COMPARISON OF TRADITIONAL AND NEURAL CLASSIFIERS FOR PAVEMENT-CRACK DETECTION[a]

By Mohamed S. Kaseko,[1] Member, ASCE, Zhen-Ping Lo,[2] and Stephen G. Ritchie,[3] Member, ASCE

(Reveiwed by the Highway Division)

ABSTRACT: This paper presents a comparative evaluation of traditional and neural-network classifiers to detect cracks in video images of asphalt-concrete pavement surfaces. The traditional classifiers used are the Bayes classifier and the k-nearest neighbor (k-NN) decision rule. The neural classifiers are the multilayer feed-forward (MLF) neural-network classifier and a two-stage piecewise linear neural-network classifier. Included in the paper is a theoretical background of the classifiers, their implementation procedures, and a case study to evaluate their performance in detection and classification of crack segements in pavement images. The results are presented and compared, and the relative merits of these techniques are discussed. The research reported in this paper is part of an ongoing research project, the objective of which is to develop a neural-network-based methodology for the processing of video images for automated detection, classification, and quantification of cracking on pavement surfaces.

INTRODUCTION

Currently, the predominant survey technique for collecting data on pavement-surface condition involves a manual process with pavement-maintenance personnel physically inspecting and recording the condition of the pavement. The principle limitations of this manual process relate to its safety, labor intensiveness, and subjectiveness. To improve the execution of this data-collection process and the quality of the data collected, a number of automated systems involving the acquisition and processing of pavement images have been proposed (Mendelsohn 1987; Butler 1989; Fukuhara et al. 1989; Caroff et al. 1990; Hosin 1990; Ritchie 1990; Ritchie et al. 1991; Fundakowski et al. 1991; Mahler et al. 1991). These systems are based on the application of computer-vision and image-processing techniques. While automation of the image-acquisition phase of the process has reached an advanced stage, there is scope for considerable improvement in the processing and interpretation of the images into distress classifications.

The results reported in this paper are part of an on-going research project, the objective of which is to demonstrate the feasibility of a new approach to pavement-surface-distress evaluation based on the automated processing and interpretation of video images. This new approach, which uses artificial neural networks for the pattern-classification stages of the approach, has

[a]This paper was presented in part at the ASCE International Conference on Applications of Advanced Technologies in Transportation Engineering held in Minneapolis in 1991.

[1]Asst. Prof., Dept. of Civ. and Envir. Engrg., Univ. of Nevada, Las Vegas, NV 89154-4015.

[2]Res. Specialist, Printrak International, Inc., Anaheim, CA 92807.

[3]Prof., Inst. of Transp. Studies and Dept. of Civ. and Envir. Engrg., University of California, Irvine, CA 92717.

Note. Discussion open until January 1, 1995. To extend the closing date one month, a written request must be filed with the ASCE Manager of Journals. The manuscript for this paper was submitted for review and possible publication on October 16, 1992. This paper is part of the ***Journal of Transportation Engineering***, Vol. 120, No. 4, July/August, 1994. ©ASCE, ISSN 0733-947X/94/0004-0552/$2.00 + $.25 per page. Paper No. 4709.

demonstrated the potential to accurately classify pavement images by type, severity, and extent of distress, and to distinguish between major types of cracking, such as transverse, longitudinal, alligator, and block cracking (Kaseko et al. 1993). The approach is divided into five major stages: (1) Image segmentation; (2) feature extraction; (3) decomposition of the image into tiles and identification of tiles with cracking; (4) integration of the results from step 3 and classification of the type of cracking in each image; and (5) computation of the severities and extents of cracking detected in each image. These stages are shown in Fig. 1.

Stages 3 and 4 of the approach, which are central to distress classification of the pavement images, involve pattern classification. Therefore, selection of a suitable pattern-classification methodology is a very important part of the development of the new approach. Our initial research on tile classification utilized a multilayer feed-forward neural network and a back-propagation learning algorithm (Ritchie et al. 1991). This paper presents a comparative evaluation of the performance of four different pattern-classification techniques using pavement-image data collected for the U.S. National Cooperative Highway Research Program (NCHRP) project 1-27 (Fundakowski et al. 1991). The classifiers evaluated include two well-known traditional classifiers, i.e. Bayes classifier and the k-nearest-neighbor (k-NN) classifier (Fukunaga 1972; Duda et al. 1973). The other two classifiers are based on applications of artificial neural-network models, one is a multilayer feed-forward (MLF) network, and the other is a two-stage piecewise linear neural classifier.

CLASSIFIER ALGORITHMS

Pattern classification can generally be defined as a mapping of a large set of unclassified data into a smaller set of prespecified classes using some measurement criterion. The process of pattern classification is mainly composed of two stages: a feature extraction stage, and a decision-making or classification stage. In the feature-extraction stage, the large-dimensional

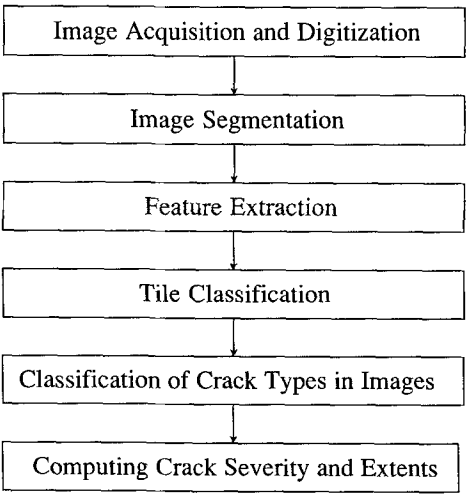


FIG. 1. Stages of Video-Image-Based Automated Pavement Surface Distress Evaluation System

pattern or signal is transformed into a set of measurements, which are called features. These features represent important intrinsic information about the patterns. In the second stage, a proper class or label is assigned to the feature vector according to a decision rule. Implementation of a decision rule usually involves the adaptation of some parameters. These parameters can be adapted by external signals or vectors so that a given performance of the classification is improved. To compare the classifier performance, the same features were used for testing all of the classifiers.

