A Pyramidal Neural Network For Visual Pattern Recognition

Son Lam Phung, *Member, IEEE*, and Abdesselam Bouzerdoum, *Senior Member, IEEE*

***Abstract*—In this paper, we propose a new neural architecture for classification of visual patterns that is motivated by the two concepts of image pyramids and local receptive fields. The new architecture, called *pyramidal neural network (PyraNet)*, has a hierarchical structure with two types of processing layers: Pyramidal layers and one-dimensional (1-D) layers. In the new network, nonlinear two-dimensional (2-D) neurons are trained to perform both image feature extraction and dimensionality reduction. We present and analyze five training methods for PyraNet [gradient descent (GD), gradient descent with momentum, resilient backpropagation (RPROP), Polak–Ribiere conjugate gradient (CG), and Levenberg–Marquadrt (LM)] and two choices of error functions [mean-square-error (mse) and cross-entropy (CE)]. In this paper, we apply PyraNet to determine gender from a facial image, and compare its performance on the standard facial recognition technology (FERET) database with three classifiers: The convolutional neural network (NN), the k-nearest neighbor (k-NN), and the support vector machine (SVM).**

***Index Terms*—Gender classification, neural network (NN), pattern recognition, pyramidal architecture, receptive field, training algorithms.**

I. Introduction

ARTIFICIAL neural networks (ANNs) have found applications in many areas: pattern classification, function approximation, data clustering, and data compression, to name a few. A strength of ANNs is that they are able to learn a task from examples in an analogous manner to their biological counterparts, and therefore are suitable in situations where an analytic solution is hard to obtain. In machine vision, neural networks (NNs) have been used to solve numerous visual recognition problems, e.g., hand-written digit recognition [1], [2], optical character recognition [3], car detection [4], face detection [5], face recognition [7], [8], and facial expression analysis [9]. There are a number of NNs for visual recognition that deal with image pixels directly. The neocognitron, introduced by Fukushima [10], is a hierarchical NN motivated by a model suggested by Hubel and Wiesel [11] of the visual cortex in mammals. The convolutional neural networks (CNNs) by LeCun *et al.* [12], on the other hand, are built upon the ideas of local receptive fields, weight sharing and subsampling in the spatial or temporal domain. The neocognitron and the CNNs both retain two-dimensional (2-D) topology of the input image.

Manuscript received June 2, 2005; revised February 6, 2006 and May 19, 2006; accepted August 8, 2006. This work was supported in part by the Australian Research Council.

The authors are with the School of Electrical, Computer and Telecommunications Engineering, University of Wollongong, Wollongong, NSW 2522, Australia (e-mail: s.phung@ieee.org; a.bouzerdoum@ieee.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2006.884677

In this paper, we propose a new NN model for visual pattern recognition, called *pyramidal neural network (PyraNet)*. The new NN is motivated by the image pyramids that have been used successfully in image processing tasks (e.g., image decomposition, image segmentation, and image compression [13]). However, PyraNet differs from the image pyramids in that nonlinear processing at pyramidal stages can be tuned, through learning, for specific recognition tasks. The PyraNet architecture also possesses several strengths of 2-D NNs, including the integration of feature extraction and classification stages into a single structure, and the use of receptive fields to retain the 2-D spatial topology of image patterns. Furthermore, PyraNet has a systematic connection scheme, which simplifies greatly the task of network design and enables generic training algorithms to be devised. The paper is organized as follows. In Section II, we address the architectural aspects of PyraNet. In Section III, we derive five training algorithms for PyraNet that are based on two choices of error functions. These training algorithms are then analyzed in a face–nonface classification task. In Section IV, we apply PyraNet to classify gender based on facial images, and compare it with three other gender classifiers. Finally, in Section V, we present some concluding remarks.

II. PyraNet Network Model

In this section, we first describe the architecture of PyraNet, and then present a detailed mathematical model of the new network.

A. Network Architecture

PyraNet has a hierarchical multilayered architecture with two types of processing layers: 2-D pyramidal layers for feature extraction and data reduction and one-dimensional (1-D) feedforward layers for classification [Fig. 1(a)]. The first pyramidal layer is connected to the input image, and it is followed by one or more pyramidal layers. The last pyramidal layer is connected to 1-D layers. In this cascading structure, the output of one layer becomes the input to the next layer. A pyramidal layer consists of neurons arranged in a 2-D array; each neuron is connected to a specific rectangular region (i.e., the receptive field) in the previous layer. A 2-D neuron computes a weighted sum of inputs from its receptive field, and then applies a nonlinear activation function to produce an output signal. The role of the 1-D feedforward layers is to process the features produced by the pyramidal layers. Several 1-D layers may be needed in applications that involve the formation of complex decision boundaries. However, it is expected that the use of pyramidal layers for 2-D feature extraction will simplify the task of feature classification by the 1-D layers. The outputs of the last 1-D layer are taken as the network outputs that represent the categories of input patterns.

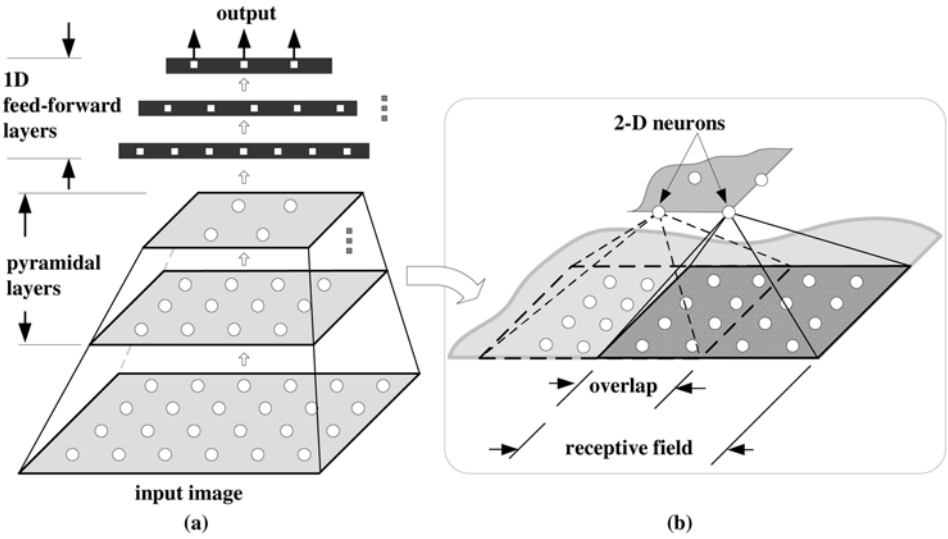

Fig. 1. Architecture of pyramidal NN: (a) network layers and (b) overlapping receptive fields.

TABLE I
ARCHITECTURAL NOTATION FOR PYRANET

Description	Symbol	Note
Input image size	N_0	$N_0 = H_0 \times W_0$ pixels
Numbers of layers	L_p, L_f, L	pyramidal, 1-D feed-forward, total
Layer index	l	$l = \begin{cases} 1, ..., L_p & \text{for pyramidal layers} \\ L_p + 1, ..., L & \text{for 1D feedforward layers} \end{cases}$
Activation function of layer l	$f_l(.)$	$l = 1, 2, ..., L$
Size of a receptive field in pyramidal layer l	r_l	$l \leq L_p$
Receptive field overlap in layer l	o_l	$l \leq L_p$
Gap factor for pyramidal layer l	g_l	$g_l = r_l - o_l$
Number of neurons in pyramidal layer l	N_l	$N_l = H_l \times W_l$, $H_l = \lfloor \frac{H_{l-1}-o_l}{g_l} \rfloor$, $W_l = \lfloor \frac{W_{l-1}-o_l}{g_l} \rfloor$
Weight associated with input position (i, j) to pyramidal layer l	$w^l_{i,j}$	$i = 1, ..., H_{l-1}$, $j = 1, ..., W_{l-1}$
Bias of neuron (u, v) in pyramidal layer l	$b^l_{u,v}$	$u = 1, ..., H_l$, $v = 1, ..., W_l$
Receptive field of 2-D neuron (u, v) in layer l	$R^l_{u,v}$	$\{(i, j) \mid (u-1)g_l + 1 \leq i \leq (u-1)g_l + r_l;$ $(v-1)g_l + 1 \leq j \leq (v-1)g_l + r_l\}$
Number of neurons in 1-D layer l	N_l	$l > L_p$
Weight from neuron m in 1-D layer $l-1$ to neuron n in layer l	$w^l_{m,n}$	$l > L_p$, $m = 1, ..., N_{l-1}$ $n = 1, ..., N_l$
Bias of neuron n in 1-D layer l	b^l_n	$l > L_p$ and $n = 1, ..., N_l$
Total number of trainable parameters (weights and biases)	P	$P = N_0 + 2 \sum_{l=1}^{L_p-1} N_l + N_{L_p} + \sum_{l=L_p+1}^{L} N_l \times (N_{l-1} + 1)$

B. Mathematical Model

The notation used to describe the functional aspects of PyraNet is summarized in Table I. The symbol l denotes the index of a network layer. For pyramidal layer l where $l = 1, 2, \ldots, L_p$, let $r_l \times r_l$ be the size of its receptive field and o_l be the horizontal or vertical overlap in pixels between two adjacent receptive fields. The difference g_l, i.e., $g_l = r_l - o_l$, is the gap between adjacent receptive fields. Since the sizes of adjacent pyramidal layers are related by $H_l = \lfloor (H_{l-1} - o_l)/g_l \rfloor$ and $W_l = \lfloor (W_{l-1} - o_l)g_l \rfloor$, g_l is also called the pyramidal step of layer l. Let $f_l(.)$ be the activation function of layer l.

