Supervised Texture Classification Using a Probabilistic Neural Network and Constraint Satisfaction Model

P. P. Raghu and B. Yegnanarayana, *Senior Member, IEEE*

***Abstract*— In this paper, the texture classification problem is projected as a constraint satisfaction problem. The focus is on the use of a probabilistic neural network (PNN) for representing the distribution of feature vectors of each texture class in order to generate a feature-label interaction constraint. This distribution of features for each class is assumed as a Gaussian mixture model. The feature-label interactions and a set of label-label interactions are represented on a constraint satisfaction neural network. A stochastic relaxation strategy is used to obtain an optimal classification of textures in an image. The advantage of this approach is that all classes in an image are determined simultaneously, similar to human perception of textures in an image.**

***Index Terms*— Constraint satisfaction, feedback neural network, Gabor filters, Gaussian mixture model, probabilistic neural network, self-organizing map, texture classification.**

I. Introduction

In the domain of image analysis, texture-based segmentation and classification of natural scenes are much complicated compared to the approaches based on pixel intensities [5]. A texture classification problem can be viewed as either feature-specific or domain-specific. A feature-specific texture classifier assigns a class label to each pixel based on the features corresponding to the pixel, independent of any domain knowledge. Examples of feature-specific texture classification schemes can be found in [7] and [12].

On the other hand, a domain-specific classifier uses domain knowledge in the form of additional constraints to achieve classification. The performance of any domain-specific approach is expected to be superior to the feature-specific approaches. Like any human decision making, an image classification problem also can be posed as one which requires simultaneous satisfaction of many constraints. One can model domain-specific knowledge as a set of constraints on the image features and on the possible labels for each pixel. A suitable constraint satisfaction model may then be used to attain a state such that the constraints are satisfied to a maximum extent. An example of the constraint satisfaction scheme is a rule-based expert system where the constraints are described as rules in its knowledge base [19].

Manuscript received February 2, 1997; revised November 27, 1997.

P. P. Raghu is with the LG Software Development Center, Bangalore, India 560 001.

B. Yegnanarayana is with the Department of Computer Science and Engineering, Indian Institute of Technology, Madras, India 600 036.

Publisher Item Identifier S 1045-9227(98)03688-1.

In most of the image processing situations, the use of rule-based approach is difficult because of the following problems.

1) Most rule-based systems are designed to deal with symbolic logic and reasoning, whereas image features are numerical and fuzzy in nature. It is difficult to translate these numerical features into a set of crisp rules.
2) It is difficult to describe symbolically the knowledge used by a human interpreter in analyzing a given image.
3) Constraints modeled in rule-based system are generally hard constraints (which *must be* satisfied), whereas many real-world constraints are weak constraints [1] which *ought to be* satisfied to attain an acceptable solution.
4) Rule-based systems are rigid and deterministic which do not allow pattern variability in terms of exceptions and randomness. Generally, domain-specific constraints are ambiguous, and some times conflicting, but useful enough to decide the outcome of the processing.

The human reasoning process is superior to a rule-based system in these situations because our brain uses a computational architecture with several neurons working in parallel, thus representing a large number of loosely bound constraints. We also deal well with ambiguity in problems and usually have little difficulty in correctly determining the missing information. In this respect, biologically inspired artificial neural-network models provide greater flexibility than the rule-based approach as a constraint satisfaction mechanism to handle constraints that ought to be satisfied rather than satisfying all the specified constraints.

Thus it is useful to conceptualize an artificial neural-network system as a constraint satisfaction model where each node represents a hypothesis and connection between two nodes represents the constraint between corresponding hypotheses [16]. The importance and nature of the constraint are decided by the numerical value of the connection weight. A node may have an external input and a bias providing *a priori* information regarding the truth value of the corresponding hypothesis. The process of iteratively seeking a solution which satisfies a large number of weak constraints is called a *relaxation* strategy.

Constraint satisfaction neural-network models were developed for classification of textured images using Markov random field (MRF) models based on the modeling in the pixel intensity domain [3]. But for images which are of texture-type, proper representation of texture information in terms of textural features becomes crucial [5]. In this scenario, it is useful to combine the textural features as well as the constraint satisfaction neural-network models to achieve superior classification performance.

The work reported in [13] uses this domain-specific approach for texture classification. The texture classification was considered as a constraint satisfaction problem consisting of image-specific constraints represented by three random processes, namely *feature formation process, partition process*, and *label competition process*. The feature formation process was derived using a statistical model for the textural features extracted from a bank of Gabor filters [2]. The constraints were represented on a Hopfield network [6], and a deterministic relaxation strategy was used to attain an optimal classification of an image.

In this paper, we show that a PNN representation of feature vectors produces a better representation of the feature formation process. Conventional PNN involves defining the Gaussians at every training pattern for each class. We propose a modification in the PNN for better representation of the feature formation process. The main objective of this paper is to show the significance of this modification of PNN on classification of textured images.

Section II gives a brief description of the constraint satisfaction neural-network model presented in [13]. The feature modeling using PNN is described in Section III. The section also presents our proposed modification of PNN. In Section IV, we provide a set of experimental results to show the efficacy of the proposed methods.