There are two approaches for designing the classifier, parametric and nonparametric. In the parametric approach, a statistical model of the data is postulated a priori, such as in the traditional Bayes classifier. In the nonparametric approach, there is no need to assume any statistical model for the data. Neural-network classifiers and k-NN classifiers are nonparametric classifiers.

Bayes Classifier

Let $C = \{c_1, c_2, \ldots, c_n\}$ be a finite set of n classes and $\mathbf{X}$ be an M-dimensional feature vector to be classified. The Bayes-decision theory is based on the assumption that the conditional density $p(\mathbf{X}|c_j)$ and the a priori probability $p(c_j)$ of every class c_j are known. Decision making depends on the posteriori probability $p(c_j|\mathbf{X})$ such that the vector $\mathbf{X}$ is assigned to class c_i, if

$$p(c_i|\mathbf{X}) > p(c_j|\mathbf{X}); \qquad \forall\, j \neq i \tag{1}$$

where

$$p(c_j|\mathbf{X}) = \frac{p(\mathbf{X}|c_j)p(c_j)}{P(\mathbf{X})} \tag{2}$$

and

$$P(\mathbf{X}) = \sum_{j=1}^{n} p(\mathbf{X}|c_j)p(c_j) \tag{3}$$

In pattern recognition, we rarely have complete knowledge about the probability density function of the problem. However, a sample-data set can be used to estimate the a priori probabilities and probability densities necessary to design a classifier. While estimation of the a priori probabilities is usually very simple, finding the class conditional densities is quite difficult. One way to simplify the problem is to assume that the form of $p(\mathbf{X}|c_j)$ is known. If, for example, we assume that $p(\mathbf{X}|c_j)$ has a Gaussian distribution, then $p(\mathbf{X}|c_j)$ can be estimated as a function of the mean vector $\mathbf{m}_j$ and covariance matrix $\mathbf{v}_j$ of each class c_j in the sample (design) data (Duda et al. 1973), i.e.

$$\mathbf{m}_j = \frac{1}{N_j} \sum^{N_j} \mathbf{X}; \qquad \forall\, \mathbf{X} \in c_j \tag{4}$$

and

$$\mathbf{v}_j = \frac{1}{N_j - 1} \sum^{N_j} (\mathbf{X} - \mathbf{m}_j)(\mathbf{X} - \mathbf{m}_j)^T \tag{5}$$

where N_j = total number of vectors of class c_j in the design data set. Then

$$p(\mathbf{X}|c_j) = \frac{1}{\sqrt{2\pi|\mathbf{v}_j|}} e^{-1/2(\mathbf{X}-\mathbf{m}_j)^T \mathbf{v}_j^{-1}(\mathbf{X}-\mathbf{m}_j)} \tag{6}$$

k-Nearest Neighbor (k-NN) Classifier

In the k-NN classifier, when a feature vector $\mathbf{X}$ is to be classified, the k-nearest neighbors of $\mathbf{X}$ are found among all the feature vectors in a design (or training) data set, and the vector $\mathbf{X}$ is assigned to the class most frequently represented amongst the k-nearest neighbors. Nearness is measured by any convenient metric, such as Euclidean distance. The idea of the k-NN classifier is based on the estimation of the class conditional densities. Let k_j be the number of class c_j vectors within the k-nearest neighbors of the input vector $\mathbf{X}$. Then the class conditional probability densities can be estimated as

$$p(\mathbf{X}|c_j) = \frac{k_j - 1}{N_j} \frac{1}{V} \tag{7}$$

where V = volume of the set of all points whose distance from $\mathbf{X}$ is less than the distance between $\mathbf{X}$ and the kth-nearest neighbor of $\mathbf{X}$. Since $\mathbf{k}_1, \ldots, \mathbf{k}_i, \ldots, \mathbf{k}_n$ are vectors drawn from the same hypersphere V, then, according to the Bayesian-decision rule, the vector $\mathbf{X}$ is assigned to class c_i if

$$\frac{N_i}{N} p(\mathbf{X}|c_i) > \frac{N_j}{N} p(\mathbf{X}|c_j); \qquad \forall\, j \neq i \tag{8}$$

which reduces to

$$k_i > k_j; \qquad \forall\, j \neq i \tag{9}$$

where N = total number of vectors for all classes. This means that the vector $\mathbf{X}$ can be classified by simply determining k_j, where $j = 1, \ldots, n$, after choosing the k-nearest neighbors.

As can be noted from the aforementioned, the k-NN decision rule is a nonparametric classification procedure. It does not require a priori knowledge of the density functions of the classes. However, the probability of misclassification, R, of the k-NN decision rule has been shown to always be greater than the minimum possible within the boundaries (Cover et al. 1967)

$$R^* \leq R \leq R^* \left(2 - \frac{nR^*}{n-1}\right) \tag{10}$$

where R^* = minimum possible Bayes probability of error; and n = number of classes. This means that for data with a known probability density function, if the Bayes classifier is correctly modeled, it should always perform better than the k-NN classifier.

Multilayer Feed-Forward Neural Network (MLF)

An artificial neural network is a parallel, distributed-information processing system composed of many simple processing elements, which are interconnected via synaptic or weighted connections. Each processing element receives and processes weighted inputs from previous processing elements and transmits its output to the following set of processing elements

through another set of weighted interconnections. An important mechanism for the neural network is how to adapt these connection weights so that the network processes the information properly. The processing of adaption of the weights is called learning. Learning methods can be classified into two categories, supervised learning and unsupervised learning. Supervised learning is a process that requires an external teacher or global information; i.e., it requires the data set used for learning to have a desired output corresponding to each input. Unsupervised learning does not require an external teacher; when input-pattern vectors are presented to the neural network, the network automatically extracts important features from the input vectors and self-organizes the input data into different clusters based on these extracted feature vectors.

