{w=(0.25-0.21i, 0.19-0.75i, 0.36-0.44i)}$	**-0.82+0.58i**	**0.034**	
	3	$w=(0.25-0.21i, 0.19-0.75i, 0.36-0.44i)$	0.55-0.83i	0.052	
7	**1**	$\boldsymbol{w=(0.26-0.21i, 0.19-0.74i, 0.36-0.43i)}$	**0.71+0.71i**	**0.025**	**0.0009**
	2	$\boldsymbol{w=(0.26-0.21i, 0.19-0.74i, 0.36-0.43i)}$	**-0.81+0.58i**	**0.039**	
	3	$\boldsymbol{w=(0.26-0.21i, 0.19-0.74i, 0.36-0.43i)}$	**0.58-0.82i**	**0.022**	

the output Y_{12} for the neuron $(2,1)$ is computed according to (4) as follows:

$$z_{21} = (0.23-0.38i) + (0.19-0.46i)(-0.78+0.62i) + (0.36-0.33i)(-0.78+0.62i) = 0.29+0.57i$$

and finally $Y_{21} = z_{21}/|z_{21}| = 0.45+0.89i$.

3) *Check condition (17)*: The argument of the output Y_{21} is compared to the argument of the desired output $T_{12} = T_j$

$$|\arg(Y_{21}) - \arg(T_{12})| = |1.10-0.76| = 0.34 > 0.05.$$

The condition does not hold and the training must be continued.

4) *Weights update*: The network error at the output neuron is $\delta_{12}^* = 0.27-0.21i$. The error (11) for the output neuron is $\delta_{12} = 0.09-0.07i$ and the errors (12) for the hidden layer neurons are $\delta_{11} = 0.88+0.53i$ and $\delta_{12} = 1.04+0.11i$. Weight corrections can be performed using (15) for the hidden layer and (13) for the output layer that results in the following weights:

$$W=(0.52-0.21i, -0.10-0.28i, 0.36-0.67i), \quad \text{for } (1,1)$$
$$W=(0.58-0.35i, 0.00-0.17i, 0.21-0.65i), \quad \text{for } (1,2)$$
$$W=(0.26-0.41i, 0.20-0.50i, 0.38-0.36i), \quad \text{for } (2,1).$$

5) *Completion of the training epoch*: Repeat steps 1–4 for all pattern vectors (till $j=3$).

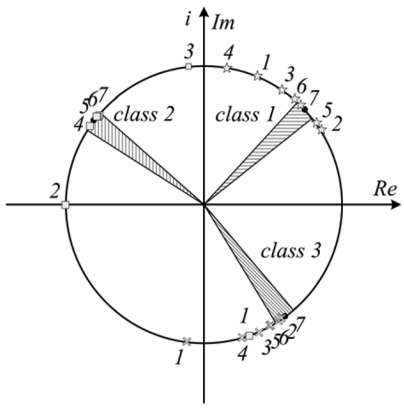


Fig. 9. Update of the 2 → 1 MLMVN output for three classes at every iteration.

6) *Termination of training*: Compute MSE according to (10) and check the condition (16). If it holds, the learning process has converged. If it does not hold then increase t and perform steps 1–5 again. This process should continue until the condition (16) is satisfied.

The process is illustrated in detail in Table IV. The rows highlighted by the bold font indicate that the MLMVN for the particular training vector X_j has reached the desired precision. For better visualization, Fig. 9 illustrates how the outputs of the network are distributed for all seven iterations. It is clearly visible that the training error is decreasing very quickly and the actual outputs approach the desired ones with the quickly increasing precision starting from the fourth training epoch.

REFERENCES

[1] I. Aizenberg, C. Moraga, and D. Paliy, "A feedforward neural network based on multi-valued neurons," in *Computational Intelligence, Theory and Applications. Advances in Soft Computing*, B. Reusch, Ed. Berlin, Germany: Springer, 2005, pp. 599–612, XIV.

[2] I. Aizenberg and C. Moraga, "Multilayer feedforward neural network based on multi-valued neurons (MLMVN) and a backpropagation learning algorithm," *Soft Comput.*, vol. 11, no. 2, pp. 169–183, Jan. 2007.

[3] N. N. Aizenberg, Y. L. Ivaskiv, and D. A. Pospelov, "About one generalization of the threshold function," (in Russian) *Doklady Akademii Nauk SSSR (The Reports of the Academy of Sciences of the USSR)*, vol. 196, no. 6, pp. 1287–1290, 1971.

[4] N. N. Aizenberg and I. N. Aizenberg, "CNN based on multi-valued neuron as a model of associative memory for gray-scale images," in *Proc. 2nd IEEE Int. Workshop Cellular Neural Netw. Their Appl.*, Germany, Oct. 14–16, 1992, pp. 36–41.