Suppose we need to analyze an image pattern $\mathbf{x}$ of size $H_0 \times W_0$ pixels. The input image is partitioned into overlapping regions; each region consists of $r_1 \times r_1$ pixels and is considered as a receptive field to a neuron in layer 1 [Fig. 1(b)]. Each pixel in the input image is associated with an adjustable weight: Let $w^1_{i,j}$ denote the weight for image pixel at position (i, j). Let $b^1_{u,v}$ be the bias of neuron (u, v) of layer 1. Although the network layers in theory can be constructed from any type of neurons, e.g., radial basis function (RBF) neurons or sigmoidal neurons, we will focus on the PyraNet that is based on sigmoidal neurons in this paper. The output of 2-D neuron (u, v) in layer 1 is, therefore, given by

$$y^1_{u,v} = f_1 \left(\sum_{(i,j) \in R^1_{u,v}} w^1_{i,j} x_{i,j} + b^1_{u,v} \right) \tag{1}$$

where $R^1_{u,v}$ is the receptive field of neuron (u,v)

$$R^1_{u,v} = \{(i,j) | (u-1)g_1 + 1 \le i \le (u-1)g_1 + r_1, \times (v-1)g_1 + 1 \le j \le (v-1)g_1 + r_1\}. \quad (2)$$

Similarly, for other pyramidal layers, let $w^l_{i,j}$ be the synaptic weight associated with the input position (i,j) to layer l, and $b^l_{u,v}$ be the bias of neuron (u,v) in layer l. The output of the 2-D neuron is given by

$$y^l_{u,v} = f_l \left(\sum_{(i,j) \in R^l_{u,v}} w^l_{i,j} y^{l-1}_{i,j} + b^l_{u,v} \right), \qquad l = 2, 3, \ldots, L_p \quad (3)$$

where $R^l_{u,v}$ is the receptive field of neuron (u,v) in layer l

$$R^l_{u,v} = \{(i,j) | (u-1)g_l + 1 \le i \le (u-1)g_l + r_l, \times (v-1)g_l + 1 \le j \le (v-1)g_l + r_l\}. \quad (4)$$

The output $\{y^{L_p}_{u,v}\}$ of the last pyramidal layer is rearranged into a column vector, and used as input to the succeeding 1-D layer

$$\left\{ y^{L_p}_{u,v} | u = 1, \ldots, H_{L_p}; v = 1, \ldots, W_{L_p} \right\} \rightarrow \left\{ y^{L_p}_m | m = 1, \ldots, N_{L_p} \right\}. \quad (5)$$

In this paper, the 2-D and 1-D formats of the last pyramidal layer are used interchangeably. For 1-D feedforward layers, let $w^l_{m,n}$ be the synaptic weight from neuron m in layer $l-1$, to neuron n in layer l. Let b^l_n be the bias of neuron n in layer l; the output of the 1-D neuron is given by

$$y^l_n = f_l \left(\sum_{m=1}^{N_{l-1}} w^l_{m,n} y^{l-1}_m + b^l_n \right). \quad (6)$$

The outputs of the neurons in the last layer $\{y^L_n, n = 1, \ldots, N_L\}$ form the final network outputs.

PyraNet shares three properties with 2-D network models such as the CNNs [12], [14]: 1) the network is connected directly to pixels in the input image; 2) 2-D neurons are connected only to local regions; and 3) 2-D layers form a compressed representation of the preceding layers. Note that the 2-D layers in PyraNet are not limited to dyadic image pyramids: Depending on the application, each 2-D layer can have a different pyramidal step g_l. However, PyraNet differs from CNNs in a number of aspects. Most important, the CNNs are based on the weight-sharing principle, i.e., all neurons in a given feature map share the same set of weights or convolution mask. While weight sharing reduces the number of trainable parameters, it requires several planes or feature maps to be included in each convolution layer so that enough features can be extracted to support complex decision tasks. A feature map in convolutional network detects a feature at all input locations, whereas a 2-D neuron in PyraNet reveals the presence of a feature (not limited to low-level features such as edges or lines) at a specific input location. This is because each synaptic weight in PyraNet is associated with a specific input position.

PyraNet differs from a number of NNs with a pyramid structure that have been developed in recent years. Hoshino and Chao [15] proposed a pyramid network in which each output neuron is connected to all input and hidden neurons; the output neuron is, therefore, considered as the top of a pyramid that is formed from all input and hidden layers. Compared with PyraNet, Hoshino and Chao's network consists of fully connected layers and, therefore, less suitable for processing 2-D images. Cantoni and Petrosimo [16] proposed a pyramidal structure in which a Gaussian image pyramid is constructed from the input image, and the detail image at the top of the pyramid is used as input to a neural classifier. In this pyramidal structure, the coefficients of the low-pass filters are fixed, whereas in PyraNet the coefficients of the receptive fields are adaptive.

III. PYRANET TRAINING

For the new PyraNet to learn different visual recognition tasks, efficient and fast training algorithms must be devised. The objective of PyraNet training is to reduce iteratively an error function that is defined in terms of the network outputs and the desired outputs for a given application. In general, there are two approaches to network training [17]: The first approach uses the network to represent the discriminant function directly, whereas the second approach uses the network to model the posterior probabilities of class membership. Accordingly, there are two types of error functions that are commonly used: The mean-square-error (mse) and the cross-entropy (CE). Both error functions will be addressed in this section. The various definitions used in PyraNet training are summarized in Table II. For each image $\mathbf{x}^k$ in the training set, let vector $\mathbf{d}^k$ be the corresponding desired outputs.

- **MSE function.** The overall error is defined as the mse between the network outputs and the desired outputs

$$E_{\text{mse}}(\mathbf{w}) = \frac{1}{K \times N_L} \sum_{k=1}^{K} \sum_{n=1}^{N_L} |y^{L,k}_n - d^k_n|^2 \quad (7)$$

  where $\mathbf{w}$ is a vector representing all trainable parameters. This error function has an origin in regression and interpolation applications. For a network trained with the mse function, when a new input image is presented to the network, the outputs provide a classification directly. It has been shown [17] that if the target class membership can be defined as a deterministic function of the input with additive Gaussian noise, the mse function can be derived from the principle of maximum likelihood.
- **CE error function.** This error function is sometimes used in classification problems when networks are trained to estimate the posterior probabilities of class membership. One output neuron is typically allocated for each class (i.e., the one-of-C encoding), and the estimated probabilities p^k_n must satisfy the following constraints:

$$p^k_n \ge 0, \qquad n = 1, 2, \ldots, N_L \text{ and } \sum_{n=1}^{N_L} p^k_n = 1. \quad (8)$$

TABLE II
NOTATION FOR PYRANET TRAINING ALGORITHMS

Description	Symbol	Formula
Training image index	k	$k = 1, 2, ..., K$
Input image k	$\mathbf{x}^k$	$\mathbf{x}^k = (x^k_{i,j}, i = 1, ..., H_0; j = 1, ..., W_0)$
Desired output sample k	$\mathbf{d}^k$	$\mathbf{d}^k = (d^k_1, d^k_2, ..., d^k_{N_L})^T$
Network input or output of layer 0	$y^{0,k}_{i,j}$	$(y^{0,k}_{i,j}) = (x^k_{i,j}), i = 1, ..., H_0; j = 1, ..., W_0$
Weighted sum input to neuron (u, v) in pyramidal layer l, for input sample k	$s^{l,k}_{u,v}$	$s^{l,k}_{u,v} = \sum_{i=i_{\text{low}}}^{i_{\text{high}}} \sum_{j=j_{\text{low}}}^{j_{\text{high}}} w^l_{i,j}\ y^{l-1}_{i,j} + b^l_{u,v}$ $u = 1, ..., H_l; v = 1, ..., W_l$ $i_{\text{low}} = (u-1)g_l + 1; i_{\text{high}} = (u-1)g_l + r_l$ $j_{\text{low}} = (v-1)g_l + 1; j_{\text{high}} = (v-1)g_l + r_l$
Output of neuron (u, v) in pyramidal layer l for input image k	$y^{l,k}_{u,v}$	$y^{l,k}_{u,v} = f_l(s^{l,k}_{u,v})$
2-D to 1-D mapping at the last pyramidal layer		$\{y^{L_p,k}_{i,j},\ i = 1, ..., H_{L_p}; j = 1, ..., W_{L_p}\}$ $\rightarrow \{y^{L_p,k}_m, m = 1, ..., N_{L_p}\}$
Weighted sum input to neuron n in 1-D layer l	$s^{l,k}_n$	$s^{l,k}_n = \sum_{m=1}^{N_{l-1}} w^l_{m,n}\ y^{l-1,k}_m + b^l_n$
Output of neuron n in 1-D layer l	$y^{l,k}_n$	$y^{l,k}_n = f_l(s^{l,k}_n)$
Softmax mapping for sample k (only for CE)	p^k_n	$p^k_n = \exp(y^k_n) / \sum_{i=1}^{N_L} \exp(y^k_i)$
The nth error for image k	e^k_n	$e^k_n = \begin{cases} y^{L,k}_n - d^k_n, & \text{for MSE} \\ p^k_n - d^k_n, & \text{for CE} \end{cases}$
Error function	$E(\mathbf{w})$	$\begin{cases} E_{MSE} = \frac{1}{K \times N_L} \sum_{k=1}^{K} \sum_{n=1}^{N_L} (y^k_n - d^k_n)^2, \\ E_{CE} = -\sum_{k=1}^{K} \sum_{n=1}^{N_L} d^k_n \ln p^k_n \end{cases}$
Error sensitivity of neuron (u, v) in pyramidal layer l	$\delta^{l,k}_{u,v}$	$\delta^{l,k}_{u,v} = \partial E / \partial s^{l,k}_{u,v}, \quad l \le L_p$
Error sensitivity of neuron n in 1-D layer l	$\delta^{l,k}_n$	$\delta^{l,k}_n = \partial E / \partial s^{l,k}_n, \quad l > L_p$

Let $d^k_n, n = 1, 2, \ldots, N_L$ be the desired probabilities for the training sample $\mathbf{x}^k$, i.e.,

$$d^k_n = \begin{cases} 1, & \text{if } \mathbf{x}^k \text{ belongs to class } n \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

The CE error function [17] is defined as

$$E_{\text{CE}}(\mathbf{w}) = -\sum_{k=1}^{K} \sum_{n=1}^{N_L} d^k_n \ln p^k_n. \tag{10}$$

There are two common approaches to minimizing the error function in (10) subject to the constraints in (8). The first approach uses Lagrange multipliers [18] to enforce the summation constraint implicitly, whereas the second approach [17] defines p^k_n so that both constraints are met explicitly. Both approaches require calculation of the error gradient using the backpropagation method. In this paper, we adopt the second approach, which applies a *softmax* function [19] on the output neurons to obtain an estimate of *a posteriori* probabilities

$$p^k_n = \exp(y^{L,k}_n) / \sum_{i=1}^{N_L} \exp\left(y^{L,k}_i\right), \qquad n = 1, 2, \ldots, N_L. \tag{11}$$

Using this approach, it turns out that, except for the output layer, the gradients of the mse and CE functions can be computed in a similar way, as shown in Section III-A.

A. PyraNet Error Gradient Computation

The gradient is computed through error sensitivities, which are defined as the partial derivatives of the error function E (E can be E_{mse} or E_{CE}) with respect to the weighted sum input

$$\text{for 2-D neurons:} \quad \delta^{l,k}_{u,v} = \frac{\partial E}{\partial s^{l,k}_{u,v}}, \qquad l \le L_p \tag{12}$$

$$\text{for 1-D neurons:} \quad \delta^{l,k}_n = \frac{\partial E}{\partial s^{l,k}_n}, \qquad l > L_p \tag{13}$$

where $s^{l,k}_{u,v}$ and $s^{l,k}_n$ denote the weighted sum input to a 2-D neuron and 1-D neuron, respectively. Using Table II and the chain rule of differentiation, we can express the error sensitivities as follows.