Recent work on artificial neural networks raises the possibility of new approaches to the pattern-classification problem. Neural networks offer several potential advantages over existing pattern-classification approaches (Rumelhart et al. 1986; Lo et al. 1991a). Multilayer feed-forward (MLF) neural networks, trained using the back-propagation algorithm (Rumelhart et al. 1986), are the most commonly used neural approaches for nonparametric pattern classification. The MLF neural network consists of three or more layers of processing elements with each processing element in a layer connected to all processing elements in the preceding and following layers of processing elements through weighted interconnections. Such a network has the ability to generate any number of hyperplanes to separate classes. In implementation of the neural network for pattern classification, the inputs into the MLF are the feature vectors or parameters of the input patterns; therefore, the number of processing elements in the first layer is always set equal to the number of elements in the input vector. The output vector produced by the MLF should correspond to the class membership of the input pattern and therefore the number of processing elements in the output layer is normally set equal to the number of classes. However, the number of processing elements in the intermediate (hidden) layer(s) depends on the complexity of the input-vector space and can often only be determined empirically by trial and error. In this study, we use a three-layer MLF neural network, which has one hidden layer between the input and output layers. Before the network can be used for classification, it has to be trained so that it can be able to produce the desired outputs when presented with the input patterns. The objective of the training is to let the network adapt the connection weights and other parameters such that the classification-error rate is minimized.

Assume that the network has M processing elements in the input layer, Q processing elements in the output layer, and H processing elements in the hidden layer. Then, the output activation of the lth processing element of the hidden layer is given by

$$h_l(t + 1) = g\left[\sum_{i=1}^{m} W_{il}(t)x_i + \theta_l\right] \tag{11}$$

and those of the output-layer processing elements are given by

$$o_j(t + 1) = g\left[\sum_{l=1}^{H} V_{lj}(t)h_l + \phi_j\right] \tag{12}$$

where t = discrete time index; x_i = output of the ith processing element

of input layer; W_{il} = weight of the interconnection between processing element i of the input layer and processing element l of the hidden layer; and V_{lj} = weight of the interconnection between processing element l of the hidden layer and processing element j of the output layer. Moreover, θ_l and ϕ_j = thresholds for the lth processing element in the hidden layer and the jth processing element in the output layer, respectively; and $g(.)$ = a nonlinear activation function, which is often a sigmoid function. The learning rule for the MLF neural network uses simple-error feedback to learn the associations between inputs and outputs through a training-data set. The procedure used is known as error back propagation (Rumelhart et al. 1986) and is briefly discussed in this paper. The error associated with the processing element j in the output layer of the MLF is defined as

$$e_j = d_j - o_j \tag{13}$$

where d_j and o_j = desired output and the actual MLF output, respectively, of the jth processing element of the output layer. This error is used to adjust the weights of the connections feeding into the output layer by using the relationship

$$V_{lj}(t + 1) = V_{lj}(t) + \delta V_{lj}(t) \tag{14}$$

where

$$\delta V_{lj}(t) = \eta h_l \delta o_j + \alpha \delta V_{lj}(t - 1) \tag{15}$$

and

$$\delta o_j = e_j o_j (1 - o_j) \tag{16}$$

where η = a parameter known as the learning rate and α = momentum gain. The thresholds in the output-layer processing elements are adjusted according to the relationship

$$\phi_j(t + 1) = \phi_j(t) + \delta \phi_j(t) \tag{17}$$

where

$$\delta \phi_j(t) = \eta \delta o_j + \alpha \delta \phi_j(t - 1) \tag{18}$$

Similarly, adaptation of weights for the hidden-layer processing elements is given by

$$W_{il}(t + 1) = W_{il}(t) + \delta W_{il}(t) \tag{19}$$

where

$$\delta W_{il}(t) = \eta x_i \delta h_l + \alpha \delta W_{il}(t - 1) \tag{20}$$

and

$$\delta h_l = h_l(1 - h_l) \sum_{j=1}^{Q} \delta o_j V_{lj} \tag{21}$$

The threshold of the hidden layer is adjusted according to

$$\theta_l(t + 1) = \theta_l(t) + \delta \theta_l(t) \tag{22}$$

where

$$\delta\theta_l(t) = \eta\delta h_l + \alpha\delta\theta_l(t - 1) \tag{23}$$

The design of the multilayer feed-forward neural network can be summarized as follows:

- Step 1: Specify the number of layers and the number of processing elements for each layer.
- Step 2: Set all weight parameters and offsets to the small random values.
- Step 3: Present an input vector and its corresponding target output to the neural network.
- Step 4: Use (12) to calculate outputs of the network and (13) to calculate the error between the target output and the network output.
- Step 5: Adapt the weights and offset parameters starting at the output layer and working back to the first layer according to (14)–(23).
- Step 6: Repeat by going to step 3 until the summation error of the training-data set is stabilized.

Two-Stage Piecewise Linear Neural Classifier

The second neural network used in our study is a two-stage piecewise linear neural classifier (Lo et al. 1991b). This recently proposed neural network has shown good performance in classification problems in several real-data sets (Lo et al. 1991b; Lo et al. 1992). The idea of the piecewise linear neural classifier is to find the best representative vectors for each class, such that the classification error is minimized when these vectors are used as prototype vectors to classify given input vectors whose class memberships are unknown. An input pattern or vector is assigned to the class membership corresponding to its nearest neighboring prototype vector. Each such prototype vector in the classifier corresponds to a synaptic weight vector of one processing element. The training of this neural classifier is then a process of adapting values to these prototype vectors.

