A Vector Neural Network for Emitter Identification

Ching-Sung Shieh and Chin-Teng Lin, *Senior Member, IEEE*

***Abstract*—This paper proposes a three-layer vector neural network (VNN) with a supervised learning algorithm suitable for signal classification in general, and for emitter identification (EID) in particular. The VNN can accept interval-value input data as well as scalar input data. The input features of the EID problems include the radio frequency, pulse width, and pulse repetition interval of a received emitter signal. Since the values of these features vary in interval ranges in accordance with a specific radar emitter, the VNN is proposed to process interval-value data in the EID problem. In the training phase, the interval values of the three features are presented to the input nodes of VNN. A new vector-type backpropagation learning algorithm is derived from an error function defined by the VNN's actual output and the desired output indicating the correct emitter type of the corresponding feature intervals. The algorithm can tune the weights of VNN optimally to approximate the nonlinear mapping between a given training set of feature intervals and the corresponding set of desired emitter types. After training, the VNN can be used to identify the sensed scalar-value features from a real-time received emitter signal. A number of simulations are presented to demonstrate the effectiveness and identification capability of VNN, including the two-EID problem and the multi-EID problem with/without additive noise. The simulated results show that the proposed algorithm cannot only accelerate the convergence speed, but it can help avoid getting stuck in bad local minima and achieve higher classification rate.**

***Index Terms*—Convergence, emitter identification (EID), interval value, pulse repetition interval, pulsewidth, radio frequency, supervised learning, vector neural network, vector-type backpropagation.**

I. Introduction

MODERN RADARS have been widely used to detect aircrafts, ships, or land vehicles, or they can be used for searching, tracking, guidance, navigation, and weather forecasting [1]. In military operation, radar is an important piece of equipment which is used to guide weaponry [2]. Hence, an electronic support measure (ESM) system such as radar warning receiver (RWR) is needed to intercept, identify, analyze, and locate the existence of emitter signals. The primary function of the RWR is to warn the crew of an immediate threat with enough information to take evasive action. To accomplish this function, a powerful emitter identification (EID) function must be involved in the RWR system. As the signal pulse density increases, further demands will be put on the EID function. Clearly, the EID function must be sophisticated enough to face the complex surroundings [3].

Manuscript received August 3, 2000; revised June 27, 2001. This work was supported by the MOE Program for Promoting Academic Excellence of Universities under Grant 91-E-FA06-4-4, R.O.C.

The authors are with the Department of Electrical and Control Engineering, National Chiao-Tung University, Hsinchu, Taiwan, R.O.C.

Publisher Item Identifier 10.1109/TAP.2002.801387.

Many conventional signal recognition techniques including k-nearest neighbor classification, and template matching rely on algorithms which are computationally intensive and require a key man to validate and verify the analysis [4]. At present, a histogramming approach is accessed by radio frequency (RF), pulse width (PW), and pulse repetition interval (PRI) of the collected pulse descriptor words (PDWs). This approach is used in a current EID system designed for sorting and comparing tabulated emitter parameters with measured signal parameters. However, these techniques are inefficient and time-consuming for solving EID problems; they often fail to identify signals under high signal density environment, especially, in near real time.

For many practical problems, including pattern matching and classification, function approximation, optimization, vector quantization, data clustering and forecasting, neural networks have drawn much attention and been applied successfully in recent years [4]–[7]. Neural networks have a large number of highly interconnected nodes that usually operate in parallel and are configured in regular architectures. The massive parallelism results in the high computation rate of neural networks and makes the real-time processing of large data feasible.

In this paper, the EID problem is considered as a nonlinear mapping problem. The input features, including RF, PW, and PRI, are extracted from PDWs. Since the values of these features vary in interval ranges in accordance with a specific radar emitter, a vector neural network (VNN) is proposed to process interval-value input data. The VNN can accept either interval-value or scalar-value input and produce scalar output. The proposed VNN is used to construct a functional mapping from the space of the interval-value features to the space of emitter types. The input and output of the VNN are related through interval arithmetics. To train the VNN, a suitable learning algorithm should be developed. The training goal is to find a set of optimal weights in the VNN such that the trained VNN can perform the function described by a training set of if-then rules. Most existing learning methods in neural networks are designed for processing numerical data [4], [8]–[10]. Ishibuchi and his colleagues extended a normal (scalar-type) backpropagation (BP) learning algorithm to the one that can train a feedforward neural network with fuzzy input and fuzzy output [11]. This BP algorithm was derived based on an error function defined by the difference of fuzzy actual output and the corresponding nonfuzzy target output through fuzzy arithmetics. Similar to their approach, we derive a conventional vector-type backpropagation (CVTBP) algorithm for training the proposed VNN. Although, this algorithm can train the VNN for the if-then type training data, it has the problems of slow convergence and bad local minima as a normal scalar-type backpropagation (BP) algorithm does. To obtain better learning results and efficiency for the VNN, we further propose a new vector-type backprop-

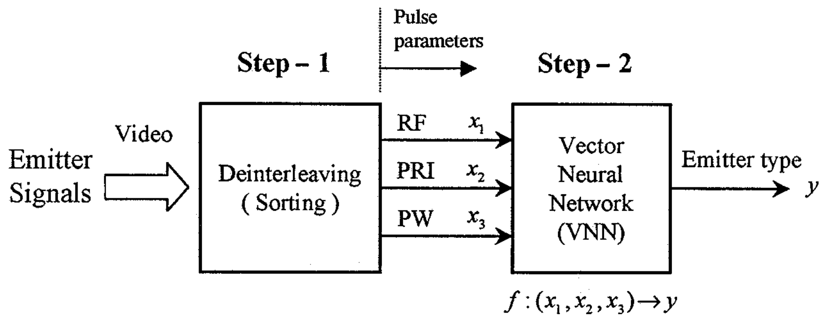

Fig. 1. The flowchart of emitter signal classification.

agation (NVTBP) algorithm derived from a different form of error function. This learning algorithm has a higher convergence rate and is not easily stuck in bad local minima. After training, the VNN can be used to identify the emitter type of the sensed scalar-value features from a real-time received emitter signal. The representation power of the VNN and the effectiveness of the NVTBP learning algorithm are demonstrated on several EID problems, including the two-EID problem and the multi-EID problem with and without additive noise.

The rest of this paper is organized as follows. Section II gives the problem formulation. In Section III, the basic structure of VNN is introduced. Section IV derives an NVTBP algorithm for training the VNN. In Section V, four examples are simulated to demonstrate the identification capability of the proposed VNN with a NVTBP algorithm for emitter signals with and without additive noise. Finally, conclusions are made in Section VI.