[5] N. N. Aizenberg and Y. L. Ivaskiv, *Multiple-Valued Threshold Logic* (in Russian). Kiev, Ukraine: Naukova Dumka Publisher House, 1977.

[6] I. Aizenberg, N. Aizenberg, and J. Vandewalle, *Multi-Valued and Universal Binary Neurons: Theory, Learning, Applications*. Boston, MA: Kluwer, 2000.

[7] S. Jankowski, A. Lozowski, and J. M. Zurada, "Complex-valued multistate neural associative memory," *IEEE Trans. Neural Netw.*, vol. 7, no. 6, pp. 1491–1496, Nov. 1996.

[8] H. Aoki and Y. Kosugi, "An image storage system using complex-valued associative memory," in *Proc. 15th Int. Conf. Pattern Recognit.*, Barcelona, Spain, 2000, vol. 2, pp. 626–629.

[9] M. K. Muezzinoglu, C. Guzelis, and J. M. Zurada, "A new design method for the complex-valued multistate Hopfield associative memory," *IEEE Trans. Neural Netw.*, vol. 14, no. 4, pp. 891–899, Jul. 2003.

[10] H. Aoki, E. Watanabe, A. Nagata, and Y. Kosugi, "Rotation-invariant image association for endoscopic positional identification using complex-valued associative memories," in *Lecture Notes in Computer Science*, J. Mira and A. Prieto, Eds. Berlin, Germany: Springer-Verlag, 2001, vol. 2085, Bio-inspired Applications of Connectionism, pp. 369–374.

[11] I. Aizenberg, E. Myasnikova, M. Samsonova, and J. Reinitz, "Temporal classification of drosophila segmentation gene expression patterns by the multi-valued neural recognition method," *J. Math. Biosci.*, vol. 176, no. 1, pp. 145–159, 2002.

[12] I. Aizenberg, T. Bregin, C. Butakoff, V. Karnaukhov, N. Merzlyakov, and O. Milukova, "Type of blur and blur parameters identification using neural network and its application to image restoration," in *Lecture Notes in Computer Science*, J. R. Dorronsoro, Ed. Berlin, Germany: Springer-Verlag, 2002, vol. 2415, pp. 1231–1236.

[13] W. K. Pratt, *Digital Image Processing*, 2nd ed. New York: Wiley, 1992.

[14] J. G. Nagy and D. P. O'Leary, "Restoring images degraded by spatially variant blur," *SIAM J. Sci. Comput.*, vol. 19, no. 4, pp. 1063–1082, Jul. 1998.

[15] C. Rushforth, "Signal restoration, functional analysis, and Fredholm integral equations of the first kind," in *Image Recovery: Theory and Application*. New York: Academic, 1987.

[16] E. Hecht, *Optics*, 4th ed. Reading, MA: Addison-Wesley, 2002.

[17] A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-Posed Problems*. New York: Wiley, 1977.

[18] V. Katkovnik, K. Egiazarian, and J. Astola, "A spatially adaptive nonparametric image deblurring," *IEEE Trans. Image Process.*, vol. 14, no. 10, pp. 1469–1478, Oct. 2005.

[19] R. Neelamani, H. Choi, and R. G. Baraniuk, "Forward: Fourier-wavelet regularized deconvolution for ill-conditioned systems," *IEEE Trans. Signal Process.*, vol. 52, no. 2, pp. 418–433, Feb. 2003.

[20] M. Figueiredo and R. Nowak, "An EM algorithm for wavelet-based image restoration," *IEEE Trans. Image Process.*, vol. 12, no. 8, pp. 906–916, Aug. 2003.

[21] R. L. Lagendijk, J. Biemond, and D. E. Boekee, "Identification and restoration of noisy blurred images using the expectation-maximization algorithm," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 38, no. 7, pp. 1180–1191, Jul. 1990.

[22] G. B. Giannakis and R. W. Heath, "Blind identification of multichannel FIR blurs and perfect image restoration," *IEEE Trans. Image Process.*, vol. 9, no. 11, pp. 1877–1896, Nov. 2000.

[23] G. Harikumar and Y. Bresler, "Perfect blind restoration of images blurred by multiple filters: Theory and efficient algorithm," *IEEE Trans. Image Process.*, vol. 8, no. 2, pp. 202–219, Feb. 1999.

[24] F. Sroubek and J. Flusser, "Multichannel blind deconvolution of spatially misaligned images," *IEEE Trans. Image Process.*, vol. 14, no. 7, pp. 874–883, Jul. 2005.

[25] V. Katkovnik, D. Paliy, K. Egiazarian, and J. Astola, "Frequency domain blind deconvolution in multiframe imaging using anisotropic spatially-adaptive denoising," in *Proc. 14th Eur. Signal Process. Conf. (EUSIPCO)*, Florence, Italy, Sep. 2006, CD-ROM.