The classifier has two stages and a feedback loop. In the first stage, the competitive learning rule (Rumelhart et al. 1986) is used to find the approximate positions of a set of prototype vectors representative of each class. This is an unsupervised learning algorithm, which converges to an approximate solution, learning the distribution of each class in a separate neural-network module. Hence, for n classes, n such modules are developed in the first stage, one module for each class. The number of processing elements in each module corresponds to the number of prototype vectors representative of each class. The discrete adaptation form of the competitive learning rule is

$$\mathbf{Z}_{lj}(t + 1) = \mathbf{Z}_{lj}(t) + \eta(t)[\mathbf{X} - \mathbf{Z}_{lj}(t)] \tag{24}$$

where $\mathbf{Z}_{lj}$ = lth prototype vector in module j. Alternatively, $\mathbf{Z}_{lj}$ = weight vector of the lth processing element in module j. Note that in this case, the learning rate η should be a decaying function of time in order to achieve convergence.

In the second stage, the Kohonen learning vector quantization 2 (LVQ2) (Kohonen et al. 1988) supervised learning algorithm is used to make fine adjustments to the positions of the representative prototype vectors in each of the modules to reduce the classification error. Furthermore, the accuracy

of the classifier may be improved by adding an adaptive feedback scheme, which may further reduce the classification error by iteratively varying the number of prototype vectors in each module during the learning process and selecting the topology producing the minimum classification error. The optimum number of prototype vectors in each module depends on the complexity of the class distribution and overall partitioning of the data structure and can only be determined empirically by such an iterative trial-and-error procedure.

In the LVQ2 algorithm, the adaptation of a weight vector is carried out only if a misclassification of the input vector occurs and several conditions are satisfied (Kohonen et al. 1988). The conditions depend on the position of input vector, $\mathbf{X}$, relative to the position of the hyperplane (i.e. decision surface) separating class c_i, which is the correct classification for the input vector, and c_j to which the vector has been assigned. Adjustments to the weights are carried out such that the prototype vector associated with the correct classification, $\mathbf{Z}_{li}$, is moved closer to $\mathbf{X}$, while $\mathbf{Z}_{hj}$ is moved farther away from $\mathbf{X}$, i.e.

$$\mathbf{Z}_{li}(t+1) = \mathbf{Z}_{li}(t) + \eta(t)[\mathbf{X} - \mathbf{Z}_{li}(t)] \tag{25}$$

$$\mathbf{Z}_{hj}(t+1) = \mathbf{Z}_{hj}(t) - \eta(t)[\mathbf{X} - \mathbf{Z}_{hj}(t)] \tag{26}$$

Such a training rule moves the hyperplane towards the correct classification. Thus, the weight vectors associated with the processing elements will asymptotically best represent the classes. In all the other cases, no weight adaptation is carried out and the hyperplane is not changed. The algorithm for designing the classifier is summarized as follows:

- Step 1: Specify the number of processing elements for each class module of the neural network, give the error threshold, ε (classification-error rate), and specify upper bound on the number of processing elements for each class.
- Step 2: Present an input vector, $\mathbf{X}$, to the neural network classifier.
- Step 3: If $\mathbf{X}$ belongs to class c_i, then let the ith module learn it using the competitive learning rule.
- Step 4: Go to step 2 until all the synaptic-weight vectors in each module converge to local means.
- Step 5: Assign the weight vectors of each module the initial value for the second stage of training.
- Step 6: Present the new input vector $\mathbf{X}$ to the network.
- Step 7: Update the weight vectors according to the LVQ2 algorithm.
- Step 8: Go to step 6 until the learning converges.
- Step 9: Check the classification-error rate and the number of processing elements of each class. If the error rate is smaller than the error computed in the previous iteration and larger than the error threshold, ε, and the number of processing elements is less than an upper bound, then increase the number of processing elements of the module corresponding to that class and go to step 2, otherwise stop.

DATA DESCRIPTION

Image Acquisition, Digitization, and Segmentation

Pavement images are typically acquired by video or 35-mm cameras mounted on vehicles that travel over the pavement at up to normal highway speeds.

The images are then digitized, often into 512 × 512 pixel 256-gray scale images, for further processing and interpretation. The pavement images used in this research were a collection of images compiled and stored in a laser video disk by Triple Vision, Inc. of Minneapolis for NCHRP Project 1-27 (Fundakowski et al. 1991). This is the same data set as the one reported in Kaseko and Ritchie (1993). There were over 10,000 full-lane width images of highways from different parts of the United States, with each image representing approximately 3.6 m × 3.6 m of pavement surface. To improve the resolution of the images, the laser disc also contained zoomed images, each zoomed image being a quadrant of the full-lane image, representing 1.8 m × 1.8 m of pavement surface. The laser disk contained images representing all the major types of pavement distress, including alligator, transverse, longitudinal, and block cracking for both asphalt-concrete and portland-cement concrete pavements.

About 250 asphalt-concrete pavement zoomed images were selected for use in this research. The selection was done such that each of the distress types considered in this research was well represented. These 1.8 m × 1.8 m images were then digitized into 512 × 464 pixel images with an eight-bit gray scale resulting in a resolution of about 3.7 mm per pixel. Fig. 2 shows an example of a raw digitized image.