II. Problem Formulation

In general, the problem of emitter signal classification is performed in a two-step process as illustrated in Fig. 1. The first step is called deinterleaving (or sorting), which sorts received pulse trains into "bins" according to the specific emitter from the composite set of pulse trains received from passive receivers in the RWR system. After deinterleaving, the second step is to infer the emitter type by each bin of received pulses to differentiate one type from another type. A PDW is generated from the sorting process. Typical measurements include RF, PW, time of arrival (TOA), and (PRI). Thus, a PDW describes a state vector in a multidimensional space. The primary focus of the paper will be on the problem of EID. Emitter parameters and performance are affected by the RF band in which they operate. Likewise, the range of frequency band chosen for a specific emitter is determined by the radar's mission and specifications. The frequency information is very important for both sorting and jamming. By comparing the frequency of the received pulses, the pulse trains can be sorted out and identified for different radars [2]. When the frequency of the victim radar is known, the jammer can concentrate its energy in the desired frequency range. The parameter PW can be used to provide coarse information on the type of radars. For example, generally speaking, weapon radars have short pulses. Another parameter of interest in electronic warfare (EW) receiver measurements is the PRI. The information is the time difference between the leading edge of consecutive transmission waves and is the reciprocal of pulse repetition frequency (PRF). The parameter varies for different radars.

In this paper, the EID problem is considered as a nonlinear mapping problem, the mapping from the space of feature vectors of emitter signals to the space of emitter types. The three parameters, $\mathrm{RF}(x_1)$, $\mathrm{PRI}(x_2)$, and $\mathrm{PW}(x_3)$, are used to form the feature vector $[x_1, x_2, x_3]$ in this problem. Such a nonlinear mapping function can be approximated by a suitable neural network [4], [5]. However, these parameters operate in interval ranges for a specific radar emitter; for example, RF ranges from 15.6 to 16.6 GHz, PRI ranges from 809 to 960 μs, and PW ranges from 1.8 to 3.6 μs for some specific emitter type. To endow a neural network with the interval-value processing ability, we propose a VNN that can accept either interval-value or scalar-value input and produce scalar output. In the training phase, the VNN is trained to form a functional mapping from the space of interval-value features to the space of emitter types based on N_t samples of training pairs $(\tilde{\mathbf{x}}_p; \mathbf{d}_p)$ for the EID problem, where $p = 1, \ldots, N_t$ indicating the pth training pair, $\tilde{\mathbf{x}}_p = [\tilde{x}_{p1}, \tilde{x}_{p2}, \tilde{x}_{p3}]$. In each training pair, $\tilde{x}_{pi}$ is an interval value represented by $[x_{pi}^L, x_{pi}^U]$, and $\mathbf{d}_p$ is an m-dimensional $\{0, 1\}$ vector containing only one 1 to indicate the emitter type among m candidates. Hence, the VNN has three-input nodes with each node receiving one feature's value, and m output nodes with each node representing one emitter type. The input–output relationship of the VNN is denoted by

$$\mathbf{y}_p = \hat{f}(\tilde{\mathbf{x}}_p) \tag{1}$$

where $\mathbf{y}_p$ is a m-dimensional vector indicating the actual output of the VNN, and $\hat{f}$ represents the approximated function formed by the VNN. More clearly, the VNN is trained to represent the EID mapping problem in the following if-then form:

$$\text{IF } x_{p1} \text{ is in } \left[x_{p1}^L, x_{p1}^U\right] \text{ and } \cdots \text{ and } x_{pn} \text{ is in } \left[x_{pn}^L, x_{pn}^U\right] \quad \text{THEN } \mathbf{x}_p = [x_{p1}, \ldots, x_{pn}] \text{ belongs to } C_k \tag{2}$$

where C_k denotes the kth emitter type.

The objective of learning is to obtain an approximated model $\hat{f}(\cdot)$ for the mapping in (1) and (2) such that the error function indicating the difference between $\mathbf{d}_p$ and $\mathbf{y}_p$, $p = 1, 2, \ldots, N_t$, is minimized. Two different error functions are used in this paper, one is the common root-mean-square error function, and the other is

$$E(w) = -\sum_{p=1}^{N_t} \left\{\mathbf{d}_p \ln \mathbf{y}_p + (1 - \mathbf{d}_p) \ln(1 - \mathbf{y}_p)\right\}. \tag{3}$$

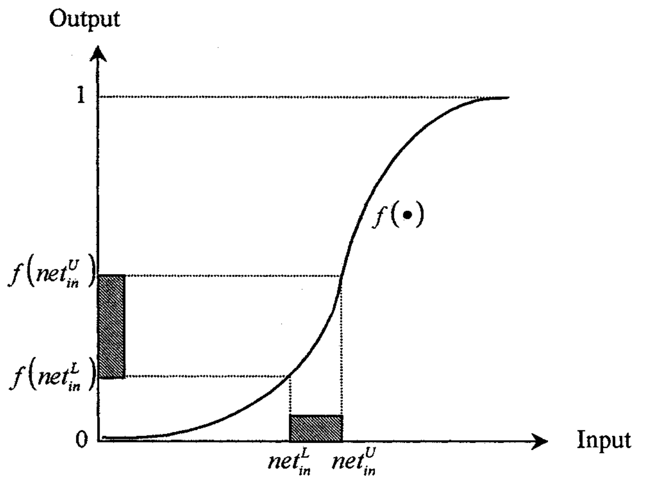

Fig. 2. Interval sigmoid function of each node in the VNN, where $(\mathrm{net}_{\mathrm{in}}^L, \mathrm{net}_{\mathrm{in}}^U)$ and $f\,(\mathrm{net}_{\mathrm{in}}^L, \mathrm{net}_{\mathrm{in}}^U)$ are an interval input and an interval output, respectively.

After training, the trained VNN can be used in the functional phase or the so-called testing phase. In this phase, the VNN on-line accepts a feature vector, $\mathbf{x} = [x_1, x_2, x_3]$ containing scalar values x_i from the sensors, and produces an output vector $\mathbf{y}_p$ with the highest value element in $\mathbf{y}_p$ indicating the identified emitter type.

III. Structure of VNN

In this section, we shall introduce the structure and function of the VNN which can process interval-value as well as scalar-value data. Before doing so, let us review some operations of *interval arithmetic* that will be used later. Let $A = [a^L, a^U]$ and $B = [b^L, b^U]$ be intervals, where the superscripts L and U represent the lower limit and upper limit, respectively. Then, we have

$$A + B = [a^L, a^U] + [b^L, b^U] = [a^L + b^L, a^U + b^U] \tag{4}$$

and

$$\begin{aligned} k \cdot A &= k \cdot [a^L, a^U] = [ka^L, ka^U] \\ &= \begin{cases} [ka^L, ka^U], & \text{if } k \ge 0 \\ [ka^U, ka^L], & \text{if } k < 0 \end{cases} \end{aligned} \tag{5}$$

where k is a real number. The activation function of a neuron can also be extended to an interval input–output relation as

$$f(\mathrm{Net_{in}}) = f\left([\mathrm{net}_{\mathrm{in}}^L, \mathrm{net}_{\mathrm{in}}^U]\right) = [f(\mathrm{net}_{\mathrm{in}}^L), f(\mathrm{net}_{\mathrm{in}}^U)] \tag{6}$$

where $\mathrm{Net_{in}} = [\mathrm{net}_{\mathrm{in}}^L, \mathrm{net}_{\mathrm{in}}^U]$ is interval-valued and $f(\cdot)$ is a sigmoid function. The sigmoid function is denoted by $f(\mathrm{Net_{in}}) = 1/(1 + \exp(-\mathrm{Net_{in}}))$. The interval activation function defined by (6) is illustrated in Fig. 2.