- **Output layer**
  — mse case

$$\delta^{L,k}_n = \frac{2}{K \times N_L} e^k_n f'_L\left(s^{L,k}_n\right), \qquad n = 1, 2, \ldots, N_L \tag{14}$$

  — CE case

$$\delta^{L,k}_n = e^k_n f'_L\left(s^{L,k}_n\right), \qquad n = 1, 2, \ldots, N_L. \tag{15}$$

  We should note that for the mse case $e^k_n = y^{L,k}_n - d^k_n$, whereas for the CE case $e^k_n = p^k_n - d^k_n$.
- **Other 1-D layers** $(L_p < l < L)$

$$\delta^{l,k}_n = f'_l(s^{l,k}_n) \sum_{m=1}^{N_{l+1}} \delta^{l+1,k}_m w^{l+1}_{n,m}, \qquad n = 1, 2, \ldots, N_l. \tag{16}$$

- **Last pyramidal layer** $(l = L_p)$
  The error sensitivities $\{\delta_n^{L_p,k}\}$ are calculated using (16) for $l = L_p$, but then rearranged into a 2-D grid

$$\left(\delta_n^{L_p,k},\ n = 1,\ldots,N_{L_p}\right) \rightarrow \times \left(\delta_{u,v}^{L_p,k},\ u = 1,\ldots,H_{L_p}; v = 1,\ldots,W_{L_p}\right). \quad (17)$$

- **Other pyramidal layers** $(l < L_p)$

$$\delta_{u,v}^{l,k} = f_l^{'}\left(s_{u,v}^{l,k}\right) w_{u,v}^{l+1} \times \sum_{i=i_{\text{low}}}^{i_{\text{high}}} \sum_{j=j_{\text{low}}}^{j_{\text{high}}} \delta_{i,j}^{l+1,k} \quad (18)$$

where $u = 1, 2, \ldots, H_l, v = 1, 2, \ldots, W_l$, and

$$i_{\text{low}} = \left\lceil \frac{u - r_{l+1}}{g_{l+1}} \right\rceil + 1$$
$$i_{\text{high}} = \left\lfloor \frac{u-1}{g_{l+1}} \right\rfloor + 1 \quad (19)$$
$$j_{\text{low}} = \left\lceil \frac{v - r_{l+1}}{g_{l+1}} \right\rceil + 1$$
$$j_{\text{high}} = \left\lfloor \frac{v-1}{g_{l+1}} \right\rfloor + 1. \quad (20)$$

The error gradient can now be obtained as follows.

- **1-D layers** $(L_p < l \leq L)$
  — weights $w_{m,n}^l$

$$\frac{\partial E}{\partial w_{m,n}^l} = \sum_{k=1}^{K} \delta_n^{l,k} y_m^{l-1,k} \quad (21)$$

where $m = 1, 2, \ldots, N_{l-1}$ and $n = 1, 2, \ldots, N_l$;
  — biases b_n^l:

$$\frac{\partial E}{\partial b_n^l} = \sum_{k=1}^{K} \delta_n^{l,k} \quad (22)$$

where $n = 1, 2, \ldots, N_l$.

- **Pyramidal layers** $(l \leq L_p)$
  — weights $w_{i,j}^l$

$$\frac{\partial E}{\partial w_{i,j}^l} = \sum_{k=1}^{K} \left\{ y_{i,j}^{l-1,k} \times \sum_{u=u_{\text{low}}}^{u_{\text{high}}} \sum_{v=v_{\text{low}}}^{v_{\text{high}}} \delta_{u,v}^{l,k} \right\} \quad (23)$$

where $i = 1, 2, \ldots, H_{l-1}, j = 1, 2, \ldots, W_{l-1}$, and

$$u_{\text{low}} = \left\lceil \frac{i - r_l}{g_l} \right\rceil + 1$$
$$u_{\text{high}} = \left\lfloor \frac{i-1}{g_l} \right\rfloor + 1 \quad (24)$$
$$v_{\text{low}} = \left\lceil \frac{j - r_l}{g_l} \right\rceil + 1$$
$$v_{\text{high}} = \left\lfloor \frac{j-1}{g_l} \right\rfloor + 1 \quad (25)$$

and note that $y_{i,j}^{0,k}$ refers to the input sample;
  — biases $b_{u,v}^l$

$$\frac{\partial E}{\partial b_{u,v}^l} = \sum_{k=1}^{K} \delta_{u,v}^{l,k} \quad (26)$$

where $u = 1, 2, \ldots, H_l$ and $v = 1, 2, \ldots, W_l$.
This completes the derivation of the gradient for PyraNet.

B. PyraNet Training Algorithms

Once the error gradient is derived, numerous optimization algorithms for minimizing E can be applied to train PyraNet [20]–[22]. In this paper, we focus on five representative training algorithms, namely gradient descent (GD) [23], gradient descent with momentum and variable learning rate (GDMV) [24], resilient backpropagation (RPROP) [25], conjugate gradient (CG) [20], and Levenberg–Marquardt (LM) [26]. Three algorithms GD, GDMV, and RPROP are first-order optimization methods. The CG algorithm can be considered as an intermediate between first- and second-order methods, whereas the LM algorithm is a trust-region method that uses the Gauss–Newton approximation of the Hessian matrix. Since details of these algorithms can be found in the given references, we only summarize here their main characteristics (see Table III).

Computation of the Jacobian matrix for PyraNet is similar to computation of the gradient ∇E, as shown in (12)–(26). However, we need to modify the definitions of error sensitivities as follows:

$$\delta_{u,v}^{l,k}(q) = \frac{\partial e_q^k}{\partial s_{u,v}^{l,k}}, \qquad l \leq L_p \quad (27)$$
$$\delta_n^{l,k}(q) = \frac{\partial e_q^k}{\partial s_n^{l,k}}, \qquad l > L_p. \quad (28)$$

That is error sensitivities are now defined for each network error e_q^k, where $q = 1, 2, \ldots, N_L$, instead of the overall error function E.

C. Analysis of PyraNet Training Algorithms

In this section, we analyze the convergence speed and computation load of the five training algorithms previously presented. The objective is to identify PyraNet training algorithms that are fast, less computation-intensive, and capable of handling large training sets. Clearly, the convergence speed of an algorithm is affected by the choice of training parameters. For example, a small learning rate α in the GD algorithm leads to a slow decrease in the mse whereas a large α may cause training to diverge. Since it is impractical to evaluate all choices of training parameters, we have determined the parameters through trial-and-error. Nevertheless, the training parameters in this paper reflects the basic trend in convergence speed of the respective algorithms. Furthermore, we find that the selected parameters work for large and different training sets.

The training algorithms are assessed in a face–nonface classification task, in which the aim is to determine whether an image is a face or nonface pattern [28]. We used a data set taken from a large face and skin detection database [29]. The data set consists of 20 000 images: 10 000 face patterns that are manually cropped from Web images, and 10 000 nonface patterns that are

TABLE III
PYRANET TRAINING ALGORITHMS

Algorithm	Description
Gradient Descent **(GD)** [23]	**Weights are updated along the negative gradient** $\Delta\mathbf{w}(t) = -\alpha\nabla E(t)$ α is scalar learning rate, $\alpha > 0$
GD With Momentum and Variable Learning Rate **(GDMV)** [24]	**Weight update is a linear combination of gradient and previous weight update** $\Delta\mathbf{w}(t) = \lambda\ \Delta\mathbf{w}(t-1) - (1-\lambda)\ \alpha(t)\ \nabla E(t)$ λ is momentum parameter, $0 < \lambda < 1$ $\alpha(t)$ is adaptive scalar learning rate
Resilient Backpropagation **(RPROP)** [25]	**Weight update depends only on the sign of gradient** $\Delta w_i(t) = -\text{sign}\{\frac{\partial E}{\partial w_i}(t)\} \times \Delta_i(t)$ $\Delta_i(t)$ is adaptive step specific to weight w_i, defined as $\Delta_i(t) = \begin{cases} \eta_{\text{inc}}\ \Delta_i(t-1), & \text{if } \frac{\partial E}{\partial w_i}(t)\ \frac{\partial E}{\partial w_i}(t-1) > 0 \\ \eta_{\text{dec}}\ \Delta_i(t-1), & \text{if } \frac{\partial E}{\partial w_i}(t)\ \frac{\partial E}{\partial w_i}(t-1) < 0 \\ \Delta_i(t-1), & \text{otherwise} \end{cases}$ $\eta_{\text{inc}} > 1$, $0 < \eta_{\text{dec}} < 1$: scalar terms
Conjugate Gradient (**CG**) [20]	**Weights are updated along directions mutually conjugated w.r.t. Hessian matrix** $\Delta\mathbf{w}(t) = \alpha(t)\mathbf{s}(t)$, where search direction defined as $\mathbf{s}(t) = \begin{cases} -\nabla E(t) & \text{if } t \equiv 1 \pmod P \\ -\nabla E(t) + \beta(t)\ \mathbf{s}(t-1) & \text{otherwise} \end{cases}$ learning step $\alpha(t)$ is found through a line search [27]. $\beta(t)$ is updated according to the following Polak-Ribiere formula $\beta(t) = \frac{[\nabla E(t) - \nabla E(t-1)]^T\ \nabla E(t)}{\|\nabla E(t-1)\|^2}$
Levenberg-Marquardt **(LM)** [26]	**2nd-order Taylor expansion and Gauss-Newton approximation of Hessian matrix** $\Delta\mathbf{w}(t) = -[\mathbf{J}^T\mathbf{J} + \mu\mathbf{I}]^{-1}\ \nabla E$ $\mathbf{J}$ is the Jacobian matrix defined as $J_{(q-1)K+k,i} = \frac{\partial e_q^k}{\partial w_i}, q = 1, 2, ..., N_L; k = 1, 2, ..., K; i = 1, 2, ..., P$ Gradient ∇E is computed through the Jacobian matrix $\mathbf{J}$. μ is an adaptive parameter controlling the size of the trust region.