[26] M. Tico and M. Vehvilainen, "Estimation of motion blur point spread function from differently exposed image frames," in *Proc. 14th Eur. Signal Process. Conf. (EUSIPCO)*, Florence, Italy, Sep. 2006, CD-ROM.

[27] R. Molina, A. K. Katsaggelos, J. Abad, and J. Mateos, "A Bayesian approach to blind deconvolution based on Dirichlet distributions," in *Int. Conf. Acoust. Speech Signal Process.*, Apr. 21–24, 1997, vol. 4, pp. 2809–2812.

[28] I. Rekleitis, "Optical flow recognition from the power of spectrum of a single blurred image," in *Proc. Int. Conf. Image Process.*, Sep. 16–19, 1996, vol. 3, pp. 791–794.

[29] F. Rooms, W. Philips, and J. Portilla, "Parametric PSF estimation via sparseness maximization in the wavelet domain," *Proc. SPIE*, vol. 5607, Wavelet Applications in Industrial Processing II, pp. 26–33, 2004.

[30] A. C. Likas and N. P. Galatsanos, "A variational approach for Bayesian blind image deconvolution," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2222–2233, Aug. 2004.

[31] L. Chen and K.-H. Yap, "Efficient discrete spatial techniques for blur support identification in blind image deconvolution," *IEEE Trans. Signal Process.*, vol. 54, no. 4, pp. 1557–1562, Apr. 2006.

[32] L. Chen and K.-H. Yap, "A soft double regularization approach to parametric blind image deconvolution," *IEEE Trans. Image Process.*, vol. 14, no. 5, pp. 624–633, May 2005.

[33] J. Da Rugna and H. Konik, "Blur identification in image processing," in *Proc. IEEE World Congr. Comput. Intell./Int. Joint Conf. Neural Netw.*, Vancouver, BC, Canada, Jul. 2006, pp. 4843–4848.

[34] J. Qiao, J. Liu, and C. Zhao, "A novel SVM-based blind super-resolution algorithm," in *Proc. IEEE World Congr. Comput. Intell./ Int. Joint Conf. Neural Netw.*, Vancouver, BC, Canada, Jul. 2006, pp. 4830–4835.

[35] I. Aizenberg, C. Butakoff, V. Karnaukhov, N. Merzlyakov, and O. Milukova, "Blurred image restoration using the type of blur and blur parameters identification on the neural network," *SPIE Proc.*, vol. 4667, Image Processing: Algorithms and Systems, pp. 460–471, 2002.

[36] I. Aizenberg, D. Paliy, and J. T. Astola, "Multilayer neural network based on multi-valued neurons and the blur identification problem," in *Proc. IEEE World Congr. Comput. Intell./Int. Joint Conf. Neural Newt.*, Vancouver, BC, Canada, Jul. 2006, pp. 1200–1207.

[37] I. Aizenberg, D. Paliy, C. Moraga, and J. Astola, "Blur identification using neural network for image restoration," in *Proc. 9th Int. Conf. Comput. Intell.*, B. Reusch, Ed., Berlin, Germany, 2006, pp. 441–455.

[38] M. M. Gupta, L. Jin, and N. Homma, *Static and Dynamic Neural Networks*. Hoboken, NJ: Wiley, 2003.

[39] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1998.

[40] F. Rosenblat, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. New York: Spartan, 1962.

[41] L. M. Minsky and A. S. Papert, *Perceptrons: An Introduction to Computational Geometry, Expanded Edition*. Cambridge, MA: MIT Press, 1968.

[42] R. Fletcher and C. M. Reeves, "Function minimization by conjugate gradients," *Comput. J.*, vol. 7, pp. 149–154, 1964.

[43] M. F. Moller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Netw.*, vol. 6, pp. 525–533, 1993.

[44] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Cambridge, MA: MIT Press, 1986.

[45] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," 2001 [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm

[46] K.-B. Duan and S. S. Keerthi, "Which is the best multiclass SVM method? An empirical study," in *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, 2005, vol. 3541, Multiple Classifier Systems, pp. 278–285.

Igor Aizenberg (M'01–SM'06) received the M.Sc. degree in mathematics from Uzhgorod National University, Uzhgorod, Ukraine, in 1982 and the Ph.D. degree in computer science from the Dorodnicyn Computing Center of the Russian Academy of Sciences, Moscow, Russia, in 1986, respectively.