We shall now describe the function of the VNN using the above *interval arithmetic* operations. The general structure of VNN is shown in Fig. 3, where the solid lines show the forward propagation of signals, and the dashed lines show the backward propagation of errors. In order to identify any n-dimensional interval-value vector, we employ a VNN that has n input nodes, l hidden nodes, and m output nodes. When the interval-value input vector $\tilde{\mathbf{x}}_p = (\tilde{x}_{p1}, \ldots, \tilde{x}_{pn})$ is presented to the input layer of VNN, the input–output relation of each node of VNN is explicitly calculated as follows, where $\tilde{x}_{pi} = [x_{pi}^L, x_{pi}^U]$.

Input nodes: Each input node just passes the external input, $\tilde{x}_{pi} = [x_{pi}^L, x_{pi}^U]$, $i = 1, \ldots, n$, forward to the hidden nodes.

Hidden nodes:

$$\begin{aligned} \tilde{z}_{pj} &= [z_{pj}^L, z_{pj}^U] = [f(\mathrm{net}_{pj}^L), f(\mathrm{net}_{pj}^U)] \\ j &= 1, \ldots, l \end{aligned} \tag{7}$$

$$\mathrm{net}_{pj}^L = \sum_{\substack{i=1 \\ w_{ji}^{(1)} \ge 0}}^{n} w_{ji}^{(1)} x_{pi}^L + \sum_{\substack{i=1 \\ w_{ji}^{(1)} < 0}}^{n} w_{ji}^{(1)} x_{pi}^U + \theta_j \tag{8}$$

$$\mathrm{net}_{pj}^U = \sum_{\substack{i=1 \\ w_{ji}^{(1)} \ge 0}}^{n} w_{ji}^{(1)} x_{pi}^U + \sum_{\substack{i=1 \\ w_{ji}^{(1)} < 0}}^{n} w_{ji}^{(1)} x_{pi}^L + \theta_j. \tag{9}$$

Output nodes:

$$\begin{aligned} \tilde{y}_{pk} &= [y_{pk}^L, y_{pk}^U] = [f(\mathrm{net}_{pk}^L), f(\mathrm{net}_{pk}^U)] \\ k &= 1, \ldots, m \end{aligned} \tag{10}$$

$$\mathrm{net}_{pk}^L = \sum_{\substack{j=1 \\ w_{kj}^{(2)} \ge 0}}^{l} w_{kj}^{(2)} z_{pj}^L + \sum_{\substack{j=1 \\ w_{kj}^{(2)} < 0}}^{l} w_{kj}^{(2)} z_{pj}^U + \theta_k \tag{11}$$

$$\mathrm{net}_{pk}^U = \sum_{\substack{j=1 \\ w_{kj}^{(2)} \ge 0}}^{l} w_{kj}^{(2)} z_{pj}^U + \sum_{\substack{j=1 \\ w_{kj}^{(2)} < 0}}^{l} w_{kj}^{(2)} z_{pj}^L + \theta_k \tag{12}$$

where the weights $w_{ji}^{(1)}$, $w_{kj}^{(2)}$ and the biases θ_j, θ_k are real parameters and the outputs $\tilde{z}_{pj}$, and $\tilde{y}_{pk}$ are intervals. It is noted that the VNN can also process scalar-value input data by setting $x_{pi}^L = x_{pi}^U = x_{pi}$, where x_{pi} is the scalar-value input. Correspondingly, the VNN can produce scalar output, $y_{pk}^L = y_{pk}^U = y_{pk}$.

IV. Supervised Learning Algorithms for VNN

In this section, we shall derive a NVTBP learning algorithm for the proposed VNN with interval-value input data. A normal cost function in the CVTBP algorithm, E_{pk}, is first considered using the interval output $\tilde{y}_{pk} = [y_{pk}^L, y_{pk}^U]$ and the corresponding desired output d_{pk} for the pth input pattern, as

$$E_{pk} = \begin{cases} \frac{(d_{pk} - y_{pk}^L)^2}{2}, & \text{if } d_{pk} = 1 \\ \frac{(d_{pk} - y_{pk}^U)^2}{2}, & \text{if } d_{pk} = 0 \end{cases} \tag{13}$$

for the case of an interval-value input vector and a crisp desired output. However, to enhance the identification power of VNN, we propose a NVTBP algorithm, which the error cost function instead of the squares of the differences between the actual interval output $\tilde{y}_{pk}$ and the corresponding desired output d_{pk} as in (13), where the subscript pk represents the pth-input pattern and kth-output node. The new error cost function is defined as

$$E_{pk} = \begin{cases} -d_{pk} \ln y_{pk}^L - (1 - d_{pk}) \ln(1 - y_{pk}^L), & \text{if } d_{pk} = 1 \\ -d_{pk} \ln y_{pk}^U - (1 - d_{pk}) \ln(1 - y_{pk}^U), & \text{if } d_{pk} = 0. \end{cases} \tag{14}$$

The learning objective is to minimize the error function in (13) and (14). The weight updating rules for the VNN are illustrated