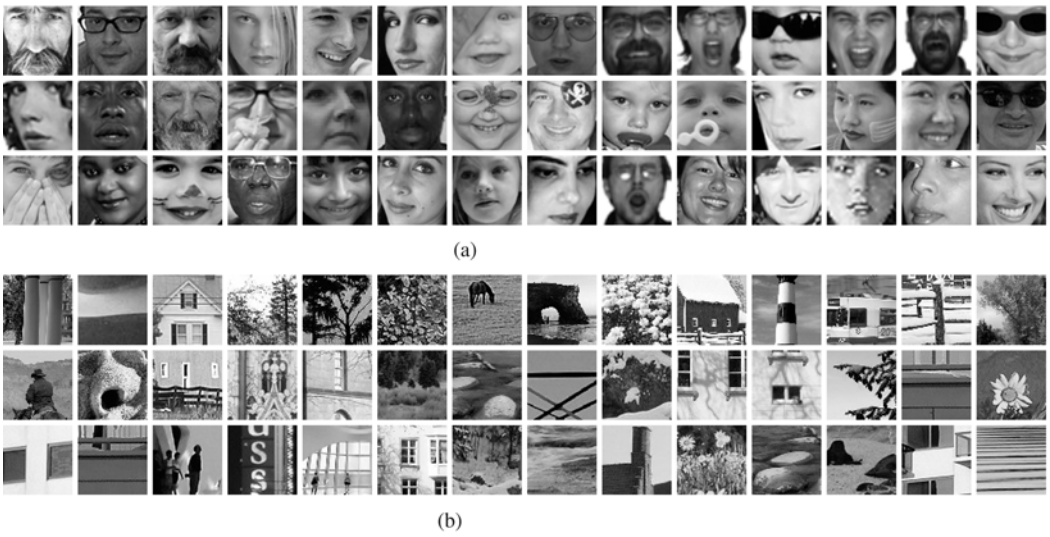

Fig. 2. Example images in the face–nonface data set used for comparing training algorithms. (a) Face patterns. (b) Nonface patterns.

randomly extracted from thousands of scenery photos. Sample face and nonface patterns in this data set are shown in Fig. 2. The entire data set of 20 000 images was divided into five subsets of equal size, and training was conducted in five folds. In each fold, a design set (for network training and validation) of 16 000 images was formed from four subsets, and a test set of 4000 images was formed from the remaining fifth subset. The design set was split into a training set (90%) and a validation set (10%); hence, each network was trained on 14 400 images.

1) Comparison of Five PyraNet Training Algorithms (MSE Function): Since the focus of this section is on training algorithms rather than network structure, we only present here the results with one large network structure. The input image size is 20×20 pixels, which is similar to the image sizes used by several authors for face–nonface classification [5], [30]–[32]. The PyraNet has two pyramidal layers and an output layer with one neuron. The receptive fields of the first and second pyramidal layers are 5×5 pixels and 4×4 pixels, respectively; the overlap factor in both cases is 2 pixels. The activation function is the $f(x) = (e^x - e^{-x})/(e^x + e^{-x})$, which is widely used for ANNs. The PyraNet has a total of 481 trainable parameters (weights and biases). Using the five training algorithms

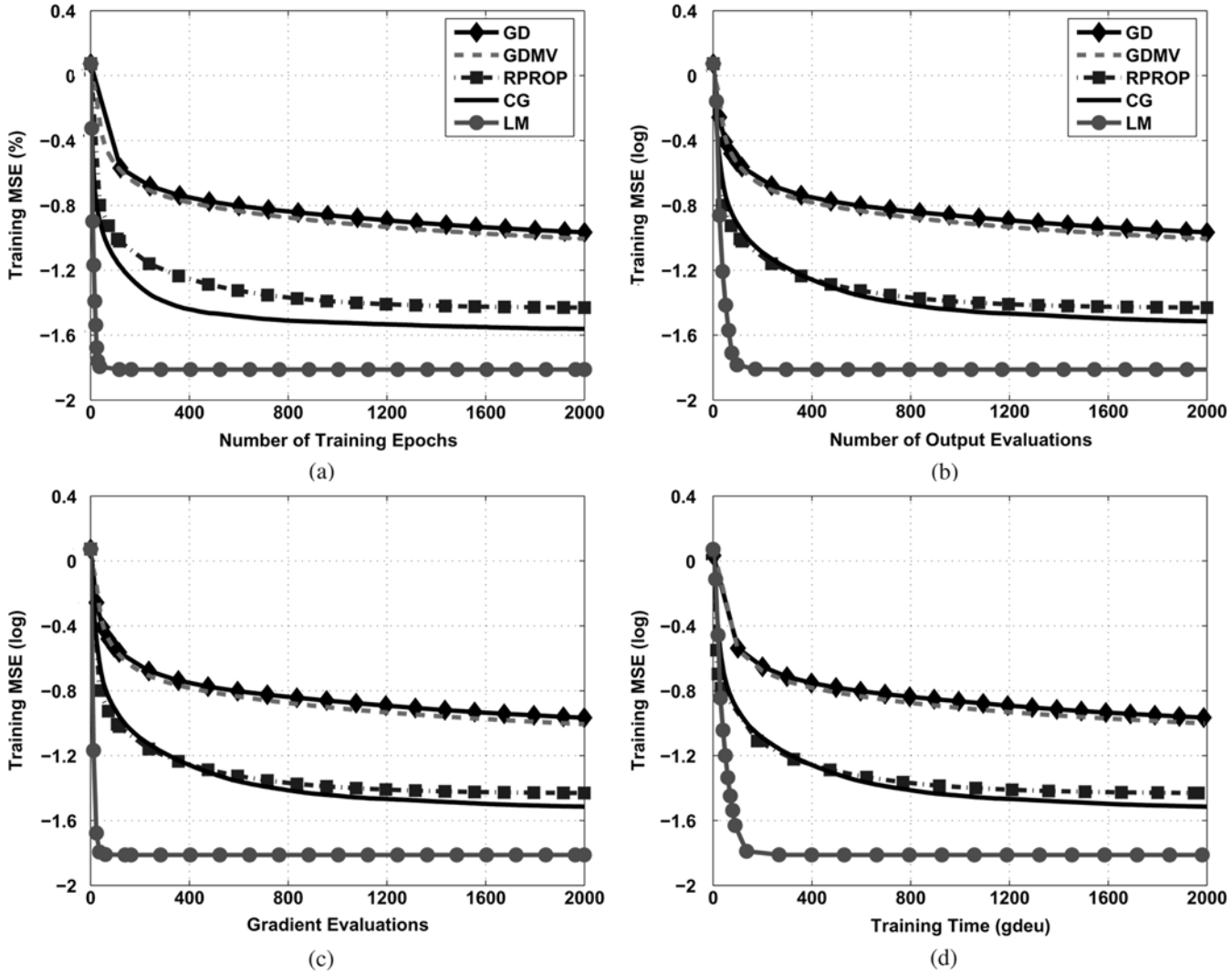

Fig. 3. Comparison on the face–nonface classification data set of the five training algorithms in terms of the training mse versus (a) the number of training epochs, (b) the number of output evaluations, (c) the number of gradient evaluations, and (d) the training time. Training images $K = 14\,400$, image size 20×20, network parameters $P = 481$.

and the mse error function, the PyraNet is trained to produce an output of 1.0 for a face pattern and -1.0 for a nonface pattern. In testing, a threshold is applied to the network output to determine the class label.

In each fold, two networks with different weight initialization were created. The two networks were then trained for 2000 epochs, using each of the five training algorithms, namely GD, GDMV, RPROP, CG, and LM. For comparison purposes, various indicators including the training mse, the training time, the number of training epochs, and the number of output evaluations[1] were recorded and averaged over five folds. To obtain a more machine-independent comparison of the training speed, we measured the training time in terms of the *gradient descent epoch time unit* or *gdeu*. One *gdeu* is defined as the average time taken to perform one GD training epoch on a fixed training set and a fixed-size network; it remains stable throughout the GD training process. On our PCs with P4 2.8-GHz CPU and 1-GB RAM, one *gdeu* time unit is approximately 3.55 s for the above training configuration (14 400 training images of size 20×20 pixels and 481 trainable parameters).

Comparison results of the five training algorithms are shown in Fig. 3. At any given epoch count, Fig. 3(a) shows that the LM algorithm achieves the smallest mse among the five algorithms, and the CG algorithm achieves lower mse compared to the RPROP algorithm. However, the amount of computation required for each epoch differs among the five algorithms. In this experiment, an LM epoch uses on average 2.9 output evaluations, whereas a CG epoch needs about 2.4 output evaluations and a GD, GDMV, and RPROP epochs use exactly one output evaluation. Furthermore, an LM epoch takes 3.8 *gdeu* on average, whereas a CG epoch takes 2.4 *gdeu* and a GD, GDMV, and RPROP epochs take approximately 1.0 *gdeu*. At the same number of output evaluations, Fig. 3(b) shows that the LM algorithm reaches lower mse compared to the RPROP and CG algorithms. For example, to reach a mse of 0.05, the LM algorithm uses 43.9 output evaluations, whereas the RPROP and CG algorithms need 515.7 and 475.5 output evaluations, respectively. The RPROP and the CG algorithms have similar performances,

[1]An output evaluation involves computing network outputs for the entire training set.

TABLE IV
COMPUTATION AND MEMORY REQUIRED BY FIVE TRAINING ALGORITHMS: TRAINING IMAGES $K = 14400$, IMAGE SIZE 20×20, NETWORK PARAMETERS $P = 481$, TARGET mse $= 0.04$

Measure	GD	GDMV	RPROP	CG	LM
Time ($gdeu$)	26941.5	20726.2	1035.5	735.6	65.5
Epochs	26941.5	20520.5	1042.5	308.9	17.2
Network output evaluations	26941.5	20520.5	1042.5	736.1	50.0
Error gradient evaluations	26941.5	20520.5	1042.5	736.1	17.2
Hessian matrix evaluations	0.0	0.0	0.0	0.0	17.2
Memory usage	$K \times P$	$K \times P$	$(K+2) \times P$	$(K+3) \times P$	$K \times P \times N_L + P^2$

with the CG algorithm reaching smaller mses when the number of output evaluations is above 390.0.

In terms of training time, Fig. 3(d) shows that all four algorithms—GDMV, RPROP, CG, and LM—are faster than the standard GD algorithm. The GDMV algorithm achieves some speed advantage over the GD algorithm only if the momentum parameter λ is within the range of (0.6, 0.9). Furthermore, imposing an upper limit on the adaptive learning rate α makes GDMV training more stable. The RPROP and CG algorithms have similar speeds, and they both converge faster than the GDMV algorithm. Although the LM algorithm requires several inversions of Hessian matrix in each training epoch, it is the fastest among the five training algorithms.

We also trained the PyraNet using the five algorithms until the mse reached 0.04 (the CR on the training set at this mse is approximately 99%). The computation required to reach the target mse is summarized in Table IV. Among the five training algorithms, the LM algorithm is the fastest, taking on average 65.5 *gdeu* to reach the target mse. The CG algorithm requiring 735.6 *gdeu* is the next fastest, followed by the RPROP algorithm that takes 1035.5 *gdeu*. The GDMV and GD algorithms are the slowest, taking over 20 700 and 26 900 *gdeu*, respectively. To achieve the target mse, the LM algorithm is 11.2 times faster than the CG algorithm, and 15.8 times faster than the RPROP algorithm.