Before the images can be classified, they have to be processed for purposes

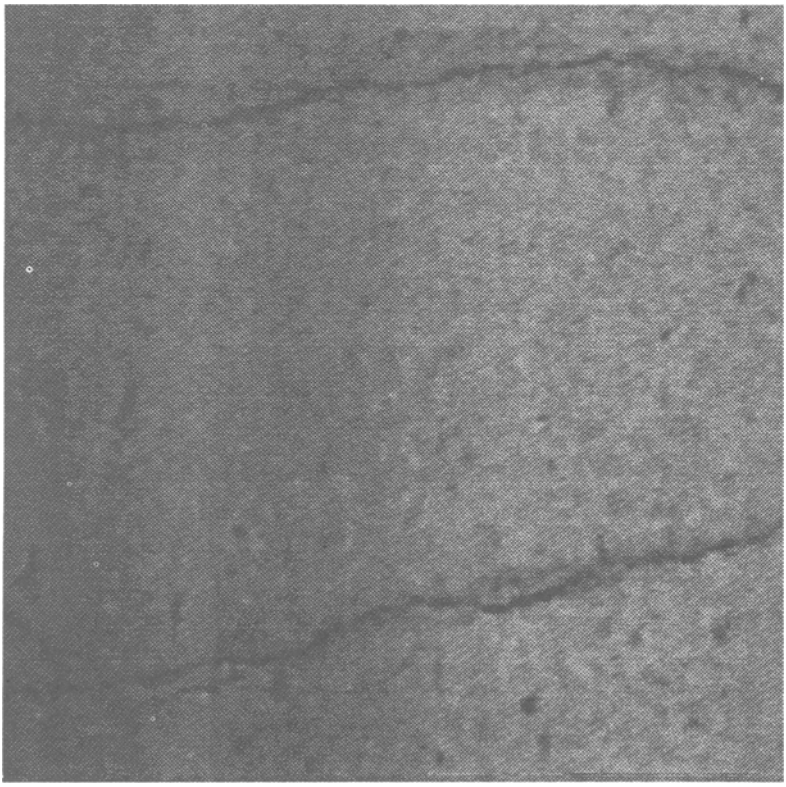


FIG. 2. Raw Pavement Image

of extracting the objects of interest in the image, i.e. cracks. This process, known as image segmentation results in binary images, whereby each pixel on an image takes only one of two states, i.e. it is either an object pixel or a background pixel. Image segmentation is particularly difficult with pavement images mainly due to characteristics of pavement texture, occasional existence of nonuniform background lighting, shadow effects, and the relatively smaller number of object pixels in pavement images compared to background pixels. For this reason, most traditional image-segmentation techniques do not work satisfactorily with pavement images, and hence research in the area is ongoing (El-Korchi et al. 1990; Wittels et al. 1990; Koutsopoulos et al. 1991; Lan et al. 1991). Probably the most common technique for image segmentation is thresholding, whereby, a pixel in a gray-scale image is identified as an object pixel if its brightness value is below a threshold value, otherwise it is identified as a background pixel. In this research, thresholding of the images is performed automatically using a neural-network-based methodology. Before thresholding, the images are first normalized, a process that transforms an image into one with more uniform background brightness across the image and which results in improved performance of the thresholding process. These processes are described in Kaseko and Ritchie (1993).

Feature Extraction

In pavement-image processing, feature extraction is typically performed on binary images. Suitable features that provide information on line locations, orientations, lengths, and thicknesses have to be extracted from the images. These features are later used for classification and quantification of pavement surface distresses by types, severity, and extent of cracking. Some of the conventional methods for the feature extraction process include the Hough transform and some line fitting algorithms (Duda et al. 1973; Otsu 1984). These techniques are generally not suitable for application in automated pavement-image processing because they tend to be too computationally intensive and do not provide information on line lengths and thicknesses.

The feature parameters extracted from the images in this research were derived from projection histograms of the number of distressed pixels in a binary image. A projection histogram is a histogram of the number of distressed pixels per line perpendicular to the direction of projection. In this study, these histograms are applied on subdivisions of the images, called tiles, and four directions of projections are used, namely, transverse, longitudinal, and the two diagonal directions. Each 512 × 464 image is divided into 256 tiles, with each tile being 32 × 29 pixels in size. Fig. 3 shows an example of a thresholded image divided into subimages. The trade-offs involved in selection of the tile size are discussed in Kaseko and Ritchie (1993). From these histograms the following parameters for each tile are computed:

- a_m = the relative number of distressed pixels as a proportion (percentage) of the total number of pixels in the tile;
- v_{tr} = variance of the number of distressed pixels per line in the transverse direction;
- v_{ln} = variance of the number of distressed pixels per line in the longitudinal direction;

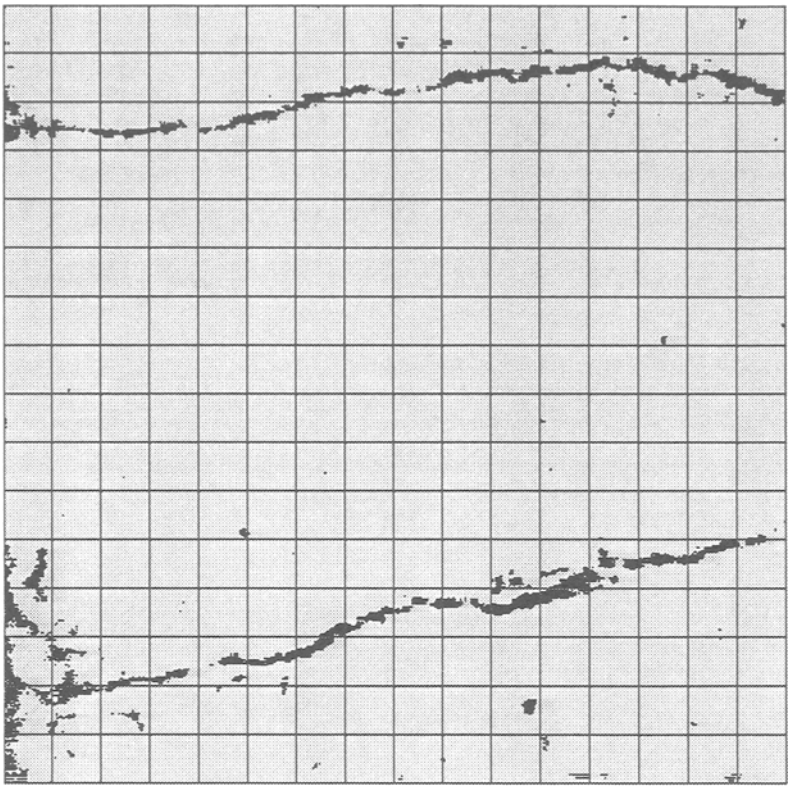
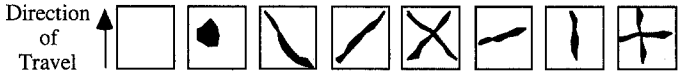


FIG. 3. Thresholded and Segmented Image Divided into Subimages (Tiles)



Transverse:	L	H	L	L	L	H	L	H
Longitudinal:	L	H	L	L	L	L	H	H
Diagonal-1:	L	H	H	L	H	L	L	L
Diagonal-2:	L	H	L	H	H	L	L	L

L: Low variance; H: High Variance.

FIG. 4. Variances in Each Direction versus Orientation of Cracking

- v_{d1} = variance of the number of distressed pixels per line in a diagonal direction; and
- v_{d2} = variance of the number of distressed pixels per line in the other diagonal direction.

Fig. 4 illustrates how the values of these parameters compare for different orientations of cracking in an image. These parameters result in a feature vector

$$\mathbf{X} = (a_m, v_{tr}, v_{ln}, v_{d1}, v_{d2}) \quad (27)$$

for each tile. These feature vectors are then used in the tile-classification stage to identify the existence and orientation of crack segments in each tile. This is a pattern-classification process and the selection of a suitable pattern classifier for use in the classification of the tiles is the focus of this paper.

CASE STUDY

Application of Classifiers

The four classifiers were evaluated by comparing their performance on the classification of tiles by the type of cracking present in the tiles. Each tile was to be classified into one of the following five classes, which describes the existence and orientation of crack segments in the tiles:

- No cracking
- Transverse
- Longitudinal
- Diagonal
- Combination cracking

A tile with combination cracking is one with two or more crack segments at different orientations. At this stage of the process, no attempt is made to detect alligator and block cracking, potholes, and patches, since the procedure only focuses on identification of crack segments within the individual tiles independently. Integration of the tile classification results for analysis of entire images is discussed in Kaseko and Ritchie (1993) but is outside the scope of this paper.

The data set used for evaluation of the classifiers was generated from 20 of the 250 images, selected at random. The actual classification of each tile was determined by human visual observation of two observers. Two data sets were generated; one with 230 tiles was used as a training-data set, and the other with 230 tiles was used as a test-data set. The training-data set was used to train each of the classifiers and the test-data set was used to evaluate the performance of each classifier in terms of how accurately it was able to reproduce the actual tile classifications in the test-data set. The classifiers were coded in the C programming language on a SUN SPARC station computer. For the Bayes classifier, a Gaussian distribution model for the data was assumed. Thus, the mean vector $\mathbf{m}_i$ and covariance matrix

TABLE 1. Classification Performance of Bayes Classifier

Type of cracking (1)	Number of Tiles Classified by Bayes Classifier					Accuracy (%) (7)
	None (2)	Transverse (3)	Longitudinal (4)	Diagonal (5)	Combination (6)	
None	28	4	2	0	1	80.0
Transverse	2	45	0	0	3	90.0
Longitudinal	0	0	50	1	2	94.3
Diagonal	1	2	0	37	2	88.1
Combination	0	4	4	0	42	84.0

Note: Total accuracy = 87.8%.

$\mathbf{v}_i$ of the training data set for each class i was computed and used for classification of the feature vectors in the test data set. The classification results are given in Table 1. In the table, the diagonal entries are the number of correctly classified patterns. The off-diagonal entries in each row are the misclassified patterns. For example, in the second row of the table, 45 of the 50 tiles with transverse cracking are correctly classified while 3 are misclassified as having combination cracking, and two having no cracking. This translates into a 90% accuracy rate for detection of tiles with transverse cracking. In the k-NN classifier, of the different values of k tested, the best results were found when $k = 1$ (Table 2).

The three-layer MLF network was implemented with five processing elements each in the input and output layers, corresponding to the number of features in the input vectors and the number of output classes, respectively. Many experiments were carried out testing the effect of the various parameters, including the learning rate, η; momentum gain, α; and number of processing elements in the hidden layer, on the performance of the network. Each of these features can affect not only the convergence rate of the network, but also the accuracy of classification of the test-data set. For example, too few hidden processing elements may not be able to capture the essential features of the presented training data, while too many hidden processing elements may also result in poor classification accuracy. The parameters η and α are normally restricted to values between 0 and 1. Lower values of η and α may make the network take longer to converge than with corresponding higher values. However higher values of η and α may cause network instability. After experimenting with several different combinations of these parameters, the relatively best results (Table 3) were

TABLE 2. Classification Performance of k-NN Classifier

Type of cracking (1)	Number of Tiles Classified by k-NN Classifier					Accuracy (%) (7)
	None (2)	Transverse (3)	Longitudinal (4)	Diagonal (5)	Combination (6)	
None	32	0	3	0	0	91.4
Transverse	3	46	0	0	1	92.0
Longitudinal	1	0	50	0	2	94.3
Diagonal	4	2	1	34	1	81.0
Combination	0	0	3	0	47	94.0

Note: Total accuracy = 90.9%.

TABLE 3. Classification Performance of Multilayer Neural Network Classifier

Type of cracking (1)	Number of Tiles Classified by MLF Classifier					Accuracy (%) (7)
	None (2)	Transverse (3)	Longitudinal (4)	Diagonal (5)	Combination (6)	
None	35	0	0	0	0	100.0
Transverse	3	46	0	0	1	92.0
Longitudinal	1	0	52	0	0	98.1
Diagonal	1	0	2	37	2	88.1
Combination	0	0	2	4	44	88.0

Note: Total accuracy = 93.0%.

obtained when the number of hidden processing elements, the learning gain, and the momentum rate were set at 5, 0.1, and 0.7, respectively.

For the piecewise linear neural classifier, the learning rate, $\eta(t)$, is very critical to the convergence of the network and is normally a decaying function ranging between 1 and 0. Several experiments were carried out with different function forms of $\eta(t)$ and the following functional form was found to be most appropriate for this study:

$$\eta(t) = a + \frac{1}{b + ct} \tag{28}$$

where t = time index; and parameters a, b, and c, = constants, which determine the shape and rate of decay of the learning function and therefore affect the rate of convergence. The most suitable parameter values are determined empirically. Parameter a is usually set to be less than 0.01, while parameter b is set to be between 0.95 and 1 and parameter c is set to be between 0 and 0.1. Selection of the number of processing elements in each module depends on the specified error threshold or classification-error rate. If the minimum possible classification error is to be achieved, then the network has to be trained several times varying the number of processing elements in the modules from 1 to the upper bound. The topology providing the minimum classification rate is then selected for implementation. The upper bound for the number of processing elements is normally 10 M, based on our experience, where M is the dimension of the input vector. The best results for this network, shown in Table 4, were obtained when the parameter values were set at $a = 0.0001$; $b = 0.96$; and $c = 0.05$. The corresponding value for the number of processing elements in each module was three.

Comment on Results

Of the four classifiers implemented in this study, the results show that the neural classifiers performed slightly better than the two traditional classifiers. However, in order to achieve this good classification performance, several parameters such as the number of processing elements in the hidden layer, learning rate, and momentum term in the MLF network had to be carefully selected. Similarly, for the piecewise linear neural classifier the decay function of the learning rate, the processing elements in each module, and the time to terminate the training also had to be carefully selected. Since this can only be done empirically, training of the classifiers can be a time-consuming process. However, once trained, a neural classifier can

TABLE 4. Classification Performance of Neural Piecewise Linear Classifier

Type of cracking (1)	Number of Tiles Classified by Piecewise Classifier					Accuracy (%) (7)
	None (2)	Transverse (3)	Longitudinal (4)	Diagonal (5)	Combination (6)	
None	34	0	1	0	0	97.1
Transverse	3	45	0	0	2	90.0
Longitudinal	1	0	49	1	2	92.5
Diagonal	0	0	0	40	2	95.2
Combination	0	1	1	0	48	96.0

Note: Total accuracy = 93.9%.

perform classifications very fast. In addition, these neural classifiers offer several other potential advantages over the traditional approaches. Neural networks can provide the high computation rates required in pattern recognition problems using many simple processing elements operating in a parallel-distributed fashion. Neural-network algorithms are self-organizing, and be designed to capture new phenomena as they are observed, which makes them good candidates for building adaptive classifiers. The networks provide a greater degree of fault tolerance because computation is distributed among many processing elements, each with primarily local connections. In general, damage to a few nodes or links may not impair the overall performance of the network significantly.

For a large design-data set, the performance of k-NN should normally be very good. However, there are two main drawbacks with the k-NN classifier. It is necessary to store and keep all the training data in computer memory. Also, during the classification process, each vector to be classified has to be compared to all the vectors in the training-data set for classification. This results in a very computationally intensive procedure, which is a significant disadvantage for real-time application.

The Bayes classifier results were not as good as the other three classifier results possibly because the data did not fit well the assumed Gaussian distribution. However, if the underlying density functions for the input data were known, it is possible that the results could have easily matched or even surpassed those of the other classifiers. Thus, if the underlying density functions for the input data are known, it is generally more advantageous to design an optimal classifier using the Bayesian approach, since the Bayes classifier is relatively easier to develop and implement. However, the fact that in many pattern recognition applications the underlying statistical distributions of data or their functional forms are unknown, makes this approach less attractive. The common Gaussian forms often do not fit the densities actually encountered in practice. But it should be noted also that there are procedures for designing other classifiers using nonparametric approaches (Duda et al. 1973). These procedures are generally very computationally intensive and their performance greatly depends on the selection of various input parameters and are unlikely to match the performances of the neural classifiers.

CONCLUSION

Four classifiers have been implemented to detect cracks in pavement video images. The results show that the neural-network classifiers performed slightly better than the two traditional classifiers on the test-data set. However, in order to achieve such performance, several parameters needed to be carefully selected and extensive empirical training performed. The relative advantages and disadvantages of the four classifiers have been discussed in the paper. This has provided an experimental basis for the decision to integrate a neural-network classifier in the classification stage of the automated pavement distress evaluation system under development by the writers. It is hoped that the results of this study will provide additional insight to researchers and practitioners into the process of selecting suitable classifiers for use in pattern-classification problems in transportation engineering.

ACKNOWLEDGMENTS

This research was supported by the National Science Foundation under Grant No. MSM-8657501. We also wish to thank Triple Vision Inc. of Minneapolis, Minnesota and the National Cooperative Highway Research Program (NCHRP) for their assistance in providing us with the images used in this research. We extend our thanks also to the Department of Teacher Education, University of California, Irvine, for letting us use their computer facilities for some of our work and Prof. Behnam Bavarian of the Department of Electrical and Computer Engineering at the University of California, Irvine, for his early advice on various aspects of implementation of neural-network models.

APPENDIX I. REFERENCES

Butler, B. (1989). "Pavement surface distress segmentation using real-time imaging." *Proc., 1st Int. Conf. on Applications of Advanced Technol. in Transp. Engrg.*, ASCE, New York, N.Y.

Carroff, G., Leycure, P., Prudhomme, F., and Soussain, G. (1990). "MACADAM: An operating system of pavement deterioration diagnosis by image processing." *Paper No. 890393, 69th Annu. Transp. Res. Board Meeting*, Washington, D.C.

Cove, T. M., and Hart, P. E. (1967). "Nearest neighbor pattern classification." *IEEE Trans. on Information Theory*, 13(1), 21–27.