In 1982–1990, he was a Research Scientist at the Institute for Information Transmission Problems, Russian Academy of Sciences. In 1990–1996 and 1998–1999, he was an Assistant Professor and then an Associate Professor at the Department of Cybernetics, Uzhgorod National University. In 1996–1998, he was a Research Scientist at the Department of Electrical Engineering, Catholic University of Leuven, Leuven, Belgium. In 1999–2002, he was a Chief Research Scientist at the Neural Networks Technologies, Israel. In 2002–2006, he was a Visiting Research Fellow at the University of Dortmund, Dortmund, Germany, Tampere University of Technology, Tampere, Finland, National Institute of Advanced Industrial Science and Technology, Tsukuba, Japan. In 2003, he was also serving as a Visiting Professor in the summer schools at the University of Zaragoza, Zaragoza, Spain, and at the University of Nis, Nis, Serbia. Since March 2006, he has been at the Department of Computer and Information Sciences, Texas A&M University—Texarkana, Texarkana, where he has a faculty position. His research interests include neural networks, spectral techniques, image processing, and pattern recognition.

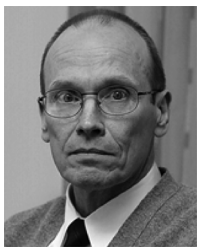
Dmitriy V. Paliy received the B.Sc. and M.Sc. degrees in applied mathematics from Uzhgorod National University, Uzhgorod, Ukraine, in 2000 and 2001, respectively, and the Ph.D. degree from the Tampere University of Technology, Tampere, Finland, in 2007.

Currently, he is a Postdoctoral Researcher at the Department of Information Technology, Tampere University of Technology. His research interests include image restoration (denoising, demosaicing, deblurring), neural networks, and statistical signal processing.

Jacek M. Zurada (M'82–SM'83–F'96) received the M.S. and Ph.D. degrees (with distinction) in electrical engineering from the Technical University of Gdansk, Gdansk, Poland, in 1968 and 1975, respectively.

Since 1989, he has been a Professor, and since 1993 a distinguished Samuel T. Fife Professor at the Electrical and Computer Engineering Department, University of Louisville, Louisville, KY. He was Department Chair from 2004 to 2006. He has published 280 journal and conference papers in the areas of neural networks, computational intelligence, data mining, image processing, and very large scale integration (VLSI) circuits. He also has authored or coauthored three books and coedited a number of volumes in Springer *Lecture Notes on Computer Science*. He has held visiting appointments at Princeton University, Northeastern University, Auburn University, and at foreign universities in Australia, Chile, China, France, Germany, Hong Kong, Italy, Japan, Poland, Singapore, Spain, South Africa, and Taiwan.

Dr. Zurada was an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART I: REGULAR PAPERS and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART II: EXPRESS BRIEFS, and served on the Editorial Board of the PROCEEDINGS OF IEEE. From 1998 to 2003, he was the Editor-in-Chief of the IEEE TRANSACTIONS ON NEURAL NETWORKS. He is an Associate Editor of *Neurocomputing*, *Schedae Informaticae*, *International Journal of Applied Mathematics and Computer Science*, Advisory Editor of *International Journal of Information Technology and Intelligent Computing*, and Editor of Springer Natural Computing Book Series. He has served the profession and the IEEE in various elected capacities, including as President of IEEE Computational Intelligence Society in 2004–2005. He has been member and Chair of various IEEE CIS and IEEE TAB committees and Chair of a number of IEEE symposiums and conferences. He has received a number of awards for distinction in research, teaching, and service including the 1993 Presidential Award for Research, Scholarship and Creative Activity, 1997 Polish Ministry of National Education Award, 1999 IEEE Circuits and Systems Society Golden Jubilee Medal, and the 2001 Presidential Distinguished Service Award for Service to the Profession. He is a Distinguished Speaker of IEEE CIS. In 2003, he was conferred the Title of National Professor by the President of Poland. In 2004 and 2006, respectively, he received Honorary Professorships from Hebei University, and from the University of Electronic Science and Technology of China, Chengdu, China. Since 2005, he has been a Member of the Polish Academy of Sciences. He serves as a Senior Fulbright Specialist (2006–2011).



Jaakko T. Astola (F'00) received the B.Sc., M.Sc., Licentiate, and Ph.D. degrees in mathematics (specializing in error-correcting codes) from Turku University, Turku, Finland, in 1972, 1973, 1975, and 1978, respectively.

From 1976 to 1977, he was with the Research Institute for Mathematical Sciences, Kyoto University, Kyoto, Japan. Between 1979 and 1987, he was with the Department of Information Technology, Lappeenranta University of Technology, Lappeenranta, Finland, holding various teaching positions in mathematics, applied mathematics, and computer science. In 1984, he was a Visiting Scientist at Eindhoven University of Technology, Eindhoven, The Netherlands. From 1987 to 1992, he was an Associate Professor in applied mathematics at Tampere University, Tampere, Finland. Since 1993, he has been Professor of Signal Processing at Tampere University of Technology and is currently Head of Academy of Finland Centre of Excellence in Signal Processing leading a group of about 80 scientists. His research interests include signal processing, coding theory, spectral techniques, and statistics.