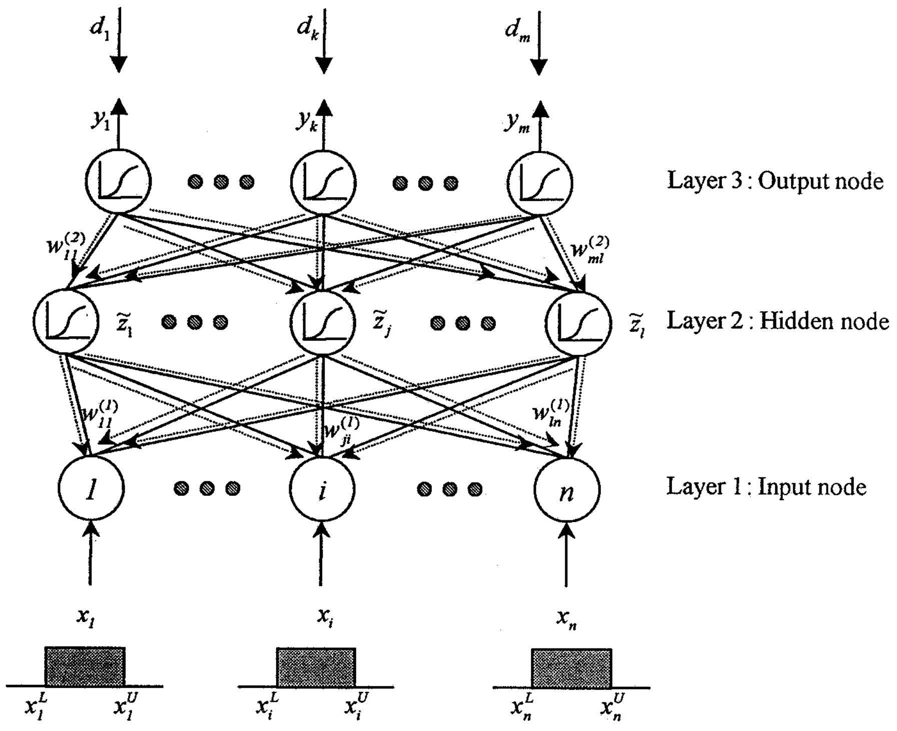

Fig. 3. The three-layer architecture of the proposed VNN.

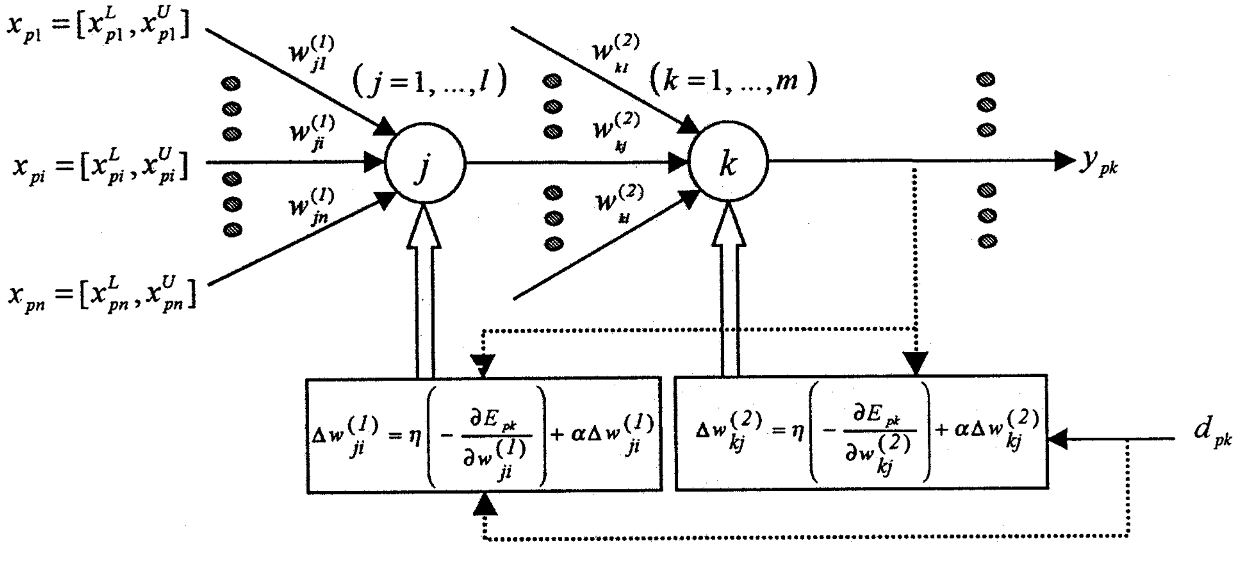
$x_{p1} = [x_{p1}^L, x_{p1}^U]$
$x_{pi} = [x_{pi}^L, x_{pi}^U]$
$x_{pn} = [x_{pn}^L, x_{pn}^U]$
$w_{j1}^{(1)}$
$w_{ji}^{(1)}$
$w_{jn}^{(1)}$
$(j = 1, \ldots, l)$
$w_{k1}^{(2)}$
$w_{kj}^{(2)}$
$w_{kl}^{(2)}$
$(k = 1, \ldots, m)$
j
k
y_{pk}
$\Delta w_{ji}^{(1)} = \eta\left(-\frac{\partial E_{pk}}{\partial w_{ji}^{(1)}}\right) + \alpha \Delta w_{ji}^{(1)}$
$\Delta w_{kj}^{(2)} = \eta\left(-\frac{\partial E_{pk}}{\partial w_{kj}^{(2)}}\right) + \alpha \Delta w_{kj}^{(2)}$
d_{pk}

Fig. 4. Illustration of backpropagation learning rule for the VNN.

in Fig. 4. To show the learning rules, we shall calculate the computation of $\partial E_{pk}/\partial w$ layer by layer along the dashed lines in Fig. 3, and start the derivation from the output nodes.

Layer 3: Using (10)–(12) to calculate $\partial E_{pk}/\partial w_{kj}^{(2)}$ for various values of the weights and desired output. The results are summarized in (15), as shown at the bottom of the page.

$$\frac{\partial E_{pk}}{\partial w_{kj}^{(2)}} = \begin{cases} -(d_{pk} - y_{pk}^L) z_{pj}^L \triangleq -\delta_{pk}^{\prime L} z_{pj}^L, & \text{if } d_{pk} = 1 \text{ and } w_{kj}^{(2)} \geq 0 \\ -(d_{pk} - y_{pk}^L) z_{pj}^U \triangleq -\delta_{pk}^{\prime L} z_{pj}^U, & \text{if } d_{pk} = 1 \text{ and } w_{kj}^{(2)} < 0 \\ -(d_{pk} - y_{pk}^U) z_{pj}^U \triangleq -\delta_{pk}^{\prime U} z_{pj}^U, & \text{if } d_{pk} = 0 \text{ and } w_{kj}^{(2)} \geq 0 \\ -(d_{pk} - y_{pk}^U) z_{pj}^L \triangleq -\delta_{pk}^{\prime U} z_{pj}^L, & \text{if } d_{pk} = 0 \text{ and } w_{kj}^{(2)} < 0 \end{cases}. \tag{15}$$

Layer 2: Using (7)–(9) to calculate $\partial E_{pk}/\partial w_{ji}^{(1)}$ for different values of the weights and desired output. The results are summarized in (16), as shown at the bottom of the page. In the previous discussion, the notations $\delta_{pk}^{\prime L}$ and $\delta_{pk}^{\prime U}$ are defined as follows:

$$\delta_{pk}^{\prime L} \triangleq \frac{\partial E_{pk}}{\partial \mathrm{net}_{pk}^{L}} = \frac{\partial E_{pk}}{\partial y_{pk}^{L}} \frac{\partial y_{pk}^{L}}{\partial \mathrm{net}_{pk}^{L}} = (d_{pk} - y_{pk}^{L}) \tag{17}$$

$$\delta_{pk}^{\prime U} \triangleq \frac{\partial E_{pk}}{\partial \mathrm{net}_{pk}^{U}} = \frac{\partial E_{pk}}{\partial y_{pk}^{U}} \frac{\partial y_{pk}^{U}}{\partial \mathrm{net}_{pk}^{U}} = (d_{pk} - y_{pk}^{U}). \tag{18}$$

Clearly, the value of δ_{pk}' is proportional to the amount of $(d_{pk} - y_{pk})$ rather than $(d_{pk} - y_{pk})y_{pk}(1 - y_{pk})$ as in the CVTBP learning rule. When the actual output y_{pk} (representing y_{pk}^{L} or y_{pk}^{U}) approaches the value of 1 or 0, the factor $y_{pk}(1 - y_{pk})$ makes the error signal δ_{pk} very small. This implies that an output node can be maximally wrong without producing a strong error signal with which the connection weights could be significantly adjusted. This decelerates the search for a minimum in the error. A detailed description of quantitative analysis can be found in [12].