II. A Constraint Satisfaction Neural-Network Model for Texture Classification

This section reviews the constraint satisfaction neural-network model proposed in [13]. Consider a textured image T_Ω designated by a domain $\Omega = \{(i, j), 0 \le i < I, 0 \le j < J\}$ of pixel positions. Let $\{g_s \in \mathrm{R}^M, \forall s \in \Omega\}$ be a set of M-dimensional feature vectors used to characterize the image, where each g_s is an M-dimensional feature vector characterizing the pixel s in Ω. The g_s can be considered as the realization of an M-dimensional random process G_s. Let L_s denote the random variable describing the texture label of the pixel s. We assume that L_s can take any value from the set of labels $\{0, 1, \cdots, K-1\}$ where K is the number of texture classes. The corresponding texture classes are denoted by $C_0 \cdots C_{K-1}$. Also, let Ω_k, a subset of Ω, be the training site for the class C_k. The notation $\mathtt{card}(\Omega)$ is used to denote the cardinality of any set Ω. Let N_s^p be a set of pixels in the pth order symmetric neighborhood of the pixel s.

A. Feature Formation Process

The feature formation process formulates the probability of feature vector of a pixel s given the model parameters of each class k, and this is given by

$$P(G_s = g_s | L_s = k) = \frac{e^{-E_f(G_s = g_s | L_s = k)}}{Z_f} \tag{1}$$

where E_f is an energy function defined by the selected statistical model and Z_f is a normalization constant considered independent of s and k.

B. Partition Process

The label of any pixel in an image depends on the labels of the pixels in its neighborhood. The partition process $P(L_s | L_r, \forall r \in N_s^p)$ describes probability of the label of each pixel s given the labels of the pixels in a uniform pth order neighborhood N_s^p of s [13]

$$P(L_s | L_r, \forall r \in N_s^p) = \frac{\exp\left[\sum_{\forall r \in N_s^p} \beta\delta(L_s - L_r)\right]}{Z_p} \tag{2}$$

where β is a positive constant, $\delta(.)$ is the Kronecker delta function and Z_p is a normalization constant.

C. Label Competition Process

The label competition process tries to reduce the probability of having another label when the pixel is already labeled. It is defined by the conditional probability of assigning a new label to an already labeled pixel, and is expressed as [13]

$$P(L_s = k | \tilde{L}_s) = \frac{\exp\left[-\alpha \sum_{l \in \tilde{L}_s} \bar{\delta}(k - l)\right]}{Z_c} \tag{3}$$

where $\tilde{L}_s$ denotes the set of labels that may be assigned to the pixel s and Z_c is the normalization constant.

D. Neural-Network Representation of Constraints

Maximization of the *a posteriori* probability $P(L_s = k | G_s, L_r, \forall r \in N_s^p, \tilde{L}_s)$ will provide the optimal classification of the given image. This probability describes the label L_s of the pixel s given the feature measurement G_s of s, the labels of the neighborhood pixels and the possible labels previously assigned to s. Using Bayes theorem, we can write

$$\begin{aligned} &P(L_s = k | G_s, L_r, \forall r \in N_s^p, \tilde{L}_s) \\ &\quad = \frac{P(G_s | L_s = k) P(L_s = k | L_r, \forall r \in N_s^p) P(L_s = k | \tilde{L}_s)}{P(G_s) P(L_s = k)}. \end{aligned} \tag{4}$$

Expressing this as the following Gibbs distribution:

$$\begin{aligned} &P(L_s = k | G_s, L_r, \forall r \in N_s^p, \tilde{L}_s) \\ &\quad = \frac{e^{-E(L_s = k | G_s, L_r, \forall r \in N_s^p, \tilde{L}_s)}}{Z} \end{aligned} \tag{5}$$

and by substituting (1)–(3) in (4), we get the Gibb's energy as

$$\begin{aligned} &E(L_s = k | G_s, L_r, \forall r \in N_s^p, \tilde{L}_s) \\ &\quad = E_f(G_s | L_s = k) - \sum_{\forall r \in N_s^p} \beta\delta(k - L_r) \\ &\qquad + \sum_{l \in \tilde{L}_s} \alpha\bar{\delta}(k - l) \end{aligned} \tag{6}$$

and the normalization constant $Z = Z_f Z_p Z_c P(G_s) P(L_s = k)$. The energy function in (6) summed over all pixels and