An estimate of the memory requirements by the five algorithms is given in Table IV. Note that there is always a tradeoff between memory usage and speed for any given algorithm. For example, we can reduce memory needs, at the cost of increased training time, by partitioning the training set into smaller subsets. The memory estimate shown in Table IV is calculated for straightforward implementations of the training algorithms, in which memory storage is allocated for intermediate results such as the weighted-sum inputs and the outputs of individual neurons. Among the five training algorithms, the GD and GDMV algorithms require almost the same amount of memory, which is proportional to the training set size (K) and the number of trainable parameters (P). The RPROP algorithm requires slightly more memory to store the signs of the previous error gradient and the learning rates for individual weights. Compared to the RPROP algorithm, the CG algorithm requires more memory because in addition to keeping the search direction and the previous error gradient, the CG algorithm involves evaluation of several temporary networks during the 1-D optimization step. Nevertheless, memory usage of the CG algorithm remains proportional to the training size and the number of weights. Among the five algorithms, the LM algorithm uses the largest amount of memory because it requires the computation of Jacobian matrix, which has $K \times P \times N_L$ entries, as well as the Hessian matrix, which has P^2 entries. However, we should note that there are reduced-memory but slower implementations of the LM algorithm [24].

We also compared the generalization performance of the networks produced by the five training algorithms. In each validation fold, the networks produced at different numbers of epochs (up to 2000 epochs) were run on the validation set. The best network on the validation set was then evaluated on the test set. The receiver operating characteristics (ROC) curves and the classification rates of the networks trained with different algorithms are shown in Fig. 4. The figure shows that the RPROP, CG, and LM algorithms produce networks with similar classification rates of 97.3%, 97.2, and 97.5%, respectively (the difference is no more than 0.3%), whereas the GD and GDMV algorithms generate networks with lower classification rates (96.5% and 96.6%). These comparative performances are consistent with the training speed comparison presented previously. For the slower algorithms such as GD and GDMV, more training time is needed to find a good solution.

2) Comparison of mse and CE Error Functions: We also compare training with the mse and CE error functions in two aspects: Training speed and generalization capability. The network has a similar structure as in the previous experiment, except that the output layer now has two neurons (as needed by the CE approach) and the total number of trainable parameters is 486. In our comparison, the RPROP algorithm is used because of two main reasons. First, as shown in the previous section the RPROP algorithm is reasonably fast (one of the fastest among the first-order training methods) and requires only modest memory (almost the same as the standard GD algorithm). Second, the RPROP algorithm does not rely on specific shapes (e.g., quadratic) of the error surfaces; it is, therefore, more suitable for a fair comparison of different error functions.

Since the two error functions (mse and CE) are different, we compare training speed in terms of the classification error rate on the training set versus the training time. Generalization capability is compared using both the ROC curves and the classification rates on the test set. Fig. 5(a) shows that the classification error rate on the training set in the mse case decreases only slightly faster compared to the CE case. At the same training point, the difference between the training classification error rates of mse and CE is less than 0.3%. Fig. 5(b) shows that there is only a small difference between the ROC curves of networks trained with the mse and CE error functions. The classification rates on the test set (averaged over five validation folds) for the mse and CE functions are 97.23% and 97.27%, respectively. We also found that when the mse function is used there is very little difference in the training speed and classification rate between a PyraNet with one output neuron and a PyraNet with two output neurons.

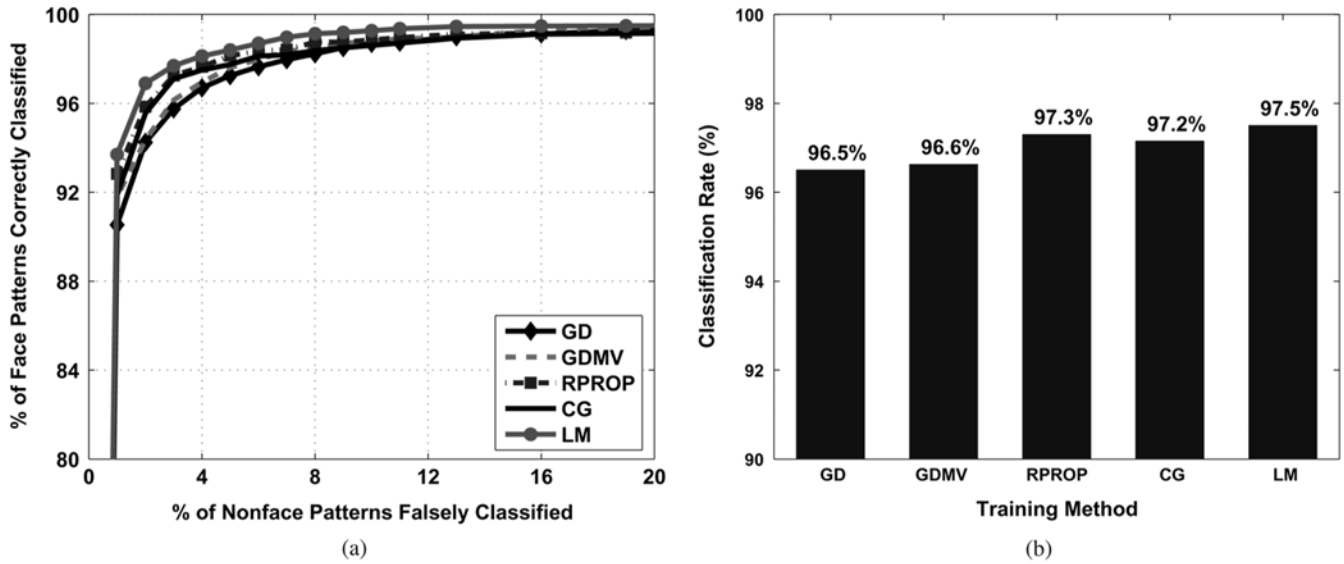

Fig. 4. Comparison of face–nonface classification performances of networks produced by the five training algorithms: (a) ROC curves and (b) classification rates.

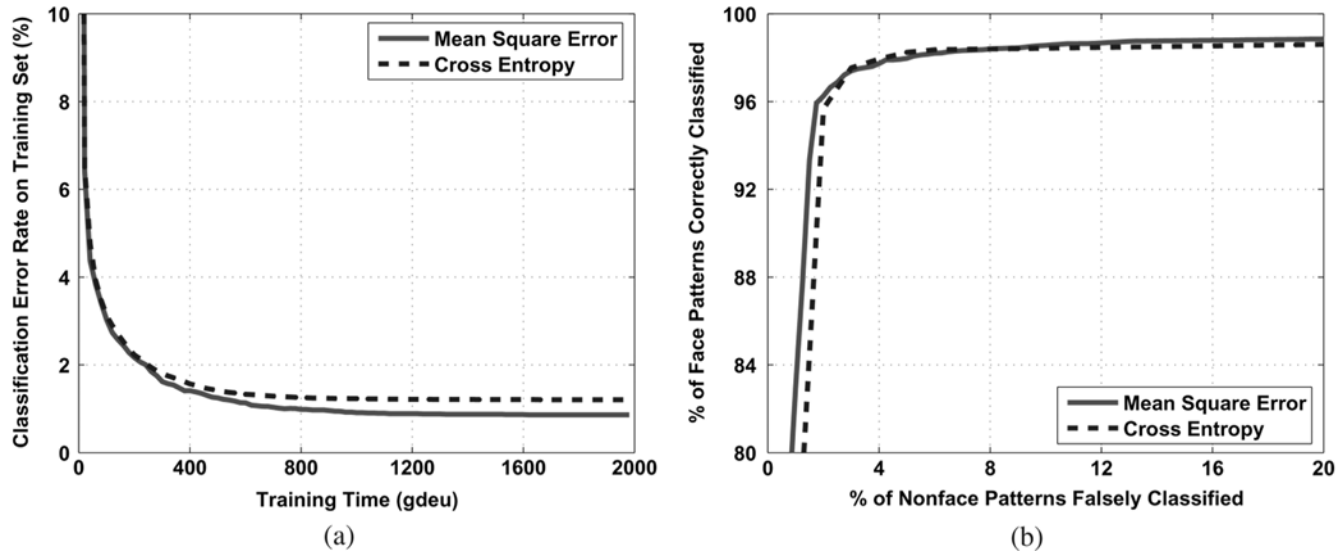

Fig. 5. Comparison of the mse and CE error functions in terms of (a) training speed (training set= 14 400 images, image size 20×20, network parameters $P = 486$) and (b) generalization capability (test set = 4000 images).

In summary, this section shows that the new NN can learn a pattern recognition task using the presented training algorithms. Compared to the GD and GDMV algorithms, the RPROP and CG algorithms have similar memory usage but are superior in convergence speed. Compared to the LM algorithm, the PROP and CG algorithms have lower convergence speeds but require much less memory storage. Furthermore, the PyraNet can be trained using either mse or CE error function. Having analyzed PyraNet training algorithms, we will focus next on one application of the new network in gender classification.

IV. Gender Classification of Facial Images Using PyraNet

In our daily interactions, we can infer a person's gender quite accurately based on information such as facial appearance, hairstyle, body posture, voice, and clothing. On one hand, it is possible to determine gender through visual cues such as hairstyle and clothing [33]; these are more extrinsic features that vary greatly even within the same gender group. On the other hand, several visual heuristics about the face can be used to differentiate different genders. For example, compared to men, women tend to have smaller and thinner nose, thinner and higher eyebrows, plumper cheeks, and a softer facial outline. In this paper, we train the new PyraNet to automatically determine a person's gender based on a facial image. Gender classification of facial images is useful in many situations. For example, using a video camera we can count the number of male or female customers that have entered a shop. Human-computer interfaces can be programmed to present appropriate options for male and female users. Gender information of the person can be incorporated into algorithms for face recognition or facial expression analysis.