In summary, the supervised learning algorithm for the VNN is outlined in the following:

```
NVTBP algorithm:
Consider a 3-layer VNN with $n$ input nodes, $l$ hidden nodes, and $m$ output
nodes. The connection weight $w_{ji}^{(1)}$ is from node $i$ of the input layer
to the $j$th node of the hidden layer, and $w_{kj}^{(2)}$ is from the $j$th node
of the hidden layer to the $k$th node of the output layer.
Input: A set of training pairs $\{(\tilde{\mathbf{x}}_p; \mathbf{d}_p), p = 1, \ldots, N_t\}$, where the input
vectors are in interval values.
  1) (Initialization): Choose $\eta > 0$ and $E_{\max}$ (maximum tolerable error).
     Initialize the weights to small random values. Set $E = 0$ and $p = 1$.
  2) (Training loop): Apply the $p$th input pattern $\tilde{\mathbf{x}}_p$ to the input layer.
  3) (Forward propagation): Propagate the signal forward through the network
     from the input layer to the output layer. Use (7)-(9) to compute the net
     input $(\mathrm{net}_{pj})$ and output $(\tilde{z}_{pj})$ of the $j$th hidden node, and use (10)-(12)
     to compute the net input $(\mathrm{net}_{pk})$ and output $(\tilde{y}_{pk})$ of the $k$th output node.
  4) (Output error measure): Compute the error signal $\delta_{pk}'$ from (17) and (18).
  5) (Error backpropagation): Propagate the errors backward to update the
     weight changes $\Delta w_{kj}^{(2)}$ between hidden and output nodes, and update the
     weight changes $\Delta w_{ji}^{(1)}$ between input and hidden nodes.
  6) (One epoch looping): Check whether the whole set of training data has
     been cycled once. If $p < N_t$, then $p = p + 1$ and go to Step 1; otherwise,
     go to Step 7.
  7) (Total error checking): Check whether the current total error is acceptable;
     if $E < E_{\max}$, then terminate the training process and output the final
     weights; otherwise, set $E = 0$, $p = 1$, and initiate the new training epoch
     by going to Step 2.
END New vector-type algorithm
```

The above algorithm adopts the *incremental* approach in updating the weights; that is, the weights are updated for each incoming training pattern. Finally, the optimal weights $\mathbf{w}_{\mathrm{opt}}^{(1)}$ and $\mathbf{w}_{\mathrm{opt}}^{(2)}$ can be obtained through the training procedure and expressed by

$$\mathbf{w}_{\mathrm{opt}}^{(1)} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} & \cdots & w_{1n}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} & \cdots & w_{2n}^{(1)} \\ \vdots & \vdots & & \vdots \\ w_{l1}^{(1)} & w_{l2}^{(1)} & \cdots & w_{ln}^{(1)} \end{bmatrix} \tag{19}$$

and

$$\mathbf{w}_{\mathrm{opt}}^{(2)} = \begin{bmatrix} w_{11}^{(2)} & w_{12}^{(2)} & \cdots & w_{1l}^{(2)} \\ w_{21}^{(2)} & w_{22} & \cdots & w_{2l}^{(2)} \\ \vdots & \vdots & & \vdots \\ w_{m1}^{(2)} & w_{m2}^{(2)} & \cdots & w_{ml}^{(2)} \end{bmatrix}. \tag{20}$$

V. Simulation Results

In this section, we employ the proposed VNN trained by the NVTBP algorithm to handle the practical EID problems in

$$\frac{\partial E_{pk}}{\partial w_{ji}^{(1)}} = \begin{cases} -\delta_{pk}^{\prime L} w_{kj}^{(2)} z_{pj}^{L}(1 - z_{pj}^{L}) x_{pi}^{L}, & \text{if } d_{pk} = 1, w_{kj}^{(2)} \geq 0 \text{ and } w_{ji}^{(1)} \geq 0 \\ -\delta_{pk}^{\prime L} w_{kj}^{(2)} z_{pj}^{L}(1 - z_{pj}^{L}) x_{pi}^{U}, & \text{if } d_{pk} = 1, w_{kj}^{(2)} \geq 0 \text{ and } w_{ji}^{(1)} < 0 \\ -\delta_{pk}^{\prime L} w_{kj}^{(2)} z_{pj}^{U}(1 - z_{pj}^{U}) x_{pi}^{U}, & \text{if } d_{pk} = 1, w_{kj}^{(2)} < 0 \text{ and } w_{ji}^{(1)} \geq 0 \\ -\delta_{pk}^{\prime L} w_{kj}^{(2)} z_{pj}^{U}(1 - z_{pj}^{U}) x_{pi}^{L}, & \text{if } d_{pk} = 1, w_{kj}^{(2)} < 0 \text{ and } w_{ji}^{(1)} < 0 \\ -\delta_{pk}^{\prime U} w_{kj}^{(2)} z_{pj}^{U}(1 - z_{pj}^{U}) x_{pi}^{U}, & \text{if } d_{pk} = 0, w_{kj}^{(2)} \geq 0 \text{ and } w_{ji}^{(1)} \geq 0 \\ -\delta_{pk}^{\prime U} w_{kj}^{(2)} z_{pj}^{U}(1 - z_{pj}^{U}) x_{pi}^{L}, & \text{if } d_{pk} = 0, w_{kj}^{(2)} \geq 0 \text{ and } w_{ji}^{(1)} < 0 \\ -\delta_{pk}^{\prime U} w_{kj}^{(2)} z_{pj}^{L}(1 - z_{pj}^{L}) x_{pi}^{L}, & \text{if } d_{pk} = 0, w_{kj}^{(2)} < 0 \text{ and } w_{ji}^{(1)} \geq 0 \\ -\delta_{pk}^{\prime U} w_{kj}^{(2)} z_{pj}^{L}(1 - z_{pj}^{L}) x_{pi}^{U}, & \text{if } d_{pk} = 0, w_{kj}^{(2)} < 0 \text{ and } w_{ji}^{(1)} < 0 \end{cases}. \tag{16}$$

TABLE I
INPUT–OUTPUT TRAINING PAIRS FOR THE TWO-EMITTER IDENTIFICATION PROBLEM IN EXPERIMENTS 1 AND 3

Pattern Number	RF (GHz)		PRI (μ s)		PW (μ s)		Emitter Type
	Lower limit	Upper limit	Lower limit	Upper limit	Lower limit	Upper limit	
1	15.6	16.6	8.09	9.60	1.80	3.60	1
2	4.2	5.0	2.20	4.50	2.45	4.75	2
3	4.3	4.9	2.30	4.40	2.50	4.70	2
4	4.4	4.8	2.40	4.30	2.55	4.65	2
5	15.70	16.50	8.19	9.50	1.90	3.50	1
6	15.80	16.40	8.29	9.40	2.00	3.40	1
7	4.50	4.70	2.50	4.20	2.60	4.60	2
8	15.90	16.30	8.39	9.30	2.10	3.30	1
9	16.00	16.20	8.49	9.20	2.20	3.20	1
10	4.55	4.60	2.60	4.10	2.65	4.55	2

TABLE II
PART OF THE TESTING SAMPLES FOR THE TWO-EMITTER IDENTIFICATION PROBLEM IN EXPERIMENTS 1 AND 3

Pattern Number	RF (GHz)		PRI (μ s)		PW (μ s)		Emitter Type
	Lower limit	Upper limit	Lower limit	Upper limit	Lower limit	Upper limit	
1	15.65	15.65	8.09	8.09	1.80	1.80	1
2	15.70	15.70	8.12	8.12	1.82	1.82	1
3	15.75	15.75	8.15	8.15	1.85	1.85	1
4	15.80	15.80	8.19	8.19	1.88	1.88	1
5	15.85	15.85	8.23	8.23	1.91	1.91	1
6	15.90	15.90	8.25	8.25	1.93	1.93	1
7	15.95	15.95	8.28	8.28	1.95	1.95	1
8	16.05	16.05	8.33	8.33	1.99	1.99	1
9	16.10	16.10	8.39	8.39	2.03	2.03	1
10	16.13	16.13	8.43	8.43	2.05	2.05	1
11	4.20	4.20	2.19	2.19	2.45	2.45	2
12	4.22	4.22	2.24	2.24	2.51	2.51	2
13	4.23	4.23	2.43	2.43	2.58	2.58	2
14	4.25	4.25	2.57	2.57	2.62	2.62	2
15	4.28	4.28	2.68	2.68	2.74	2.74	2
16	4.31	4.31	2.73	2.73	2.81	2.81	2
17	4.33	4.33	2.82	2.82	2.85	2.85	2
18	4.35	4.35	2.91	2.91	2.98	2.98	2
19	4.38	4.38	3.03	3.03	3.05	3.05	2
20	4.43	4.43	3.13	3.13	3.15	3.15	2

real-life; i.e., to map input patterns to their respective emitter types. An input pattern is determined to belong to the kth type, if the kth output node produces a higher value than all the other output nodes when this input pattern is presented to the VNN. All reference data of the simulated emitters are given by references [3], [5], and [13]. Before the input patterns are presented to the VNN, the range of each parameter must be normalized over the following bound to increase the network's learning ability:

$$\text{RF:}2.0\ \text{GHz to } 18.0\ \text{GHz}$$
$$\text{PRI:}1.0\ \mu\text{s to } 10.0\ \mu\text{s}$$
$$\text{PW:}0.1\ \mu\text{s to } 10.0\ \mu\text{s}.$$

Two problems are examined to demonstrate the identification capability of the proposed VNN in this section, the two-EID problem and three-EID problem. The performance is compared to that of the VNN trained by CVTBP algorithm on the same training and testing data.

A. Performance Evaluation Without Measurement Error

In this section, two experiments are performed for clean input data without measurement distortion to demonstrate the identification capability of the VNN trained by the NVTBP algorithm and the CVTBP algorithm, respectively.

TABLE III
INPUT–OUTPUT TRAINING PAIRS FOR THE TWO-EMITTER IDENTIFICATION PROBLEM IN EXPERIMENTS 2 AND 4

Pattern Number	RF (GHz)		PRI (μ s)		PW (μ s)		Emitter Type
	Lower limit	Upper limit	Lower limit	Upper limit	Lower limit	Upper limit	
1	15.6	16.6	8.09	9.60	1.80	3.60	1
2	4.20	5.00	2.20	4.50	2.45	4.75	2
3	4.30	4.90	2.30	4.40	2.50	4.70	2
4	16.35	17.45	5.05	6.90	5.35	7.85	3
5	15.70	16.50	8.19	9.50	1.90	3.50	1
6	15.80	16.40	8.29	9.40	2.00	3.40	1
7	16.45	17.35	5.15	6.80	5.45	7.75	3
8	16.55	17.25	5.25	6.70	5.55	7.65	3
9	4.50	4.70	2.50	4.20	2.60	4.60	2
10	15.90	16.30	8.39	9.30	2.10	3.30	1
11	16.00	16.20	8.49	9.20	2.20	3.20	1
12	16.75	17.15	5.35	6.60	5.65	7.55	3
13	4.40	4.80	2.40	4.30	2.55	4.65	2
14	16.85	17.05	5.45	6.5	5.75	7.45	3
15	4.55	4.60	2.60	4.10	2.65	4.55	2

TABLE IV
PART OF THE TESTING SAMPLES FOR THE TWO-EMITTER IDENTIFICATION PROBLEM IN EXPERIMENTS 2 AND 4

Pattern Number	RF (GHz)		PRI (μ s)		PW (μ s)		Emitter Type
	Lower limit	Upper limit	Lower limit	Upper limit	Lower limit	Upper limit	
1	15.65	15.65	8.09	8.09	1.80	1.80	1
2	15.70	15.70	8.12	8.12	1.82	1.82	1
3	15.75	15.75	8.15	8.15	1.85	1.85	1
4	15.80	15.80	8.19	8.19	1.88	1.88	1
5	15.85	15.85	8.23	8.23	1.91	1.91	1
6	15.90	15.90	8.25	8.25	1.93	1.93	1
7	15.95	15.95	8.28	8.28	1.95	1.95	1
8	16.05	16.05	8.33	8.33	1.99	1.99	1
9	16.10	16.10	8.39	8.39	2.03	2.03	1
10	16.13	16.13	8.43	8.43	2.05	2.05	1
11	4.20	4.20	2.19	2.19	2.45	2.45	2
12	4.22	4.22	2.24	2.24	2.51	2.51	2
13	4.23	4.23	2.43	2.43	2.58	2.58	2
14	4.25	4.25	2.57	2.57	2.62	2.62	2
15	4.28	4.28	2.68	2.68	2.74	2.74	2
16	4.31	4.31	2.73	2.73	2.81	2.81	2
17	4.33	4.33	2.82	2.82	2.85	2.85	2
18	4.35	4.35	2.91	2.91	2.98	2.98	2
19	4.38	4.38	3.03	3.03	3.05	3.05	2
20	16.35	16.35	5.05	5.05	5.35	5.35	3
21	16.41	16.41	5.13	5.13	5.42	5.42	3
22	16.45	16.45	5.17	5.17	5.55	5.55	3
23	16.51	16.51	5.25	5.25	5.88	5.88	3
24	16.55	16.55	5.28	5.28	5.91	5.91	3
25	16.61	16.61	5.35	5.35	5.93	5.93	3
26	16.65	16.65	5.39	5.39	5.95	5.95	3
27	16.69	16.69	5.43	5.43	5.99	5.99	3
28	16.72	16.72	5.59	5.59	6.03	6.03	3
29	16.83	16.83	5.63	5.63	6.05	6.05	3
30	4.43	4.43	3.13	3.13	3.15	3.15	2