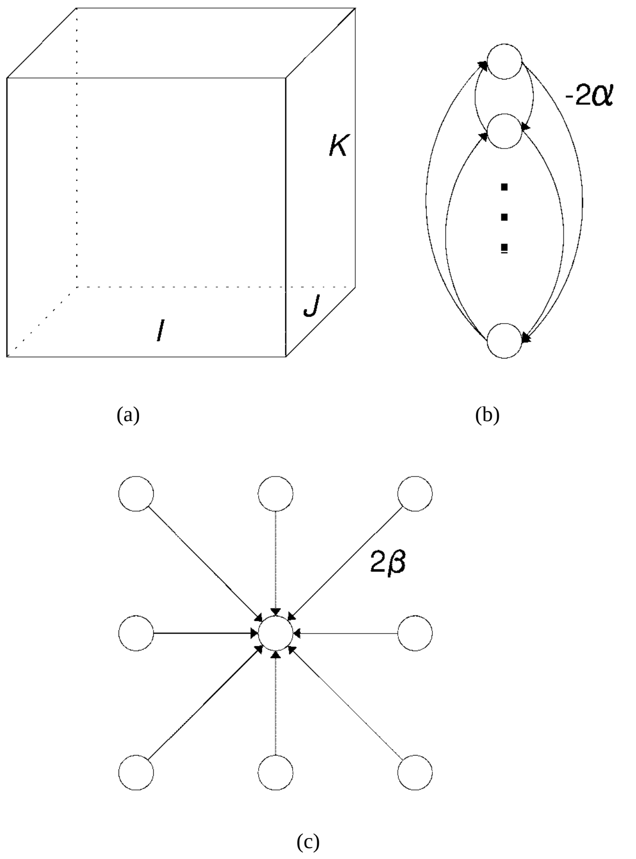

Fig. 1. Structure of the 3-D feedback network: (a) 3-D lattice of size $I \times J \times K$. (b) Connections among nodes in the label column of each pixel. Each connection is of strength -2α. (c) Connections from a set of neighboring nodes to each node in a label layer. Each connection has a strength 2β.

all possible labels will give the total energy E^{total} of the classification model

$$E^{\text{total}} = \sum_{s,\,k} E(L_s = k | G_s,\, L_r,\, \forall r \in N_s^p,\, \tilde{L}_s). \tag{7}$$

The energy function in (7) can be represented on a feedback network with nodes arranged in three-dimensional (3-D) lattice of size $I \times J \times K$, for convenience, as shown in Fig. 1. For any node $(i,\, j,\, k)$, $(i,\, j) = s$ corresponds to the pixel position and k denotes the label index for that pixel. Each node $(i,\, j,\, k)$ has a bias $B_{i,\,j,\,k}$. Also, each node $(i,\, j,\, k)$ is connected to any other node $(i_1,\, j_1,\, k_1)$ by means of a connection weight $W_{i,\,j,\,k;i_1,\,j_1,\,k_1}$. Comparing (7) with the energy function of the Hopfield network [6], one can determine the bias $B_{i,\,j,\,k}$ and the weight $W_{i,\,j,\,k;i_1,\,j_1,\,k_1}$ in a similar manner to [13] as

$$B_{i,\,j,\,k} = -E_f[G_{(i,\,j)} = g_{(i,\,j)} | L_{(i,\,j)} = k] \tag{8}$$

and

$$W_{i,\,j,\,k;i_1,\,j_1,\,k_1} = \begin{cases} 2\beta, & \text{if } (i_1,\, j_1) \in N^p_{(i,\,j)} \text{ and } k = k_1 \\ -2\alpha, & \text{if } (i_1,\, j_1) = (i,\, j) \text{ and } k \neq k_1 \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

Here, the weights are symmetric and there is no self-loop.

Let $\Lambda_{i,\,j,\,k} \in \{0,\, 1\}$ be the output of node $(i,\, j,\, k)$. Estimation of the network state configuration $\{\Lambda_{i,\,j,\,k}\}$ for all pixels $(i,\, j)$ which minimizes the Gibb's energy in (7) will yield the *maximum a posteriori* (MAP) estimate of the classification of the image. For the experimental results presented in this paper, a stochastic relaxation procedure based on simulated annealing [8] extended to this 3-D neural network [14] is used to obtain a global (or near-global) minimum energy state. The relaxation procedure derives this stable state by simultaneously satisfying to a maximum extent the possible constraints [defined by energy function in (7)] with respect to all pixels and all possible labels in the given image. Thus the neural-network model views the entire image in order to classify it, and hence brings in the global image knowledge for making decision at the local level.

III. Feature Formation Process: Feature Modeling Using PNN

Generally, one can use a standard statistical distribution to define the feature formation process. In [13], we have used a Gaussian distribution for defining the feature distribution of each class, assuming that the feature vectors of each class form a hypersphere in the M-dimensional feature space. Alternatively, a multivariate Gaussian distribution, which assumes an M-dimensional ellipsoidal feature distribution for each class, may be a better feature model [14].

A natural image data does not fit into any of the standard statistical distributions, and it is difficult to determine the underlying distribution. But the success of any analysis lies in our ability to approximate the true distribution. Gaussian and multivariate Gaussian distributions fail to capture the details of a class having number of distinct clusters in the feature space. Further, dispersion of the feature vectors for a given class may be large and of any arbitrary shape. In such cases, a Gaussian mixture model seems to be an appropriate choice.

Determination of the mixture model consists of estimating the parameters and the weight of each component. The expectation-maximization (EM) algorithm has been widely used to iteratively compute the maximum-likelihood estimates of the parameters, considering the observed data as incomplete data [4]. This general approach is very complex and requires large computation time. Alternatively, methods based on artificial neural networks are useful for the mixture modeling of the feature vectors. It is possible to capture this model using a PNN [18].

A. Feature Modeling: Gaussian Mixture Model Using PNN

The PNN model [18] is based on Parzen's results on PDF estimators [11]. PNN is a three-layer feedforward network consisting of input layer, a pattern layer, and a summation layer as shown in Fig. 2. The input layer contains M nodes to accept an M-dimensional feature vector. The pattern layer consists of K pools of pattern nodes. The kth pool in the pattern layer contains S_k number of pattern nodes, where $S_k = \texttt{card}(\Omega_k)$. Each node in the pattern layer is connected from every node in the input layer. The summation layer consists of K nodes, one node for each pool in the pattern layer. Pattern nodes of each kth pool in the pattern layer is connected to the corresponding kth summation node in

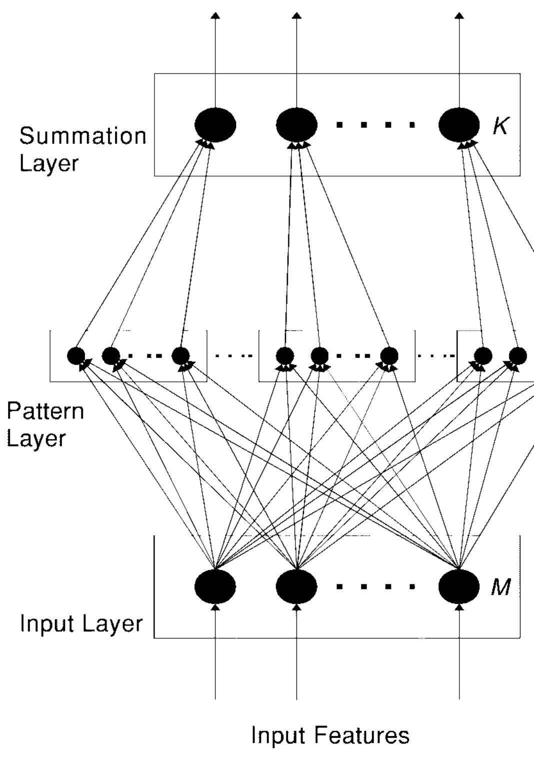

Fig. 2. Structure of the probabilistic neural network (PNN).

the summation layer. A radial basis function and a Gaussian activation function are used for the pattern nodes. For the summation nodes, a linear basis function and a linear activation function are used.

In the network, each training vector $g_m (0 \le m < S_k)$ of the class C_k is stored as the weight $W_{m,k}$ connecting the input layer and the mth pattern node in the kth pool of pattern layer. The connection weight from each pattern node in the kth pool and the summation node for the kth class is assigned as $1/S_k$. Note that the training is a one-pass algorithm, and hence it is trivial in this case.

For any input vector $g_s, s \in \Omega$, the output of the mth pattern node belonging to the kth pool is $e^{-\|g_s - W_{m,k}\|^2 / 2\sigma^2_{m,k}} / \sqrt{(2\pi)^M \sigma^2_{m,k}}$, where $\sigma_{m,k}$ is a smoothing parameter for the Gaussian activation function of that node. Output of the kth summation node gives the probability of the feature vector g_s of the pixel s, given the label of that pixel is k. Now the feature formation process in (1) is described as

$$\begin{aligned} P(G_s = g_s | L_s = k) &= \frac{1}{S_k} \sum_{m=1}^{S_k} \frac{e^{-\|g_s - W_{m,k}\|^2 / 2\sigma^2_{m,k}}}{\sqrt{(2\pi)^M \sigma^2_{m,k}}} \\ &= \exp\left\{ \ln \left[\frac{1}{S_k} \sum_{m=1}^{S_k} \frac{e^{-\|g_s - W_{m,k}\|^2 / 2\sigma^2_{m,k}}}{\sqrt{(2\pi)^M \sigma^2_{m,k}}} \right] \right\}. \end{aligned} \quad (10)$$

This relation shows that the PNN represents the input feature subspace of each class C_k as a Gaussian mixture distribution, where there are as many components as there are training patterns for that class. Each component m of the mixture for class C_k is represented by the mth training vector g_m as the mean and a constant $\sigma^2_{m,k}$ as the variance.

The complexity in the computation of $P(G_s|L_s)$ in (10) depends upon the number of classes and the number of training patterns per class. If there are a large number of training patterns, the number of nodes in the pattern layer $(= \sum_{k=0}^{K-1} S_k)$ becomes very large. This in turn increases the time for estimating the probability of all pixels in the image using (10). Also, the accuracy of the mixture model is decided by the smoothing parameters $\sigma_{m,k}$. There are only *ad hoc* methods to estimate $\sigma_{m,k}$ from the given data [17]. In order to deal with these problems, we propose a modified architecture for the PNN.

B. Feature Modeling: Modified PNN Using Clustering of Training Data

The proposed modification of PNN consists of grouping the training patterns in each class. The grouping is done using the principle of vector quantization. Consider the feature subspace for the class C_k consisting of S_k number of training patterns. A vector quantizer is used to partition this feature subspace into Q_k partition regions $R_{m,k}$, $0 \le m < Q_k - 1$. The assumption here is that $Q_k \ll S_k$. Each partition $R_{m,k}$ is associated with a subset $\Omega_{m,k}$ of Ω_k so that $\bigcup_{m=0}^{Q_k-1} \Omega_{m,k} = \Omega_k$ and $\Omega_{m,k} \bigcap \Omega_{q,k} = \emptyset$ if $m \ne q$.

For each class C_k, we first obtain a feature map using the Kohonen's self-organizing map (SOM) learning [9], [15]. We use the neighborhood characteristics of the feature map to perform the vector quantization effectively. Consider a SOM (for a class C_k) with M input nodes and an output layer arranged as a two-dimensional (2-D) lattice of nodes. Let O be the set of SOM output nodes and let us assume that the number of components Q_k in the mixture for the class C_k is less than or equal to $\mathtt{card}(O)$. Training of SOM is performed so that the network reaches an optimal state corresponds to a minimum of the distance measure $\sum_{m \in O} \sum_{s \in \Omega_k} \|g_s - W_{m,k}\|^2$, where $W_{m,k}$ is the M-dimensional weight vector connecting the input nodes and the node m in the output layer of SOM. After training, each node m in the output layer of SOM represents one partition region $R_{m,k}$. The set $\Omega_{m,k}$ of training site pixels is uniquely assigned to the node m so that every feature vector of the pixels in $\Omega_{m,k}$ makes the node m to win.

For a pixel s and label k, the feature formation process is expressed as a multivariate Gaussian mixture given by

$$P(G_s = g_s | L_s = k) = \frac{1}{Q_k} \sum_{m=0}^{Q_k-1} \frac{1}{\sqrt{(2\pi)^M |\Sigma_{m,k}|}} \cdot e^{-(1/2)(g_s - \mu_{m,k})^t \Sigma^{-1}_{m,k} (g_s - \mu_{m,k})} \quad (11)$$

where t is the transpose operation. In this, $\mu_{m,k}$ and $\Sigma_{m,k}$ are the mean and the covariance matrix, respectively, of the mth component in the mixture for the class C_k.

The model parameters $\mu_{m,k}$ and $\Sigma_{m,k}$ for each component m in each class C_k are estimated from the training site partition

$R_{m,k}$ of that class as

$$\text{Mean:}\quad \mu_{m,k} = \frac{1}{S_{m,k}} \sum_{s \in \Omega_{m,k}} g_s \tag{12}$$

Covariance matrix:

$$\Sigma_{m,k} = \begin{bmatrix} \nu_{11,m,k} & \nu_{12,m,k} & \cdots & \nu_{1M,m,k} \\ \nu_{21,m,k} & \nu_{22,m,k} & \cdots & \nu_{2M,m,k} \\ \cdots & \cdots & \cdots & \cdots \\ \nu_{M1,m,k} & \nu_{M2,m,k} & \cdots & \nu_{MM,m,k} \end{bmatrix} \tag{13}$$

where each element $\nu_{ij,m,k}$ is the covariance of each component $g_s(i)$ and $g_s(j)$ of $g_s \in \Omega_{m,k}$. It is estimated as

$$\nu_{ij,m,k} = \frac{1}{S_{m,k}} \sum_{s \in \Omega_{m,k}} [g_s(i) - \mu_{m,k}(i)][g_s(j) - \mu_{m,k}(j)] \tag{14}$$

where $S_{m,k} = \mathtt{card}(\Omega_{m,k})$ is the number of training patterns belonging to the partition $R_{m,k}$.

The partitioning of Ω_k into $\Omega_{m,k}$ is such that $0 \leq S_{m,k} \leq S_k$. This means that, 1) a node m may not be winning for any training vector in Ω_k; 2) it may win for only a subset of training vectors in the training site of class k; or 3) it may win for every training vector in the training site. If $S_{m,k} = 0$, we simply omit that node from the parameter estimation. However, when $S_{m,k} = 1$ for any node m, the covariance matrix $\Sigma_{m,k}$ for that component will become meaningless and there is a chance that $\Sigma_{m,k}$ becomes singular. In order to avoid this, we take advantage of the feature map characteristics to determine the node adjacent to this isolated vector. For an output node m such that $S_{m,k} = 1$, a node n is selected such that $\|W_{n,k} - W_{m,k}\| < \|W_{q,k} - W_{m,k}\|_{\forall q \in O - \{m\}}$, and the feature vector $g_s, s \in \Omega_{m,k}$ is reassigned to the node n, updating the partitions as, $\Omega_{n,k} \leftarrow \Omega_{n,k} \bigcup \Omega_{m,k}$ and $\Omega_{m,k} = \emptyset$. Then, the mean and covariance matrix of the component n is calculated again based on the new partition.

If we compare (10) and (11) with the general expression for feature formation process in (1), the values of bias $B_{i,j,k}$ can be determined for the cases where conventional and modified PNN are used for feature modeling. This is given by the following cases.

Case 1: Feature formation process with conventional PNN model

$$B_{i,j,k} = \ln\left[\frac{1}{S_k} \sum_{m=1}^{S_k} \frac{e^{-\|g_s - W_{m,k}\|^2 / 2\sigma_{m,k}^2}}{\sqrt{(2\pi)^M \sigma_{m,k}^2}}\right]. \tag{15}$$

Case 2: Feature formation process with modified PNN model

$$B_{i,j,k} = \ln\left[\frac{1}{Q_k} \sum_{m=0}^{Q_k - 1} \frac{1}{\sqrt{(2\pi)^M |\Sigma_{m,k}|}} \cdot e^{-(1/2)(g_s - \mu_{m,k})^t \Sigma_{m,k}^{-1} (g_s - \mu_{m,k})}\right]. \tag{16}$$

IV. Results and Discussion

We compare the performance of the proposed method with a scheme which uses a multivariate Gaussian distribution for the feature formation process, in which case the bias of the network takes the following form:

$$B_{i,j,k} = -\tfrac{1}{2}\{(g_s - \mu_k)^t \Sigma_k^{-1} (g_s - \mu_k) + \ln[(2\pi)^M |\Sigma_k|]\}. \tag{17}$$

We also compare the conventional and modified PNN models on the basis of the accuracy of classification results and the time taken for bias estimation. For the experiments with conventional PNN, we have used a constant value of $\sigma_{m,k}$ for all components m and for all classes C_k.

Gabor wavelets expressed as [10]

$$f(x, y, a, \omega, \theta, \sigma) = e^{-(a^2/2\sigma^2)(x^2+y^2) + ja\omega(x \cos\theta + y \sin\theta)} \tag{18}$$

are used for texture feature extraction. Here, θ, ω, and σ are the orientation, radial frequency, and the bandwidth of the Gabor filter. a is the wavelet scale factor chosen as $a = 2^\gamma$, where γ is an integer.

An image shown in Fig. 3(a) containing five Brodatz texture tiles (raffia D18—left upper, brick wall D95—right upper, beans—left bottom, straw—right bottom, and burlap—center) was used for the experiments. The Gabor wavelet bank used for extracting features from this image consists of eight filters derived from a mother wavelet of $\sigma = 25$, $\omega = 0.2$ using two scales ($\gamma = 3$ and 4, with wavelengths of 1.25π and 0.625π pixels/cycle) and four rotations ($\theta = 0, 45, 90$, and $135°$). This results in an eight-dimensional feature vector for each pixel. The training site per class contains 1000 pixels, and hence the total number of nodes in the pattern layer for the conventional PNN becomes 5000. The classification result when the multivariate Gaussian was used for feature formation process is shown in Fig. 3(b). The results with the conventional PNN for feature formation process are shown in Fig. 3(c) and 3(d) for $\sigma_{m,k} = 0.001$ and $\sigma_{m,k} = 0.01$, respectively. Fig. 3(e) shows the result when the modified PNN was used for the Gaussian mixture modeling of the feature vectors. A third-order ($p = 3$) partition process was used in these experiments.

An important observation is the dependency of the smoothing parameters of the conventional PNN on the classification accuracy. This is obvious if we compare the results in Fig. 3(c) and 3(d). Another observation is that the Gaussian mixture distributions implemented using the conventional and modified PNN models [Fig. 3(c) and 3(e)] perform much better than the multivariate Gaussian distribution [Fig. 3(b)]. The multivariate Gaussian is not able to capture well the distribution of the texture features compared to the Gaussian mixture models. The time requirements for bias estimation using conventional and modified PNN's are given in Table I. This shows that the modified PNN model takes significantly less time compared to the conventional PNN for estimating the bias for the constraint satisfaction network.

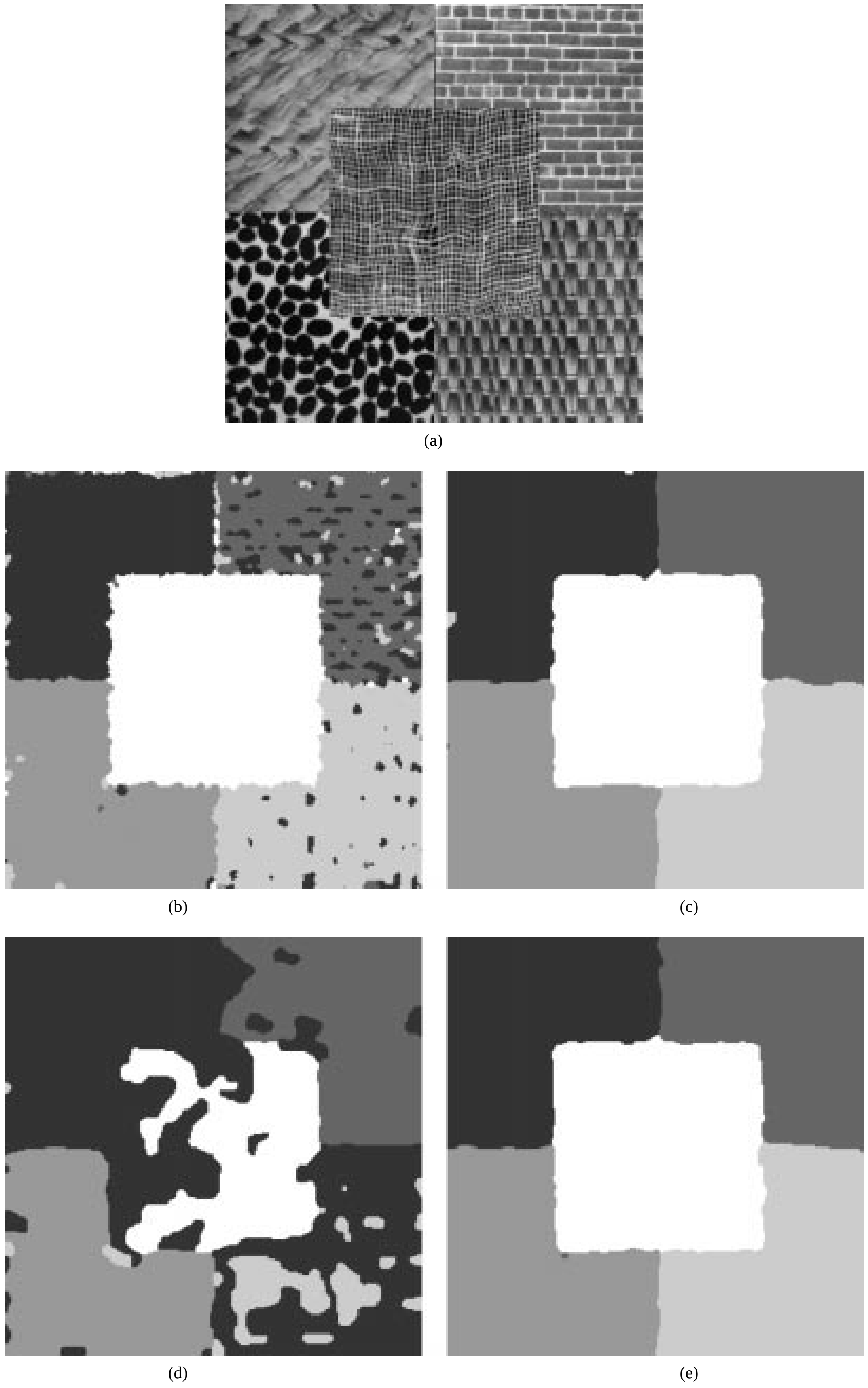

Fig. 3. Texture classification using the constraint satisfaction network: (a) image with five texture tiles, (b) result with multivariate Gaussian distribution, (c) result when PNN with $\sigma_{m,k} = 0.001$ is used (mixture of Gaussians), (d) result when PNN with $\sigma_{m,k} = 0.01$ is used (mixture of Gaussians), and (e) result when modified PNN model is used (mixture of multivariate Gaussians).

TABLE I
TIME TAKEN FOR ESTIMATING THE BIAS OF THE CONSTRAINT SATISFACTION NETWORK IN DIFFERENT CASES USING CONVENTIONAL AND MODIFIED PNN MODELS

Image	Class k	Training site size S_k	no. of mixture components Q_k	Bias estimation time (min)	
				Conventional PNN	Modified PNN
	1	1000	7		
	2	1000	10		
Fig. 3(a)	3	1000	9	439.26	21.44
	4	1000	15		
	5	1000	12		

V. CONCLUSION

In this paper, we have presented the texture classification as a constraint satisfaction problem, exploiting image-specific knowledge. To represent the image-specific constraints, we have used three random processes namely, feature formation process, partition process, and label competition process. For the formulation of feature formation process, we have used a Gaussian mixture model based on a PNN. The model was modified by incorporating a learning scheme using vector quantization principle. These image-specific constraints were represented on a constraint satisfaction neural-network model. In contrast with the conventional Hopfield network with a deterministic relaxation strategy which traps the network in local minima, the simulated annealing procedure was used here to find the global minimum state of the network. The results of our experimental studies show that the proposed mixture models using PNN are superior to multivariate Gaussian distribution for feature formation process.

An important observation of the constraint satisfaction model for classification of textured images is that the classification takes place for all the pixels of the image together. This is equivalent to say that the neural-network model is able to view the entire image to make decision on the different classes in the image. This is significant because this is one way to bring in the global image knowledge for making decision at the local level. It also corresponds to the way human beings perceive images for arriving at classification.

REFERENCES

[1] A. Blake, "The least disturbance principle and weak constraints," *Pattern Recognition Lett.,* vol. 1, pp. 393–399, 1983.

[2] A. C. Bovik, M. Clark, and W. S. Geisler, "Multichannel texture analysis using localized spatial filters," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. 12, pp. 55–73, Jan. 1990.

[3] R. Chellappa, B. S. Manjunath, and T. Simchony, "Texture segmentation with neural networks," *Neural Networks for Signal Processing,* B. Kosko, Ed. Englewood Cliffs, NJ: Prentice-Hall, 1992.

[4] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Statist. Soc. B,* vol. 39, pp. 1–38, 1977.

[5] J. M. H. du Buf, M. Kardan, and M. Spann, "Texture feature performance for image segmentation," *Pattern Recognition,* vol. 23, no. 3, pp. 291–309, 1990.

[6] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," in *Proc. Nat. Academic Sci.,* vol. 79, pp. 2554–2558, Apr. 1982.

[7] A. K. Jain and F. Farrokhnia, "Unsupervised texture segmentation using Gabor filters," *Pattern Recognition,* vol. 24, pp. 1167–1186, 1991.

[8] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science,* vol. 220, pp. 671–680, 1983.

[9] T. Kohonen, "The self-organizing map," *Proc. IEEE,* vol. 78, no. 9, pp. 1464–1480, Sept. 1990.

[10] B. S. Manjunath and R. Chellappa, "A unified approach to boundary perception: Edges, textures and illusory contours," *IEEE Trans. Neural Networks,* vol. 4, no. 1, pp. 96–108, Jan. 1993.

[11] E. Parzen, "On estimation of a probability density function and mode," *Ann. Math. Statist.,* vol. 33, pp. 1065–1076, 1962.

[12] P. P. Raghu, R. Poongodi, and B. Yegnanarayana, "A combined neural-network approach for texture classification," *Neural Networks,* vol. 8, no. 6, pp. 975–987, 1995.

[13] P. P. Raghu and B. Yegnanarayana, "Segmentation of Gabor filtered textures using deterministic relaxation," *IEEE Trans. Image Processing,* vol. 5, no. 12, pp. 1625–1636, Dec. 1996.

[14] ——, "Multispectral image classification using Gabor filters and stochastic relaxation neural network," *Neural Networks,* vol. 10, no. 3, pp. 561–572, 1997.

[15] H. Ritter, T. Martinetz, and K. Schulten, *Neural Computation and Self-Organizing Maps.* New York: Addison-Wesley, 1992.

[16] D. E. Rumelhart, P. Smolensky, J. L. McClelland, and G. E. Hinton, "Schemata and sequential thought processes in PDP models," in *Parallel Distributed Processing,* J. L. McClelland and D. E. Rumelhart, Eds., vol. 2. Cambridge, MA: MIT Press, 1986.

[17] D. F. Specht, "Generation of polynomial discriminant functions for pattern recognition," Ph.D. dissertation, Stanford Univ., CA, 1966.

[18] ——, *"Probabilistic Neural Networks,"* Neural Networks, vol. 3, pp. 109–118, 1990.

[19] J. Ton, J. Sticklen, and A. K. Jain, "Knowledge-based segmentation of Landsat images," *IEEE Trans. Geosci. Remote Sensing,* vol. 29, pp. 222–232, Mar. 1991.

P. P. Raghu was born in Kerala, India, in 1967. He received the B.Tech. degree in electrical engineering from Calicut University, Kerala, and the M.Tech. degree in computer and information sciences from Cochin University of Science and Technology, Kerala, in 1989 and 1991, respectively. He received the Ph.D. degree from the Indian Institute of Technology, Madras, India, in 1996.

From 1991 to 1995, he was Senior Project Officer in the Department of Computer Science and Engineering, Indian Institute of Technology. During 1996–1997, he worked as a Senior Systems Analyst for Tata Unisys Limited, Bangalore, India. In March 1997, he joined LG Software Development Center, Bangalore, as Senior Team Leader. His research interests include image processing, artificial neural networks, client-server applications development, and object-oriented software engineering methodologies.

B. Yegnanarayana (M'78–SM'84) was born in India on January 9, 1944. He received the B.E., M.E., and the Ph.D. degrees in electrical communication engineering from the Indian Institute of Science, Bangalore, India, in 1964, 1966, and 1974, respectively.

He was a Lecturer from 1966 to 1974 and an Assistant Professor from 1974 to 1978, in the Department of Electrical Communication Engineering, Indian Institute of Science. From 1966 to 1971, he was engaged in the development of environmental test facilities for the Acoustics Laboratory, Indian Institute of Science. From 1977 to 1980, he was a Visiting Associate Professor of Computer Science at Carnegie Mellon University, Pittsburgh, PA. He was a visiting scientist at ISRO Satellite Center, Bangalore, from July to December 1980. Since 1980, he has been a Professor in the Department of Computer Science and Engineering, Indian Institute of Technology, Madras. He was a visiting Professor at the Institute for Perception Research, Eindhoven Technical University, Eindhoven, The Netherlands, from July 1994 to January 1995. Since 1972, he has been working on problems in the area of speech signal processing. He is presently engaged in research activities in digital signal processing, speech recognition, and neural networks.

Dr. Yegnanarayana is a member of the Computer Society of India, a fellow of the Institution of Electronics and Telecommunications Engineers of India, and a Fellow of the Indian National Academy of Engineering.