Several techniques have been proposed for gender classification of facial images (see Table V). Golomb *et al.* [34] used an autoencoder with 40 hidden units to extract 40 components from an input image of size 30×30 pixels; these components are then

TABLE V
SUMMARY OF GENDER CLASSIFICATION APPROACHES

Author	Year	Approach	Image Sizes	Model Size	Database – No. Images	Classification Rate (%)
Golomb et al. [34]	1991	Auto-encoder and MLP	30×30	421	custom – 90	91.9
Gray et al. [35]	1995	* Perceptron * MLP	30×30 30×30	901 9911	custom – 90	81.0 83.5
Gutta et al. [36]	2000	decision tree of RBF network ensembles	64×72	–	FERET – 3006	96.0
Jain and Huang [37]	2004	ICA and LDA	64×96	1229000	FERET – 500	99.3
Moghaddam and Yang [38]	2002	SVMs * RBF kernel * cubic polynomial kernel	21×12	75699	FERET – 1755	 96.6 95.2
Wu et al. [39]	2003	AdaBoost and Harr-like features	24×24, 36×36	3400	custom – 13600	88.0

classified using a multilayer perceptron (MLP). Their system achieved a classification rate of 91.9% on a data set of 90 images. Using the same data set as Golomb *et al.*, Gray *et al.* [35] compared the performance of a simple perceptron and an MLP with one hidden layer of ten neurons, across a range of image sizes: 10×10, 15×15, 22×22, 30×30, and 60×60. The perceptron has a maximum CR of 81%, and the MLP achieves a maximum CR of nearly 83.5%. Among the tested image sizes, they found that the image size 30×30 gives a higher classification rate. Compared to the perceptron, the MLP performs better at large image sizes (30×30 and 60×60) and worse at smaller image sizes (10×10, 15×15, and 22×22).

Gutta [36] used several ensembles of RBF networks arranged in a decision tree (DT) structure. In their approach, each ensemble is designed to operate in a region of the input space, and the RBF networks are trained on original images as well as their distorted variations. On a facial recognition technology (FERET) data set of 3006 images, the RBF/DT hybrid classifier achieved a classification rate of 96%. Moghaddam and Yang [38] used support vector machines (SVMs) with RBF and polynomial kernels. On 1755 FERET images, they achieved classification rates of 96.6% and 95.2% with the RBF kernel and cubic polynomial kernel, respectively. Moghaddam and Yang found that the difference between classification rates when using low-resolution (21×12) and high-resolution (84×48) image is only 1%. Recently, Wu *et al.* [39] proposed a gender classifier that is based on the AdaBoost algorithm and a set of simple Harr-like features. Their classifier cascade, which was originally proposed by Viola and Jones [40] for face–nonface classification, has an accuracy of 88.0%, evaluated on 13 600 face images. Jain and Huang [37] applied independent component analysis to extract 200 features from a 64×96 face image. The 200 features are then processed by the Fisher linear classifier; the reported accuracy is 99.3%, obtained when the gender classifier is trained on 200 FERET images and tested on 300 FERET images.

A. Data Preparation and Evaluation Procedure

Our paper is conducted on a standard and publicly available database—the FERET database [41]. This database consists of 14 051 gray-scale images of human faces, with views ranging from frontal to left and right profiles, and it is divided into several data sets. There are two data sets for frontal faces as follows:

- data set fa has 1762 images, of which 1152 are male patterns and 610 are female patterns;
- data set fb has 1518 images, of which 968 are male patterns and 550 are female patterns.

Since there is a significant overlap between these two data sets, we decide to use only the images in data set fa in this paper. The ground-truth (gender and face position) for about 90% of these images is provided as part of the 2003 Color FERET DVD[2]; the missing ground-truths were manually added by us. In this data set, the face patterns include different ethnicities (Caucasian, South Asian, East Asian, and African), facial expressions (neutral and smiling), facial makeup (with/without glasses or beard), and lighting conditions (dark and normal). Examples of male and female face patterns are shown in Fig. 6.

In our experiments, the extracted face patterns were histogram-equalized (similar lighting normalization was used in [36] and [38]), and then scaled to the range $[-1, 1]$. A fivefold cross-validation was performed on the entire data set of 1762 face patterns. For each fold, 1408 patterns were used for network design, and 354 patterns were used for network testing. The data for network design were split into a training set (90%) and a validation set (10%); the validation set was used to select the network to be evaluated on the test set; this approach was used to avoid overfitting on the training set. The final classification rates were obtained by averaging over the fivefolds.

B. PyraNet Gender Classifier

We evaluated the performance of PyraNet classifiers across different image sizes and aspect ratios. The image sizes used in the experiments are (height $\times$ width) = (18×14), (20×20), (26×22), (30×26), (30×30), (32×32), (38×32), (38×38), and (50×42). The PyraNets have two to three pyramidal layers and one output layer with a single output neuron. Using the hyperbolic tangent as activation function, the networks were trained to produce an output of 1.0 for a male face pattern and -1.0 for a female face pattern. For each network structure, two networks with different initializations were trained using the RPROP algorithm for a maximum of 2000 epochs.

The gender classification rates obtained by PyraNets are shown in Table VI. Averaged over the fivefolds, the PyraNet classifier with 853 trainable parameters and input size of 26×22 pixels has a classification rate of 96.4%. The classification rates vary from 96.1% to 96.4% for image sizes between 26×22 and 38×38 pixels. The classification rates reduce

[2]http://www.itl.nist.gov/iad/humanid/colorferet/

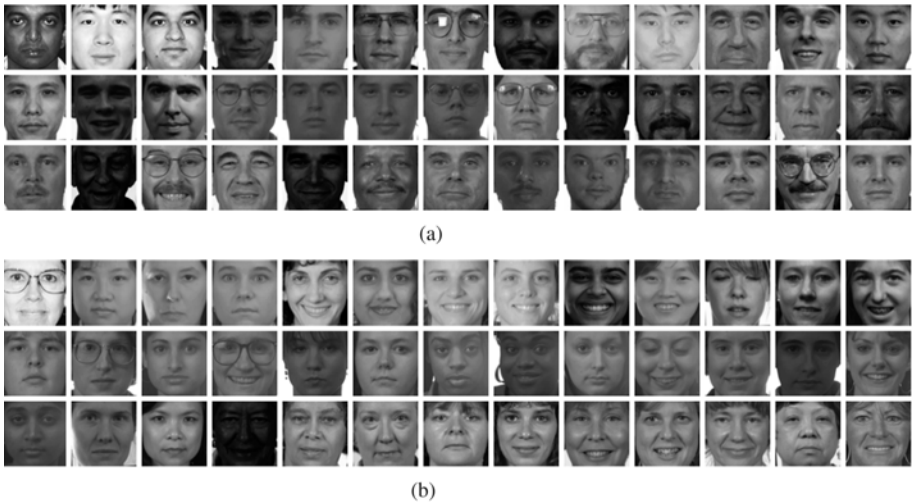

Fig. 6. Examples of face images in the FERET database. (a) Male face patterns. (b) Female face patterns.

TABLE VI
GENDER CLASSIFICATION RATES OF PYRANETS ON FERET DATA SET fa

ID	Image Size	Network Structure	Trainable Parameters	Classification Rates (%)		
				Male	Female	Overall
PyraNet1	18×14	$(r_1 = 4, o_1 = 2), (r_2 = 4, o_2 = 2)$	361	98.3	90.7	95.6
PyraNet2	20×20	$(r_1 = 5, o_1 = 2), (r_2 = 4, o_2 = 2)$	481	97.2	92.4	95.5
PyraNet3	26×22	$(r_1 = 4, o_1 = 2), (r_2 = 4, o_2 = 2)$	853	97.8	93.8	**96.4**
PyraNet4	30×26	$(r_1 = 4, o_1 = 2), (r_2 = 4, o_2 = 2)$	1177	97.6	93.8	96.3
PyraNet5	30×30	$(r_1 = 4, o_1 = 2), (r_2 = 4, o_2 = 2)$	1365	97.8	93.2	96.2
PyraNet6	32×32	$(r_1 = 5, o_1 = 2), (r_2 = 4, o_2 = 2)$	1257	97.7	93.5	96.3
PyraNet7	38×32	$(r_1 = 5, o_1 = 2), (r_2 = 4, o_2 = 2)$	1497	98.0	92.8	96.2
PyraNet8	38×38	$(r_1 = 4, o_1 = 2), (r_2 = 4, o_2 = 2)$ $(r_3 = 4, o_3 = 2)$	2239	97.6	93.5	96.1
PyraNet9	50×42	$(r_1 = 4, o_1 = 2), (r_2 = 4, o_2 = 2)$	3259	97.5	93.1	95.6

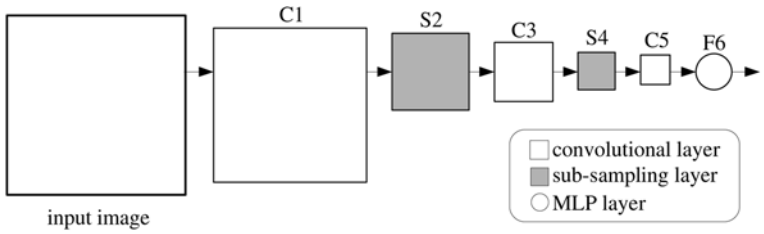

Fig. 7. Layers in the CNNs used for gender classification.

about 0.5% for smaller (18×14 pixels) and larger (50×42 pixels) image sizes.

C. Comparison With Other Gender Classifiers

For comparison purposes, we evaluated on the same data set three other classifiers: The CNN, the k-nearest neighbor (k-NN) classifier, and the SVM. The convolutional network is chosen because of its similarity with the proposed architecture. Furthermore, CNNs have been used successfully for vision tasks such as character recognition [2] and face detection [6]. The k-NN classifier is implemented to provide a comparison baseline. The SVM method [38] is considered as one of the state-of-the-art techniques in gender classification.

1) CNN Gender Classifier: Our implementation of convolutional networks is closely based on LeNet5 [2] and the network described in [6]. We investigate two image sizes: 32×32 pixels as in [2], and 36×32 pixels as in [6]. The CNNs have six layers: Three convolutional layers $C1, C3$, and $C5$, two subsampling layers $S2$ and $S4$, and one output layer $F6$ (see Fig. 7). Convolution masks with different sizes are tested for layers $C1$ and $C3$: 5×5 and 3×3 pixels. As in LeNet5, convolution masks for layer $C5$ has the same size as the feature maps of layer $S4$; layer $C5$ is, therefore, functionally equivalent to layer $N1$ in [6]. Layer $F6$ has one sigmoidal neuron, and is identical to output layer $N2$ in [6], and output layer of the above PyraNet gender classifiers. The activation function chosen for the convolutional networks is the hyperbolic tangent function.