Experiment 1: For the two-EID problem, we employ a VNN with three-input nodes, five-hidden nodes and two-output nodes (denoted by 3-5-2 network). We set the learning rate as $\eta = 0.01$ and momentum constant as $\alpha = 0.99$ in the NVTBP learning algorithm, and $\eta = 0.05$, $\alpha = 0.9$ in the CVTBP learning algorithm. The 10 input–output training pairs (five pairs for each type) are listed in Table I. In the training phase, we use these

TABLE V
(A) TESTING RESULTS OF THE 3-5-2 VNN ON THE TWO-EMITTER IDENTIFICATION PROBLEM WITH/WITHOUT NOISE. (B) TESTING RESULTS OF THE 3-8-3 VNN ON THE THREE-EMITTER IDENTIFICATION PROBLEM WITH/WITHOUT NOISE

Error Deviation Level (%)	3-5-2 VNN trained by the NVTBP algorithm			3-5-2 VNN trained by the CVTBP algorithm		
	Average Correction Rate (%)		Total Average Correction Rate (%)	Average Correction Rate (%)		Total Average Correction Rate (%)
	Type 1	Type 2		Type 1	Type 2	
15	99.54	99.87	99.71	88.08	94.01	91.04
13	99.92	99.88	99.90	93.08	94.42	93.75
11	99.94	99.88	99.91	95.07	94.63	94.85
9	99.94	99.88	99.91	96.19	94.80	95.49
7	99.94	99.88	99.91	96.74	94.93	95.83
5	99.94	99.88	99.91	97.03	95.03	96.03
3	99.94	99.88	99.91	97.19	95.11	96.15
1	99.94	99.88	99.91	97.29	95.17	96.23
0	99.94	99.88	99.91	97.31	95.21	96.26

(a)

Error Deviation Level (%)	3-8-3 VNN trained by the NVTBP algorithm				3-8-3 VNN trained by the CVTBP algorithm			
	Average Correction Rate (%)			Total Average Correction Rate (%)	Average Correction Rate (%)			Total Average Correction Rate (%)
	Type 1	Type 2	Type 3		Type 1	Type 2	Type 3	
15	63.00	89.39	74.87	75.75	57.89	87.97	70.78	72.21
13	72.20	90.36	74.93	79.16	59.06	89.36	70.88	73.10
11	74.25	92.26	74.95	80.49	60.05	90.27	70.96	73.76
9	79.15	93.78	79.35	84.09	60.85	92.15	75.50	76.17
7	86.35	96.18	85.80	89.44	67.11	93.12	80.52	80.25
5	96.01	97.94	94.16	96.04	75.16	94.04	88.69	85.96
3	99.34	99.30	99.69	99.44	80.56	94.82	92.18	89.19
1	99.60	99.87	99.93	99.80	82.71	95.48	93.70	90.63
0	99.63	99.95	99.94	99.84	83.36	95.80	94.07	91.08

(b)

training pairs to train two VNNs using the CVTBP and NVTBP algorithms, respectively, and find individually a set of optimal weights. In the testing phase, 80 testing patterns (40 patterns for each emitter type) are randomly selected from the ranges of emitter parameters and are presented to the trained VNNs for performance testing. Part of these testing patterns are shown in Table II. Once a testing pattern is fed to the trained VNNs, the networks identify immediately its emitter type in near real time. The testing results show that the two trained VNNs achieve high identification rates. However, the VNN trained by the NVTBP algorithm performs better than the VNN trained by the CVTBP algorithm; the former achieves an average identification rate of 99.91% and the latter 96.26% as listed in the last row of Table V(a).

Experiment 2: In this experiment, a three-EID problem is solved by two 3-8-3 VNNs trained by the NVTBP and CVTBP algorithms, respectively. We first set the learning constant as $\eta = 0.02$ and momentum constant as $\alpha = 0.7$ in both learning algorithms. The 15 input–output training pairs (five pairs for each type) as listed in Table III are used to train the two VNNs. After training, 120 testing patterns (40 patterns for each emitter type) are presented to the trained VNNs for performance testing. Part of these testing patterns are shown in Table IV. Again, both networks show high identification capability, where the VNN trained by the NVTBP algorithm achieves an average identification rate of 99.84% and the other VNN 91.08% as listed in the last row of Table V(b).

In the above two experiments, the results show that the two-emitter and three-EID problems can be easily handled by the VNN with the derived NVTBP algorithm and the CVTBP algorithm in real time. However, the VNN trained by the NVTBP algorithm has better identification capability than that trained by the CVTBP algorithm. These experiments are performed for clean input data without measurement distortion. The robustness of the proposed scheme in noisy environment is tested in the following experiments.

B. Performance Evaluation With Measurement Error

In this subsection, two experiments are performed to evaluate the robustness of VNN with measurement distortion. In these experiments, the measurement distortion is simulated by adding noise. To perform the testing at different levels of adding noise, we define the error deviation level (EDL) by

$$\mathrm{EDL}_i(\%) = \frac{\xi_{pi}}{x_{pi}} \times 100\%, \qquad i = 1, 2, 3 \tag{21}$$

for emitter type i, where ξ_{pi} is a small randomly generated deviation for the pth input pattern.

Experiment 3: First, we consider the two-EID problem with the input data corrupted by additive noise. The noise testing patterns are obtained by adding random noise, $\xi_{pi}(i = 1, 2, 3)$, to each testing pattern (x_{p1}, x_{p2}, x_{p3}), $p = 1, \ldots, 80$, used in Experiment 1 to form the noise testing database, $(x_{p1} \pm \xi_{p1}, x_{p2} \pm \xi_{p2}, x_{p3} \pm \xi_{p3})$. The noisy testing patterns with different EDL's (from 1% to 15%) are presented to the trained 3-5-2 VNNs in Experiment 1 for performance testing. The testing results are shown in Table V(a).

Experiment 4: In this experiment, we consider the three-EID problem with the input data corrupted by additive noise. The noise testing patterns are obtained by adding random noise, $\xi_{pi}(i=1,2,3)$, to each testing pattern (x_{p1}, x_{p2}, x_{p3}), $p = 1, \ldots, 120$, used in Experiment 2 to form the noise testing database, $(x_{p1} \pm \xi_{p1}, x_{p2} \pm \xi_{p2}, x_{p3} \pm \xi_{p3})$. The noisy testing patterns with different EDLs are presented to the trained 3-8-3 VNNs in Experiment 2 for performance testing. The testing results are shown in Table V(b).

The testing results in Table V indicate that, as expected, the VNNs identification ability decreases as EDL increasing in noisy environments. In conclusion, the proposed VNN trained by the NVTBP algorithm not only has higher identification capability, but is also relatively more insensitive to noise than that trained by the CVTBP algorithm.

VI. Conclusion

In this paper, a VNN along with a new vector-type backpropagation (NVTBP) learning algorithm was proposed to solve the EID problem. The VNN can learn the teaching patterns in the form of interval-value input and scalar-value output in the training phase, and then operate in the way of scalar-value input and scalar-value output in the testing phase by means of interval arithmetics. The main contribution of this paper is to propose an idea for integrating the processing of interval-value and scalar-value data into a single processing system and derive a NVTBP learning algorithm for solving the practical EID problem in real time. In fact, the proposed network with the NVTBP learning algorithm can not only solve the learning problem with interval-value data, but also improve the convergence of the CVTBP algorithm. The simulated results show that the proposed VNN can always produce high identification accuracy for the emitter signals. Also, it was demonstrated that the VNN is quite insensitive to the additive error. With these results achieved in this paper, the proposed VNN may be widely applied to military applications (such as reconnaissance and threat reaction) for achieving high power of identification for emitter signals to replace the EID function in the electronics support measures (such as RWR). In this paper, we have shown that the proposed VNN can be used for identifying unambiguous emitters. In the future work, we will use the extra parameters of emitters such as angle of arrival and amplitude to form a new enlarged input feature vector for handling the problem of multiple ambiguous emitter; i.e., different types of emitters have similar characteristics.

References

[1] B. Edde, *Radar Principles, Technology, Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[2] B. Y. Tsui, *Microwave Receivers With Electronic Warfare Applications*. New York: Wiley, 1986.

[3] D. C. Schleher, *Introduction to Electronic Warfare*. New York: Artech House, 1986.

[4] G. B. Willson, "Radar classification using a neural network," in *Applications of Artificial Neural Networks*. Bellingham, WA: SPIE, 1990, vol. 1294, pp. 200–210.

[5] I. Howitt, "Radar warning receiver emitter identification processing utilizing artificial neural networks," in *Applications of Artificial Neural Networks*. Bellingham, WA: SPIE, 1990, vol. 1294, pp. 211–216.

[6] K. Mehrotra, C. K. Mohan, and S. Ranka, *Elements of Artificial Neural Networks*. Cambridge, MA: MIT Press, 1997.

[7] M. H. Hassoun, *Fundamentals of Artificial Neural Networks*. Cambridge, MA: MIT Press, 1995.

[8] S. K. Pal and S. Mitra, "Multilayer perceptron, fuzzy sets and classification," *IEEE Trans. Neural Networks*, vol. 3, pp. 683–696, Feb. 1992.

[9] J. M. Keller and H. Tahani, "Backpropagation neural networks for fuzzy logic," *Inform. Sci.*, vol. 62, pp. 205–221, 1992.

[10] S. Horikawa, T. Furuhashi, and Y. Uchikawa, "On fuzzy modeling using fuzzy neural networks with the backpropagation algorithm," *IEEE Trans. Neural Networks*, vol. 3, pp. 801–806, Sept. 1992.

[11] H. Ishibuchi, R. Fujioka, and H. Tanaka, "Neural networks that learn from fuzzy if-then rules," *IEEE Trans. Fuzzy Syst.*, vol. 1, pp. 85–97, May 1993.

[12] A. V. Ooyen and B. Nienhuis, "Improving the convergence of the backpropagation algorithm," *Neural Networks*, vol. 5, pp. 465–471, 1992.

[13] M. I. Skolnik, *Radar Handbook*. New York: McGraw-Hill, 1991.

Ching-Sung Shieh received the B.S. and M.S. degrees in electronic engineering from Chung-Yuan Christian University, Taoyuan, Taiwan, R.O.C., in 1984 and 1986, respectively. He received the Ph.D. degree from the National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., in 2001.

Since 1986, he has been with the Chung-Shan Institute of Science and Technology (CSIST), Taiwan, R.O.C., where he is currently an Associate Researcher. His current research interests include radar signal processing, control systems design, neural networks, estimation theory, and applications.

Dr. Shieh was elected an Honorary Member of the society by National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., in 2002.

Chin-Teng Lin (S'88–M'91–SM'99) received the B.S. degree in control engineering from National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., in 1986 and the M.S.E.E. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, in 1989 and 1992, respectively.

Since August 1992, he has been with the College of Electrical Engineering and Computer Science, National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., where he is currently a Professor and Chairman of the Electrical and Control Engineering Department. From 1998 to 2000, he served as Deputy Dean of the Research and Development Office, National Chiao-Tung University. His current research interests include fuzzy systems, neural networks, intelligent control, human–machine interface, image processing, pattern recognition, video and audio (speech) processing, and intelligent transportation systems (ITSs). He is the coauthor of *Neural Fuzzy Systems—A Neuro-Fuzzy Synergism to Intelligent Systems* (New York: Prentice-Hall, 1996), and author of *Neural Fuzzy Control Systems with Structure and Parameter Learning* (Singapore: World Scientific, 1994). He has published over 60 journal papers in the areas of soft computing, neural networks, and fuzzy systems, including about 40 IEEE Transactions papers.

Dr. Lin is a Member of Tau Beta Pi and Eta Kappa Nu. He is also a Member of the IEEE Computer, IEEE Robotics and Automation, and IEEE Systems, Man, and Cybernetics Societies. He was the Executive Council Member of the Chinese Fuzzy System Association (CFSA) from 1995 to 2001. He has been the President of CFSA since 2002, and Supervisor of the Chinese Automation Association since 1998. Since 2000, he has been the Chairman of the IEEE Robotics and Automation Society, Taipei Chapter, and the Associate Editor of the IEEE Transactions on Systems, Man, and Cybernetics, and IEEE Transactions on Fuzzy Systems, and *Automatic*a. He won the Outstanding Research Award granted by the National Science Council (NSC), Taiwan, R.O.C., in 1997, the Outstanding Electrical Engineering Professor Award granted by the Chinese Institute of Electrical Engineering (CIEE) in 1997, and the Outstanding Engineering Professor Award granted by the Chinese Institute of Engineering (CIE) in 2000. He was also elected to be one of the Ten Outstanding Young Persons in Taiwan, R.O.C., in 2000.