Duda, R. O., and Hart, P. E. (1973). *Pattern classification and scene analysis*. John Wiley & Sons, New York, N.Y.

El-Korchi, T., Gennert, M. A., Ward, M. O., and Wittels, N. (1990). "An engineering approach to automated pavement surface distress evaluation." *Proc., Automated Pavement Distress Data Collection Equipment Seminar*, Federal Highway Administration, Ames, Iowa, 165–172.

Fukuhara, T., Terada, K., Nagao, M., Kasahara, S., and Ichihashi, J. (1989). "Automatic pavement distress system." *Proc., 1st Int. Conf. on Applications of Advanced Technol. in Transp. Engrg.*, ASCE, New York, N.Y.

Fukunaga, K. (1972). *Introduction to statistical pattern recognition*. Academic Press, New York, N.Y.

Fundakowski, R. A., Graber, R. K., Fitch, R. C., Skok, E. L., and Lukanen, E. O. (1991). "Video image processing for evaluating pavement surface distress." *Final Rep. for the Nat. Cooperative Hwy. Res. Program (NCHRP)*, Triple Vision, Inc., Project 1-27, Minneapolis, Minn.

Hosin, L. (1990). "Evaluation of pavedex-computerized pavement image processing system in Washington." *Proc., Automated Pavement Distress Data Collection Equipment Seminar*, Federal Highway Administration (FHWA), Washington, D.C.

Kaseko, M. S., and Ritchie, S. G. (1993). "A neural network-based methodology for pavement crack detection and classification." *Transp. Res., Part C*, 1(4), 275–291.

Kohonen, T., Barna, G., and Chrisley, R. (1988). "Statistical pattern recognition with neural networks: benchmarking studies." *Proc., IEEE Int. Conf. on Neural Networks*, Vol. 1, IEEE, San Diego, Calif., 182–185.

Koutsopoulos, H. N., and Sanhouri, I. E. (1991). "Methods and algorithms for automated analysis of pavement images." *Transp. Res. Record, No. 1311*, National Research Council, Washington, D.C., 103–111.

Lan, L., Chan, P., and Lytton, R. L. (1991). "Detection of thin cracks on noisy pavement images." *Transp. Res. Record, No. 1311*, National Research Council, Washington, D.C., 133–135.

Lo, Z. P., and Bavarian, B. (1991a). "Comparison of a neural network and linear classifier." *Pattern Recognition Letters*, Vol. 12, 649–655.

Lo, Z. P., and Bavarian, B. (1991b). "A neural piecewise linear classifier for pattern classification." *Proc., IEEE Int. Joint Neural Network Conf.*, Vol. 1, 264–268.

Lo, Z. P., Yu, Y. Q., and Bavarian, B. (1992). "Derivation of learning vector

quantization algorithms." *Proc., IEEE Int. Joint Conf. on Neural Networks*, Vol. 3, IEEE Neural Network Council, Baltimore, Md., 561–567.
Mahler, D. S., Kharoufa, Z. B., Wong, E. K., and Shaw, L. G. (1991). "Pavement distress analysis using image processing techniques." *Microcomp. in Civ. Engrg.*, Vol. 6, Elsevier Science Publishers Ltd., New York, N.Y., 1–14.
Mendelsohn, D. H. (1987). "Automated pavement crack detection: an assessment of leading technologies." *Proc., 2nd North Am. Conf. on Managing Pavements*, Vol. 3, FHWA, Washington, D.C., 297–314.
Otsu, N. (1984). "Karhunen-Loeve line fitting and linearity measure." *Proc., Int. Conf. on Pattern Recognition*, IEEE Computer Society Press, Silver Spring, Md., 486–489.
Ritchie, S. G. (1990). "Digital imaging concepts and applications in pavement management." *J. Transp. Engrg.*, ASCE, 116(3).
Ritchie, S. G., Kaseko, M. S., and Bavarian, B. (1991). "Development of an intelligent system for automated pavement evaluation." *Transp. Res. Record, No. 1311*, National Research Council, Washington, D.C., 112–119.
Rumelhart, D. E., McClelland, J. L., and the PDP Research Group. (1986). *Parallel distributed processing: explorations in the microstructure of cognition, Vol. I*, MIT Press, Cambridge, Mass.
Wittels, N., El-Korchi, T., Gennert, M. A., and Ward, M. O. (1990). "Images for testing pavement surface distress evaluation systems." *Proc., Automated Pavement Distress Data Collection Equipment Seminar*, Federal Highway Administration, Ames, Iowa, 153–163.

APPENDIX II. NOTATION

The following symbols are used in this paper:

C = finite set of n classes;
c_j = class j;
d_j = desired output value for jth PE in output layer;
e_j = error in output value of jth PE in output layer;
H = number of processing elements in hidden layer of MLF;
h_l = output value of lth PE in hidden layer;
k_i = number of class **i** vectors within k nearest neighbors;
M = number of elements in feature vector **X**;
$\mathbf{m}_i$ = vector of means for class **i** vectors;
N = total number of vectors of all classes;
N_i = total number of class i vectors in training set;
n = total number of classes;
o_j = output value of jth PE in output layer;
Q = number of processing elements in output layer of MLF;
R = probability of error in classification;
R^* = minimum possible Bayes probability of error;
t = discrete time index;
V = volume of set of points whose distance from **X** is less than distance between **X** and kth-nearest neighbor;
V_{il} = interconnection weight between ith input element and lth PE in hidden layer;
$\mathbf{v}_i$ = covariance matrix for class **i** vectors;
W_{lj} = interconnection weight between lth PE in hidden layer and jth output PE;
$\mathbf{X}$ = input feature vector to be classified;
x_i = element i of **X**;

$\mathbf{Z}_{lj}$ = lth prototype vector in module j;
α = momentum gain;
η = learning rate;
θ_l = threshold value for lth PE in hidden layer; and
ϕ_j = threshold value for jth PE in output layer.