In convolutional networks, some interlayer connections are fixed including those from the input image to layer $C1$, from layer $C5$ to output layer, and from each convolutional layer to the succeeding subsampling layer (e.g., from $C1$ to $S2$). The

TABLE VII
GENDER CLASSIFICATION RATES OF CNNS ON FERET DATA SET fa

ID	Image Size	Layer $C1$		Layer $C3$		Layer $C5$		Connection Type		Trainable Parameters	Classification Rate (%)
		FM	MS	FM	MS	FM	MS	$S2$ to $C3$	$S4$ to $C5$		
CNN1	32×32	2	5×5	4	5×5	8	5×5	partial	partial	385	85.5
CNN2	32×32	3	5×5	6	5×5	6	5×5	partial	full	1165	88.9
CNN3	32×32	3	5×5	5	5×5	7	5×5	partial	full	1239	88.6
CNN4	32×32	6	5×5	6	5×5	8	5×5	partial	full	1853	89.3
CNN5	36×32	4	5×5	14	3×3	14	7×6	partial	1-to-1	951	**89.8**
CNN6	36×32	3	5×5	6	3×3	6	7×6	full	partial	1033	88.7

Notes FM = number of feature maps, MS = convolution mask size in pixels
Architectural properties of generic convolutional neural networks
◇ Number of feature maps maps: $C1 = S2, C3 = S4$
◇ Full connections: input to $C1$, $C5$ to $F6$
◇ 1-to-1 connections: $C1$ to $S2$, $C3$ to $S4$

connection from each subsampling layer to the succeeding convolutional layer (e.g., from $S2$ to $C3$) can be set by the designer to be either a *full* or *partial* connection. We experimented with both full and partial connections, as described in [2] and [6]. Furthermore, we took some care in varying the number of feature maps in each layer and the size of convolutional masks. Hence, CNNs with different numbers of trainable parameters were considered. The training procedure for the CNNs was the same as for the PyraNets. The classification rates of the CNN gender classifiers are shown in Table VII. Among the tested convolutional networks, CNN5 with 951 trainable parameters has the highest classification rate of 89.8%. The next best network is CNN4 with 1853 trainable parameters and a classification rate of 89.3%.

2) Nearest-Neighbor Gender Classifier: The k-NN classifier stores selected samples in its training set; these samples are also called *prototypes*. During testing, the class label of a new sample is determined through majority-voting and the class labels of its k-nearest prototypes. An advantage of the k-NN classifier is that its training is fast and requires little tuning from the designer. Therefore, we use the k-NN classifier to provide a basis for comparison. The k-NN classifiers are typically used in conjunction with some feature extraction techniques. However, since the PyraNet, CNN, and SVM classifiers operate directly on image pixels, we only examine k-NN classifiers that work on image pixels. The distance function is the Euclidean distance,[3] and the image size is 32×32 pixels. We experimented with the k-NN classifiers that keep 100%, 75%, 50%, 25%, 10%, 5%, and 1% of its design set and use $k = 1, 3, 5$, and 7 nearest neighbors. Note that male and female prototypes were randomly selected, according to the ratio of male-to-female patterns in the design set.

The classification rates of the k-NN gender classifiers are shown in Table VIII. The classification rates depend significantly on the number of prototypes used. For classifier k-NN1 storing 100% of the design set, the classification rate is 92.5% $(k = 1)$. In comparison, for classifier k-NN2 storing 10% of the design set, the classification rate is only 83.6% $(k = 3)$. With a large number of prototypes, the k-NN classifier with $k = 1$ outperforms the classifiers with $k = 3$, 5, or 7.

[3]Our experiments also show that there is only a minor performance difference between different distance functions such as *Euclidean* and *negative cosine angle*.

TABLE VIII
GENDER CLASSIFICATION RATES OF k-NN CLASSIFIERS ON FERET DATA SET fa

Percentage of Design Set Used as Prototypes	Stored Parameters	Classification Rates (%)			
		$k = 1$	$k = 3$	$k = 5$	$k = 7$
100% (1408 prototypes)	1,441,792	92.5	90.7	89.9	89.3
75% (1056 prototypes)	1,081,344	89.6	88.9	87.3	87.3
50% (704 prototypes)	720,896	88.0	87.5	87.1	86.4
25% (352 prototypes)	360,448	86.0	85.7	85.0	85.1
10% (140 prototypes)	143,360	82.7	83.4	83.1	83.6
5% (70 prototypes)	71,680	80.5	81.4	81.4	81.5
1% (14 prototypes)	14,336	75.6	76.8	73.2	68.7

3) SVM Gender Classifier: Moghaddam and Yang [38] used SVM for gender classification and evaluated their classifier on a set of 1755 FERET face images (1044 male faces and 713 female faces). Note that their data set differs from the FERET data set fa or fb in terms of the number of male and female patterns. Hence, to provide a meaningful comparison with PyraNet, we evaluate the SVM gender classifier on the same fa data set used in this paper.

Our SVM implementation is based on an software package, known as library for support vector machines (LIBSVM) [42] and developed by Chang and Lin at National Taiwan University, Taipei, Taiwan. As in [38], input images are normalized for rotation, translation and lighting conditions and the kernel is the RBF. Different input image sizes were examined: 21×12 as in [38], 26×22 and 32×32. The SVM classifier has two key parameters: The penalty parameter C and the spread γ of the RBF kernel. Several values of C in the range [0.1, 100] and γ in the range [0.001, 0.1] were experimented; the SVM performance was found to degrade for C and γ values outside these ranges. The CRs obtained through a fivefold cross validation along with the C and γ parameters are shown in Table IX for different input image sizes. Classifier SVM1 based on [38] has a CR of 96.5%. Classifiers SVM2 and SVM3 for image sizes 26×22 and 32×32 have CRs of 96.0% and 96.3%. In general, the performances of the SVM classifiers are consistent with Moghaddam and Yang's results.

4) Discussion: The highest CR obtained by PyraNet gender classifiers is 96.4% (PyraNet3). In comparison, the highest CRs obtained by CNN, k-NN, and SVM gender classifiers are 89.8% (CNN5), 92.5% (k-NN1), and 96.5% (SVM1), respectively [see Fig. 8(a)]. The CR of PyraNet3 is similar (only 0.1% lower) to that of SVM1, and is significantly higher

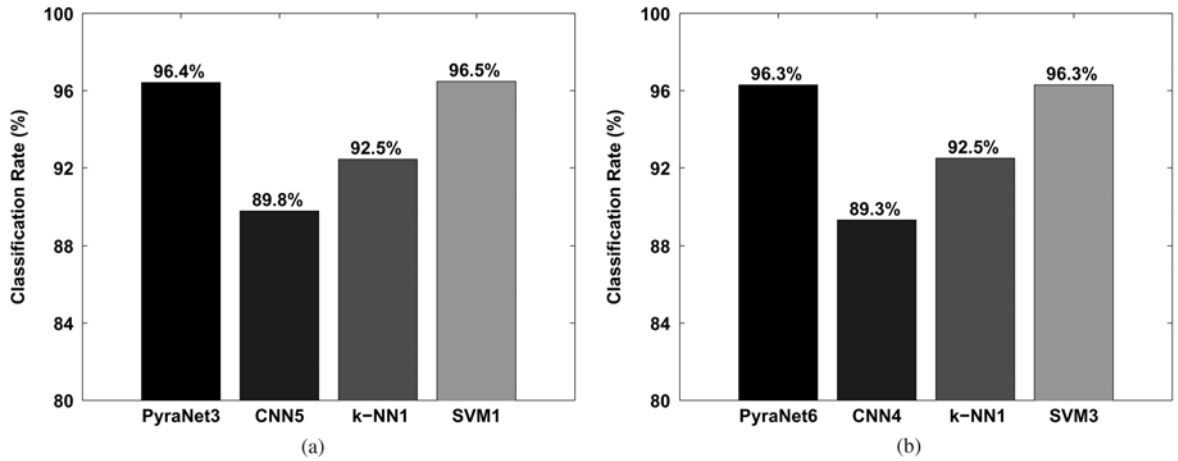

Fig. 8. Gender classification rates of PyraNet, CNN, k-NN, and SVM classifiers: (a) for classifiers with the highest classification rates and (b) for classifiers that use the same input image size of 32×32 pixels.

TABLE IX
GENDER CLASSIFICATION RATES OF SVMS ON FERET DATA SET fa

ID	Image Size	C	γ	Average Number of Support Vectors	Classification Rates (%)
SVM1	21×12	3.5	0.029	751	**96.5**
SVM2	26×22	2.0	0.009	650	96.0
SVM3	32×32	2.0	0.003	521	96.3

than that of CNN5 and k-NN1. Note that PyraNet3 has 853 trainable parameters and CNN5 has 951 trainable parameters, whereas k-NN1 has 1 441 792 stored parameters and SVM1 has on average 751 support vectors or 190 004 stored parameters. In terms of processing speed, PyraNet3 takes on average 0.31 ms to process one input image on a P4 2.8-GHz machine. It is 17.1 times faster than CNN5, 273.0 times faster than k-NN1, and 72.4 times faster than SVM1.

We also compare PyraNet, CNN, k-NN, and SVM gender classifiers that use the same input image size of 32×32 pixels. The classification rates of PyraNet6, CNN4, k-NN1, and SVM3 are 96.3%, 89.3%, 92.5%, and 96.3%, respectively [see Fig. 8(b)]. For this image size, the PyraNet has the same CR as the SVM and a significantly higher CR than the CNN and the k-NN. In terms of memory storage, PyraNet6 has 1257 parameters, CNN4 has 1853 parameters whereas k-NN1 has 1 441 792 stored parameters and SVM3 has on average 521 support vectors or 534 026 stored parameters. In terms of processing speed, PyraNet6 takes on average 0.39 ms to process one input image (32×32 pixels); it is 14.0 times faster than CNN4, 222.6 times faster than k-NN1, and 310.7 times faster than SVM3. Compared to k-NN1, SVM3 is about 1.4 times slower mainly because it involves evaluation of several RBFs.

The processing time and computational complexity of the gender classifiers depend on the input image size and the classifier structure. For the same image size, PyraNet typically requires a lower number of operations than the CNN, k-NN, and SVM because of the following reasons.

- For PyraNet, the majority of computation is for calculating the pyramidal layers. To compute a pyramidal layer where the input layer has a size of $h \times w$, the number of operations (additions, multiplications, and activation function evaluations) required is approximately $[1+(r/g)^2] \times h \times w$, where r is the receptive field's width and g is the gap factor. Typically, $(r/g)^2$ is less than 10.
- For the CNN, most computation is needed for evaluating feature maps. To compute a feature map that is connected to n previous feature maps, each having size $h \times w$, the number of operations required is approximately $n \times (2r^2 - 1) \times h \times w$, where r^2 is the number of pixels in a convolution mask. Typically, n is less than 10 and r^2 is less than 50. However, in object detection tasks where it is necessary to scan every pixel location of a large input image, the CNN has a computational advantage in that convolution can be performed on the entire input image [6].
- For the k-NN, if there are k prototypes, each is an image of size $h \times w$, the number of operations required to compute the Euclidean distances is approximately $3k \times h \times w$. Typically, k is in the order of hundreds. The amount of computation could be reduced by using other forms of the distance function.
- For the SVM with the RBF kernel, if there are k support vectors, each is an image of size $h \times w$, the number of operations required is approximately $3k \times h \times w$. Typically, k is in the order of hundreds.

In summary, our experimental results show that PyraNet gender classifiers have similar classification rates compared to the SVMs, and higher classification rates compared to the convolutional networks and k-NN classifiers. In addition, PyraNet gender classifiers use much smaller numbers of parameters compared to the SVM, and take much shorter time to process an input image compared to the SVM, k-NN, and CNN classifiers.

V. CONCLUSION

A new architecture for visual pattern recognition, called pyramidal NN, has been presented. The new NN processes image pixels directly, and has 2-D layers organized in a pyramidal structure similar to the traditional image pyramids. Feature extraction at pyramidal layers is determined entirely through training. In PyraNet, overlapping 2-D neurons are trained to extract image features that have strong spatial dependency. In

this paper, we have developed five generic training algorithms for PyraNet that use two types of error functions, namely mse and CE. Our analysis of the five training algorithms reveals that the RPROP and CG algorithms have reasonable convergence speeds and require small memory storage, whereas the LM algorithm is fast but requires significantly more memory. We also show that PyraNet can be trained using either the mse or CE error functions, and there is no significant difference in performance between the two. We have applied the new pyramidal NN to the task of gender classification of facial images. Evaluated on the FERET data set fa, PyraNet achieves a classification rate of 96.4%. The performance of the PyraNet gender classifier is significantly better compared to the CNN and the k-NN classifiers, and similar to the SVM classifier. Although sigmoidal neurons have been used in this paper, other types of neurons such as RBF neurons can also be used with the PyraNet architecture.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their comments and valuable suggestions. Portions of the research in this paper use the color FERET database of facial images collected under the FERET program.

REFERENCES

[1] K. Fukushima and N. Wake, "Handwritten alphanumeric character recognition by the neocognitron," *IEEE Trans. Neural Netw.*, vol. 2, no. 3, pp. 355–365, Mar. 1991.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[3] M. D. Ganis, C. L. Wilson, and J. L. Blue, "Neural network-based systems for handprint OCR applications," *IEEE Trans. Image Process.*, vol. 7, no. 8, pp. 1097–1112, Aug. 1998.

[4] S. Yamaguchi and H. Itakura, "A car detection system using the neocognitron," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 1991, vol. 2, pp. 1208–1213.

[5] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 1, pp. 23–38, Jan. 1998.

[6] C. Garcia and M. Delakis, "Convolutional face finder: A neural architecture for fast and robust face detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 11, pp. 1408–1423, Nov. 2004.

[7] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE Trans. Neural Netw.*, vol. 8, no. 1, pp. 98–113, Jan. 1997.

[8] M. J. Er, W. Chen, and S. Wu, "High-speed face recognition based on discrete cosine transform and RBF neural networks," *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 679–691, May 2005.

[9] M. Rosenblum, Y. Yacoob, and L. S. Davis, "Human expression recognition from motion using a radial basis function network architecture," *IEEE Trans. Neural Netw.*, vol. 7, no. 5, pp. 1121–1138, Sep. 1996.

[10] K. Fukushima, "Neocognitron: A hierarchical neural network capable of visual pattern recognition," *Neural Netw.*, vol. 1, no. 2, pp. 119–130, 1988.

[11] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture," *J. Physiol.—London*, vol. 160, no. 1, pp. 106–154, 1962.

[12] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, 1989.

[13] R. C. Gonzalez and R. E. Woods, *Digital Image Pprocessing*. Englewood Cliffs, NJ: Prentice-Hall, 2002.

[14] F. Tivive and A. Bouzerdoum, "Efficient training algorithms for a class of shunting inhibitory convolutional neural networks," *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 541–556, May 2005.

[15] M. Hoshino and J. Chao, "On representation and generalization capability of pyramid neural networks," in *Proc. Int. Joint Conf. Neural Netw.*, 2002, vol. 2, pp. 1166–1171.

[16] V. Cantoni and A. Petrosino, "Neural recognition in a pyramidal structure," *IEEE Trans. Neural Netw.*, vol. 13, no. 2, pp. 472–480, Mar. 2002.

[17] C. M. Bishop, *Neural Networks For Pattern Recognition*. Oxford, U.K.: Clarendon, 1996.

[18] M. C. Joshi and K. M. Moudgalya, *Optimization Theory and Practice*. Harrow, U.K.: Alpha Science International Ltd., 2004.

[19] J. S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," in *Neurocomputing: Algorithms, Architectures and Applications*, F. Souli and J. Hrault, Eds. New York: Springer-Verlag, 1990, pp. 227–236.

[20] E. K. P. Chong and S. H. Zak, *An Introduction to Optimization*. New York: Wiley, 1996.

[21] S.-C. Ng, C.-C. Cheung, and S.-H. Leung, "Magnified gradient function with deterministic weight modification in adaptive learning," *IEEE Trans. Neural Netw.*, vol. 15, no. 6, pp. 1411–1423, Nov. 2004.

[22] N. Zhang, W. Wu, and G. Zheng, "Convergence of gradient method with momentum for two-layer feedforward neural networks," *IEEE Trans. Neural Netw.*, vol. 17, no. 2, pp. 522–525, Mar. 2006.

[23] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Cambridge, MA: Bradford Books, 1986, vol. I, pp. 318–362.

[24] M. T. Hagan, H. B. Demuth, and M. H. Beale, *Neural Network Design*. Boston, MA: PWS-Kent, 1996.

[25] M. Riedmiller and H. Braun, "A direct adaptive method of faster backpropagation learning: The RPROP algorithm," in *Proc. IEEE Int. Conf. Neural Netw.*, San Francisco, CA, 1993, pp. 586–591.

[26] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE Trans. Neural Netw.*, vol. 5, no. 6, pp. 989–993, Nov. 1994.

[27] C. Charalambous, "A conjugate gradient algorithm for the efficient training of artificial neural networks," *IEE Proc. Part G*, vol. 139, no. 3, pp. 301–310, 1992.

[28] M.-H. Yang, D. Kriegman, and N. Ahuja, "Detecting faces in images: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 1, pp. 34–58, Jan. 2002.

[29] S. L. Phung, A. Bouzerdoum, and D. Chai, "Skin segmentation using color pixel classification: Analysis and comparison," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 1, pp. 148–154, Jan. 2005.

[30] K. K. Sung and T. Poggio, "Example-based learning for view-based human face detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 1, pp. 39–51, Jan. 1998.

[31] M.-H. Yang, D. Roth, and N. Ahuja, "A SNoW-based face detector," in *Advances in Neural Information Processing Systems 12*. Cambridge, MA: MIT Press, 2000, pp. 855–861.

[32] S. Z. Li and A. K. Jain, *Handbook of Face Recognition*. New York: Springer-Verlag, 2005.

[33] K. Ueki, H. Komatsu, S. Imaizumi, K. Kaneko, N. Sekine, J. Katto, and T. Kobayashi, "A method of gender classification by integrating facial, hairstyle, and clothing images," in *Proc. Int. Conf. Pattern Recognit.*, 2004, vol. 4, pp. 446–449.

[34] B. Golomb, D. T. Lawrence, and T. J. Sejnowski, "Sexnet: A neural network that identifies sex from human faces," *Adv. Neural Inf. Process. Syst.*, vol. 3, pp. 572–577, 1991.

[35] M. Gray, D. T. Lawrence, B. A. Golomb, and T. J. Sejnowski, "A perception revealing the face of sex," *Neural Comput.*, vol. 7, no. 6, pp. 1160–1164, 1995.

[36] S. Gutta, J. R. J. Huang, P. Jonathon, and H. Wechsler, "Mixture of experts for classification of gender, ethnic origin, and pose of human faces," *IEEE Trans. Neural Netw.*, vol. 11, no. 4, pp. 948–960, Jul. 2000.

[37] A. Jain and J. Huang, "Integrating independent components and linear discriminant analysis for gender classification," *Proc. IEEE Int. Conf. Autom. Face Gesture Recognit.*, pp. 159–163, 2004.

[38] B. Moghaddam and M.-H. Yang, "Learning gender with support faces," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 707–711, May 2002.

[39] B. Wu, H. AI, and C. Huang, "Real-time gender classification," in *Proc. SPIE Multi-Spectral Image Process. Pattern Recognit.*, Beijing, China, 2003, vol. 5286, pp. 498–503.

[40] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. Comput. Vis. Pattern Recognit.*, Kauai, Hawaii, 2001, pp. 511–518.

[41] P. J. Phillips, H. Moon, S. A. Rizvi, and P. J. Rauss, "The FERET evaluation methodology for face-recognition algorithms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 10, pp. 1090–1104, Oct. 2000.
[42] 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

Son Lam Phung (S'02–M'03) received the B.Eng. degree with first-class honors in 1999 and the Ph.D. degree in 2003, both in computer engineering, from Edith Cowan University, Perth, Australia.

He joined the University of Wollongong, Wollongong, Australia, as a Research Fellow in 2005, and he is currently a Lecturer in the School of Electrical, Computer and Telecommunications Engineering. His general research interests are in the areas of image and video processing, neural networks (NNs), pattern recognition, and machine learning.

Dr. Phung received several awards including the University and Faculty Medals in 2000 as the graduating student with the highest course average.

Abdesselam Bouzerdoum (M'89–SM'03) received the M.Sc. and Ph.D. degrees in electrical engineering from the University of Washington, Seattle, in 1986 and 1991, respectively.

He joined the University of Adelaide, Australia, in 1991. In 1998, he was appointed the Associate Professor at Edith Cowan University, Perth, Australia. Since 2004, he has been with the University of Wollongong, Wollongong, Australia, where he is currently a Professor of Computer Engineering and Head of the School of Electrical, Computer and Telecommunications Engineering. He was a Visiting Professor at Institut Galilée, University of Paris-13, France, in 2004 and 2005. He has published over 200 technical articles and graduated fifteen Ph.D. and six Research Masters students. His research interests include signal/image processing, machine learning, pattern recognition, and very large scale integration (VLSI) implementation of smart vision microsensors.

Dr. Bouzerdoum has received several fellowships and distinguished awards; among them are the Vice Chancellor's Distinguished Researcher Award in 1998 and 1999, Awards for Excellence in Research Leadership and Excellence in Postgraduate Supervision, and the Chester Sall Award for best paper in IEEE TRANSACTIONS ON CONSUMER ELECTRONICS. In 2001, he was awarded a Distinguished Researcher (Chercheur de Haut Niveau) Fellowship from the French Ministry of Research. He served as a Chair of the IEEE Washington Section Signal Processing Chapter in 2004, and he was a Chair of the IEEE SA Section NN Region Interest Group (RIG) from 1995 to 1997. Since 1999, he has been serving as an Associate Editor of IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS.