PERGAMON

Pattern Recognition 35 (2002) 2355–2364

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Robust vision-based features and classification schemes for off-line handwritten digit recognition

Loo-Nin Teow, Kia-Fock Loe*

School of Computing, National University of Singapore, Science Drive 2, S 117559, Singapore

Received 9 July 2001; accepted 9 October 2001

Abstract

We use well-established results in biological vision to construct a model for handwritten digit recognition. We show empirically that the features extracted by our model are linearly separable over a large training set (MNIST). Using only a linear discriminant system on these features, our model is relatively simple yet outperforms other models on the same data set. In particular, the best result is obtained by applying triowise linear support vector machines with soft voting on vision-based features extracted from deslanted images. © 2002 Pattern Recognition Society. Published by Elsevier Science Ltd. All rights reserved.

Keywords: Handwritten digit recognition; Biological vision; Feature extraction; Linear discrimination; Multiclass classification

1. Introduction

Automated handwritten character recognition has been an active area of research and development for at least two decades [1–8]. The literature on this topic alone is extremely huge, with a large variety of feature extraction and classification techniques being published every year. Features extracted range from geometric moments to contours and curvatures, while classification techniques range from template matching to neural networks. Some draw inspiration from biological systems, while others are based on statistics or geometry.

There are two main approaches to feature extraction. The more traditional approach is to handcraft the feature extraction process, as opposed to the other approach whereby the raw input is presented to a learning algorithm to discover whatever features are inherent in the domain. Each approach has its own merits and weaknesses. In the former approach, the main difficulty lies in determining the appropriate class of features to extract as well as in extracting those features in a robust and reliable way. Automated learning of features, on the other hand, is feasible only when there are a large number of samples available for each class. Hence, it may not be feasible for Kanji or Chinese characters, whereby the number of samples for each class is relatively few. In addition, in automated feature learning models such as neural networks, it is often difficult to analyze or even decipher the features learnt, which are in turn constrained by the activation functions or the learning algorithm itself. For example, features that are computed via non-differentiable or non-continuous functions (such as *max*, *min*, median, etc.) cannot be learnt by gradient descent. This is why, despite the availability of feature learning algorithms, the design of feature extractors continue to be an active area of research.

We set out to develop a handwritten digit recognition system that extracts features along the following principles:

(1) *Biological basis*: This is an old but reasonably successful principle that has been widely adopted in computer vision. After all, the biological visual system is the most robust recognition system we know, and hence it pays to emulate it wherever possible. In our model, we attempt to cover as many types of features as possible that are known to be extracted by the biological system.

(2) *Linear separability*: A crucial requirement for feature extraction is to minimize the within-class variability and enhance the between-class variability [9]. A qualitative

* Corresponding author. Tel.: +65-874-2786; fax: +65-779-4580.

E-mail addresses: teowloon@comp.nus.edu.sg (L.-N. Teow), loef@comp.nus.edu.sg (K.-F. Loe).

0031-3203/02/$22.00 © 2002 Pattern Recognition Society. Published by Elsevier Science Ltd. All rights reserved.
PII: S0031-3203(01)00228-X

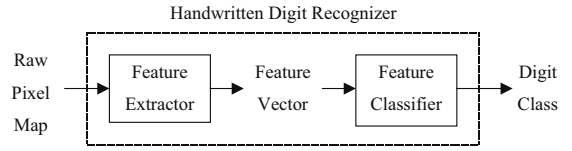

Fig. 1. General structure of a typical handwritten digit recognition system.

measure of this requirement is linear separability. By having the feature set linearly separable over the training data, we need to use only linear classifiers which are simpler, faster to train, and less prone to problems (such as overfitting and local minima) that usually plague non-linear classifiers. In handwritten digit recognition, unless the database is very small (say 1000), it is extremely difficult to achieve linear separability due to the large number of variations in writing style, stroke thickness, skew, orientation, etc. Hence, it is significant that our model has achieved this goal for a large data set of 60 000.

(3) *Clear semantics*: It is often desirable to know the *meaning* of the features either for explanatory purposes or to facilitate further analysis. By explicitly extracting well-defined features, for example, our model achieves semantic clarity.

As in traditional pattern recognition systems, our model consists of two main modules (see Fig. 1): a feature extractor that generates a feature vector from the raw pixel map, and a feature classifier that outputs the class based on the feature vector. Despite its traditional structure, our vision-based model has achieved state-of-the-art performance in handwritten digit recognition, even surpassing the current record accuracy on the MNIST data set. A significant contribution of our work is the extraction of robust features that are linearly separable over a large set of training data in a highly non-linear domain. Another contribution is a novel triowise system of combining sub-domain classifiers, which gives the best performance so far among multiclass classification schemes.

The next two sections describe the feature extraction and classification processes, respectively. Experiments are then conducted with the model on handwritten digits, and results are compared with those in other works. This paper is an enhancement and extension of the work in Ref. [10].

2. Feature extraction

A crucial step in the design of the feature extractor involves deciding what features to extract, based upon the principles outlined previously. The biological visual system is known to extract a wide variety of local spatial features, such as edges, lines, and corners of various orientations, lengths and widths [11–15]. We chose to detect edge and

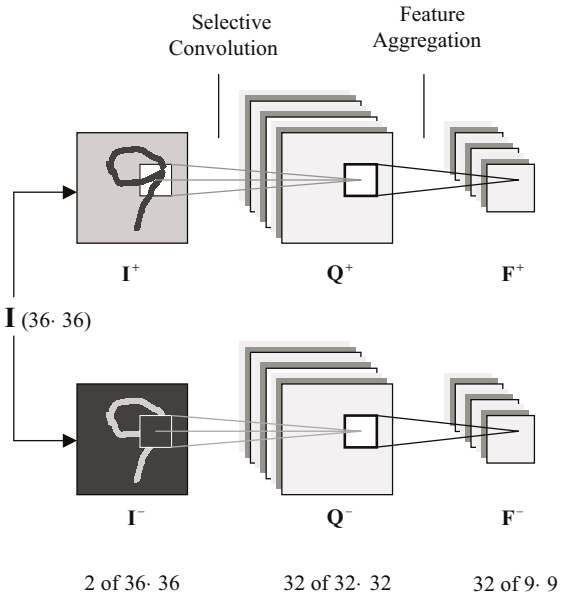

Fig. 2. The feature extraction process.

corner orientations, as these are more relevant to the domain. In addition, the visual system distinguishes between bright and dark features as evidenced by the presence of on-center and off-center receptive fields [16,17]. Hence, we chose to implement a dual-channel system as well.

In a nutshell, spatial features are extracted from the raw pixel map in a few simple steps (see Fig. 2). Details are as follows.

The raw input is a two-dimensional character pixel image denoted by $\mathbf{I}$, whereby the gray-level pixel value in position (X, Y) is given by $\mathbf{I}(X, Y)$ ranging from 0 to $MaxGrayValue$, with 0 as the background intensity. The raw input is intensity-normalized to produce $\mathbf{I}^+$ and its complement $\mathbf{I}^-$, which correspond to the on- and off-channels in the visual system [16,17], respectively.

$$\mathbf{I}^+(X, Y) = \frac{\mathbf{I}(X, Y)}{MaxGrayValue}, \tag{1}$$

$$\mathbf{I}^-(X, Y) = 1 - \mathbf{I}^+(X, Y). \tag{2}$$

We have found that a dual-channel system gives a small but significant improvement in classification accuracy compared to a single-channel system.

Selective convolution is then performed with small-size kernels. Convolution is equivalent to having local receptive fields, inspired by biological analogues in the visual system [12,13,15], whereby elementary features are detected within a small subset of inputs that are topologically "close". This allows the network to exploit the spatial topology of the image by detecting the same spatial features at different locations. We denote the jth convolution map by $\mathbf{Q}_j$, and let $\mathbf{H}_j$ denote the corresponding mask or kernel having a receptive

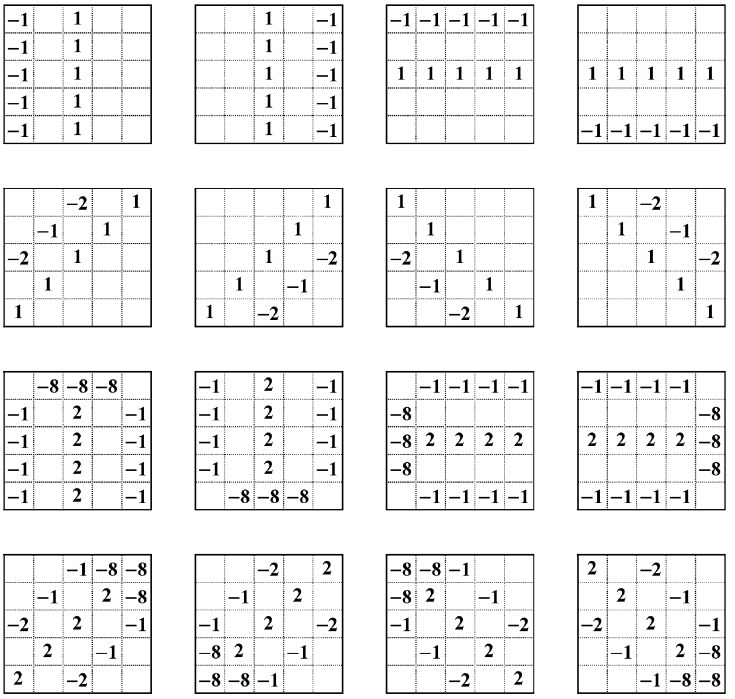

Fig. 3. The 16 mask templates used in feature extraction. The top two rows are edge detection masks while the bottom two rows are end-stop detection masks. Empty cells indicate zero values.

field of radius r (=2). Selective convolution is performed in both channels as follows:

$$\mathbf{Q}_j^{\pm}(X,Y) = \mathbf{I}^{\pm}(X,Y)\ \ell(\mathbf{G}_j^{\pm}(X,Y)), \tag{3}$$

where

$$G_j^{\pm}(X,Y) = \sum_{M=-r}^{r}\sum_{N=-r}^{r} \mathbf{H}_j(M,N)\ \mathbf{I}^{\pm}(X+M,Y+N), \tag{4}$$

$$\ell(z) = \begin{cases} z & \text{if } z > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Each convolution sum $\mathbf{G}_h^{\pm}(X,Y)$ undergoes a truncated linear function ℓ (i.e., halfwave rectification, also performed by simple cells [15]), such that the resulting feature map contains only non-negative values each indicating the strength of a feature's presence. The truncated sum is then multiplied with the central pixel value; we have found this to be a simple but very effective way of reducing the occurrence of false edges or corners. The convolution is *selective* in the sense that the presence of a feature is very much dependent on the central pixel. In practice, only one convolution pass is needed for each mask, since

$$G_j^{-}(X,Y) = \sum_{M=-r}^{r}\sum_{N=-r}^{r} \mathbf{H}_j(M,N) - \mathbf{G}_j^{+}(X,Y). \tag{6}$$

The 16 mask templates ($\mathbf{H}_j$) used are shown in Fig. 3. These mask templates are simplified models of the simple and hypercomplex receptive fields in the visual cortex [12,13,15], and serve to detect edges and end-stops of various orientations. While both simple and hypercomplex receptive fields are orientation specific, the latter has an additional inhibition region at either end. The modeling can also be done using linear combinations of Gaussians with suitable centers and radii [15], but we found it computationally more efficient to use these integer masks instead.

Applying the end-stop filters in the on-channel detects convex corners, while applying them in the off-channel detects concave corners. It is well known that removing corners or high curvature information from images degrades human recognition performance, while removing many of the edges does not [18]. Indeed, we found corners to be much more robust features than edges, and the inclusion of corner features improves the recognition accuracy significantly.

Feature aggregation, which is similar to subsampling, is now performed to reduce the number of features, which would otherwise be prohibitive. We distinguish between subsampling and feature aggregation. In subsampling, each subsampled value is usually the mean of the pixels in the corresponding window, since the objective here is to lower the resolution while maintaining as faithful to the original image as possible. In feature aggregation, on the other hand,

the objective is to detect the *presence* of a feature in each window, which should not be affected by the window size. At the same time, there should be an averaging effect among non-zero feature values, with more weight given to the larger values. To this end, we define the magnitude-weighted average (MWA):

$$\mathbf{F}_j^{\pm}(X,Y) = \sum_{M=0}^{v-1}\sum_{N=0}^{v-1} \mathbf{W}_{jXY}^{\pm}(M,N)\,\mathbf{Q}_j^{\pm}(uX+M, uY+N), \tag{7}$$

where u is the horizontal or vertical shift from one window to the next, and v is the window width. If $u < v$, then there is a window overlap of width $v-u$ pixels. As for each weight

$$\mathbf{W}_{jXY}^{\pm}(M,N) = \frac{|\mathbf{Q}_j^{\pm}(uX+M, uY+N)|^{\varphi}}{\sum_{S=0}^{v-1}\sum_{T=0}^{v-1}|\mathbf{Q}_j^{\pm}(uX+S, uY+T)|^{\varphi}}. \tag{8}$$

In our work, we set $\varphi = 1$. We have found the MWA to be more robust compared to either the simple mean, the maximum, or softmax, especially when the window size is large. We also found that it is always better to have a large degree of overlap between adjacent windows to smoothen the transition from one region to another. Like subsampling, feature aggregation confers a certain degree of local invariance to distortions and translations by reducing dependency on the features' positions.

$\mathbf{F}^+$ and $\mathbf{F}^-$ serve as the features for classification. We have found no necessity in building a hierarchy of features as done in other work [3–5,19]. The idea of a hierarchical structure that extracts increasingly complex features originated from Refs. [12,13] that suggested a hierarchical model in which each simple cell receptive field is formed by combining responses from several center-surround lateral geniculate nucleus (LGN) cells. Complex cells were viewed as the next stage of processing produced by pooling responses from a group of simple cells with overlapping receptive fields. Finally, hypercomplex cells were supposed to combine the responses of complex cells with different receptive fields.

However, as pointed out in Ref. [15], subsequent research has shown this model to be faulty. First of all, both simple and complex cells have been found which exhibit the properties of hypercomplex cells [20], i.e. the so-called hypercomplex cells are really subclasses of simple and complex cells. Secondly, it was found that orientation selectivity arises at the cortical level, and is not a function of LGN cell responses [21,22]. Finally, it was observed that many complex cells respond to visual stimuli *before* simple cells [23], sometimes even inhibiting the responses of simple cells [24]. It now appears that simple, complex cells and their subclasses represent parallel rather than hierarchical processing stages.

Moreover, given that there are so many different low-level features, the number of possible combinations would be exponential and would therefore be too large for a full hierarchy of features to be feasible. Having only a single convolution layer greatly simplifies the feature extraction process, as there is no need to consider the input configurations of subsequent convolution layers. In any case, we have found the features extracted by our model to be sufficient for excellent classification performance.

3. Feature classification

We experimented with various types of classifiers: linear discriminant systems (one-per-class, pairwise, triowise) and k-nearest neighbor classifiers (Euclidean distance, cosine similarity).

3.1. Linear discriminant systems

In our work, a linear discriminant system can either be a single-layer perceptron network trained using the perceptron learning rule [25–27], or linear support vector machines trained using quadratic programming [28–30]. While both are linear classifiers, they differ mainly by the criterion used to train them. For completeness, we shall briefly describe these classifier types in subsequent sections.

At the basic level, a linear discriminant system consists of a set of processing units, each implementing a hyperplane defined by the equation

$$f(\mathbf{x}) = \mathbf{w}\,\mathbf{x} + b, \tag{9}$$

where $\mathbf{w}$ is a weight vector, $\mathbf{x}$ is the feature vector concatenating the values of both $\mathbf{F}^+$ and $\mathbf{F}^-$, and b is a bias value. An activation function g is further applied to give

$$p = g(f(\mathbf{x})), \tag{10}$$

where p is the unit's output value, and g can be one of the following:

$$g(z) = \frac{1}{1+\mathrm{e}^{-z}}, \quad \text{(sigmoidal/logistic function)} \tag{11}$$

$$g(z) = \begin{cases} -1 & \text{if } z \leqslant -1, \\ z & \text{if } -1 < z < 1, \\ 1 & \text{if } z \geqslant 1. \end{cases} \quad \text{(semi-linear function)} \tag{12}$$

The sigmoidal or logistic function is most commonly used for the perceptron network, as it allows the outputs to be interpreted as probabilities. On the other hand, a natural choice for the support vector machine, given its formulation, is the semi-linear function.

3.1.1. Multiclass classification schemes

We consider three different schemes in multiclass classification. The classification scheme in turn determines the size and architecture of the system.

The first scheme, which is the simplest and most popularly used, consists of one unit for each class (one-per-class). Each unit is trained to separate its corresponding class from the rest of the classes. In other words, unit $\langle A \rangle$ is trained to output 1 for patterns of class A, and 0 (or -1) for patterns of

other classes. During classification, the unit with the largest output value indicates the class of the character, that is,

$$A^* = \underset{A}{\operatorname{argmax}}\, p^{\langle A\rangle}. \tag{13}$$

The second scheme, less commonly used, is known as *pairwise* classification [31–33]. The objective here is to partition the problem into easier subproblems involving only two classes. Each unit is trained to separate a class from one other class. Hence, unit $\langle AB\rangle$ is trained to output 1 for patterns of class A, 0 (or -1) for patterns of class B, and is not trained on other patterns. During classification, the winning class A^* is determined by voting:

$$\text{Soft voting}: \; A^* = \underset{A}{\operatorname{argmax}} \sum_{B\neq A} (p^{\langle AB\rangle} - p^{\langle BA\rangle}), \tag{14}$$

$$\text{Hard voting}: \; A^* = \underset{A}{\operatorname{argmax}} \sum_{B\neq A} (\delta(p^{\langle AB\rangle}) - \delta(p^{\langle BA\rangle})), \tag{15}$$

where δ is a threshold function

$$\delta(z) = \begin{cases} 1 & \text{if } z > \Phi, \\ -1 & \text{if } z \leqslant \Phi. \end{cases} \tag{16}$$

Here, Φ is a threshold value, which is equal to 0.5 for the perceptron network, and equal to 0 for the support vector machines. We have tried other ways of combining the outputs as suggested in the literature [31–33], but have found voting to give the best results consistently.

The third scheme, which to our knowledge has never been reported in the literature, extends pairwise classification to *triowise* classification, i.e. three classes per unit. Each unit is trained to separate a class from two other classes. Hence, unit $\langle ABC\rangle$ is trained to output 1 for patterns of class A, 0 (or -1) for patterns of either class B or class C, and is not trained on other patterns. During classification, the winning class A^* is again determined by voting:

Soft voting :

$$A^* = \underset{A}{\operatorname{argmax}} \sum_{\substack{B\neq A \\ C\neq A \\ B\neq C}} (p^{\langle ABC\rangle} - p^{\langle BAC\rangle} - p^{\langle CAB\rangle}), \tag{17}$$

Hard voting :

$$A^* = \underset{A}{\operatorname{argmax}} \sum_{\substack{B\neq A \\ C\neq A \\ B\neq C}} (\delta(p^{\langle ABC\rangle}) - \delta(p^{\langle BAC\rangle}) - \delta(p^{\langle CAB\rangle})). \tag{18}$$

Different classification schemes require different training strategies. We have a set of training samples $\{(\mathbf{x}_i, y_i) \mid i = 1,\ldots,n\}$, where $\mathbf{x}_i$ is a feature vector and $y_i \in \{0,\ldots,9\}$ is the class label. Let $A, B, C \in \{0,\ldots,9\}$, where $A \neq B$, $B \neq C$, $C \neq A$. In the one-per-class classification scheme, unit $\langle A\rangle$ is trained with all samples regardless of the class. In the pairwise classification scheme, unit $\langle AB\rangle$ is trained only with samples where $y_i \in \{A, B\}$. Finally, in the triowise classification scheme, unit $\langle ABC\rangle$ is trained only with samples where $y_i \in \{A, B, C\}$.

3.1.2. Perceptron learning

The perceptron learning algorithm based on gradient descent is well documented (see Refs. [25–27]) and shall not be detailed here. Suffice to say that we employ the cross-entropy cost function [25,34,35] for training, and update the weights and biases using a momentum term upon presentation of each training pattern. Furthermore, a small positive constant (say 0.01) is added to each feature value in $\mathbf{x}_i$, to overcome the problem in perceptron learning whereby connections with zero inputs do not learn. Depending on the classification scheme, the target value q_i for perceptron $\langle A\rangle$, $\langle AB\rangle$ or $\langle ABC\rangle$ is

$$q_i = \begin{cases} 1 \text{ if } y_l = A, \\ 0 \text{ otherwise.} \end{cases} \tag{19}$$

During training, an epoch constitutes a pass through all patterns in the training set. At each epoch, we further compute the average absolute error E_{AVE} and the maximum absolute error E_{MAX}. Specifically, E_{AVE} is the average absolute difference between the output and target values (i.e. $|q_i - p_i|$) over all outputs and training patterns, while E_{MAX} is the maximum of these differences. Training stops either when $E_{MAX} < E_\lambda$ (where E_λ is a predefined small positive value), or when a maximum number of epochs has been reached.

3.1.3. Linear support vector machines

Like the perceptron, the support vector machine (SVM) also performs discrimination with a separating hyperplane. However, whereas the perceptron is trained to optimize a cross-entropy cost function, the SVM is trained to maximize the distance of the hyperplane from the class boundaries (i.e., maximize the interclass margin width). In addition, unlike the perceptron which is trained by gradient descent, the SVM is trained using quadratic programming (which is beyond the scope of this paper). More importantly, the SVM has been shown to give good generalization performance [36,37].

We consider only linear SVMs trained on linearly separable data (as will be shown to be the case here). Depending on the classification scheme, the hyperplane $f(\mathbf{x})$ for SVM $\langle A\rangle$, $\langle AB\rangle$ or $\langle ABC\rangle$ is trained such that

$$f(\mathbf{x}_i) \geqslant +1 \quad \text{for } y_i = A, \tag{20}$$

$$f(\mathbf{x}_i) \leqslant -1 \quad \text{for } y_i \neq A. \tag{21}$$

The interclass margin width is correspondingly defined as

$$\min_{i|y_i=A} \frac{f(\mathbf{x}_i)}{|\mathbf{w}|} - \max_{i|y_i\neq A} \frac{f(\mathbf{x}_i)}{|\mathbf{w}|}, \tag{22}$$

where $|\mathbf{w}| = \sqrt{\mathbf{w}\cdot\mathbf{w}}$ is the weight magnitude.

It can be shown easily that the margin width is inversely proportional to the weight magnitude. For each SVM, we maximize the margin width indirectly by minimizing the weight magnitude. This can be formulated as a quadratic programming problem. The solution is expressed in terms of a set of *support vectors*, which are those samples that lie closest to the hyperplane and define the interclass boundaries.

3.2. k-nearest neighbor classifiers

For comparison, we also tried k-nearest neighbor classification, whereby we compute the distance or similarity between the features of a test sample and the features of every training sample. The class of the majority among the k nearest training samples is deemed as the class of the test sample. We experimented with both the standard Euclidean distance as well as the cosine similarity measure (equivalently the normalized dot product of two vectors):

$$\text{Euclidean distance} = |\mathbf{x}_{test} - \mathbf{x}_{train}|, \tag{23}$$

$$\text{Cosine similarity} = \frac{\mathbf{x}_{test}\cdot\mathbf{x}_{train}}{|\mathbf{x}_{test}|\cdot|\mathbf{x}_{train}|}, \tag{24}$$

where $|\mathbf{z}| = \sqrt{\mathbf{z}\cdot\mathbf{z}}$ is the vector norm.

4. Experiments on handwritten digits

We tested our model on the MNIST database of handwritten digits, since many methods have been tested on this database [4,5] and hence it would serve as a good basis for comparison. This database, which can be downloaded from the AT& T Laboratories' research web-site, consists of 60 000 training samples and 10 000 test samples, and was constructed from NIST's Special Database 1 and Special Database 3 [38]. The original binary images from NIST were size-normalized to fit in a 20×20 pixel box while preserving their aspect ratio. Anti-aliasing during the normalization resulted in gray-level images, which were then centered by positioning the center of mass of the pixels at the center of a 28×28 field. In our experiments, we add an empty border of 4-pixel width, giving a 36×36 input image.

For the feature aggregation, we use 8×8 receptive fields with overlap 5-pixel wide (i.e., $u = 3$ and $v = 8$). There are a total of 32 feature maps, each of size 9×9 after feature aggregation. Hence, the feature vector $\mathbf{x}$ has $32\times9\times9=2592$ values.

4.1. Preliminary experiments

For the single-layer perceptron network, we use a learning rate of 0.5, a momentum value of 0.3, and set the convergence threshold $E_\lambda = 0.01$. As for initializing the weights and biases, a commonly used range of initial values is $(-1/\sqrt{D}, 1/\sqrt{D})$ where D is the number of inputs to the unit. For each output unit, the total number of input connections plus the bias is equal to the number of feature values plus one, i.e., 2593. Hence, each weight and bias is initialized from the range $(-0.02, 0.02)$. Training proceeds for a maximum of 300 epochs.

For the linear support vector machines, we use the SVM^{light} software system by Joachims [39]. This system is an implementation of the SVM for pattern recognition, and is tailored for efficiently handling large data sets. Its training algorithm, based on quadratic programming, incorporates several optimization techniques such as decomposition and caching. The SVM^{light} system requires the loading of all training samples into the main memory for learning. Hence, we could not implement the one-per-class scheme due to insufficient memory resources (consider $60\,000 \times 2592$ double-precision floating point numbers for the features plus 60 000 bytes for the class labels). Instead, we implement only pairwise and triowise SVMs, since these schemes require only a subset of the training set for each SVM. Nonetheless, we shall see later that this is not a disadvantage as pairwise and triowise classifiers perform much better than one-per-class classifiers.

As for k-nearest neighbor classification, we conduct experiments with $k \in \{1, 3, 5, 7, 9\}$, and report results for the best performing k.

The classification error rates with zero rejection are shown in Table 1, and the number of epochs to convergence

Table 1
Error rates for various classifiers on the MNIST data set

Feature classifier	Scheme	Voting option	Train error (%) (60 000 samples)	Test error (%) (10 000 samples)
Perceptron network	1-per-class	—	0.00	2.14
	pairwise	Hard	0.00	0.88
		Soft	0.00	0.87
	Triowise	Hard	0.00	0.72
		Soft	0.00	0.72
Linear SVMs	Pairwise	Hard	0.00	0.98
		Soft	0.00	0.82
	Triowise	Hard	0.00	0.74
		Soft	0.00	0.72
k-nearest neighbor	Euclidean Distance	—	0.00	1.39 ($k = 3$)
	Cosine Similarity	—	0.00	1.09 ($k = 3$)

Table 2
Number of epochs to convergence for perceptron training

Scheme	# Units	# Epochs
1-per-class	10	281
Pairwise	90	57
Triowise	360	147

for perceptron training is given in Table 2. All the linear discriminant systems achieve zero error (i.e., perfect classification) on the training samples—empirical proof that the features are linearly separable over the training data. The best preliminary result on the test data (0.72% error) is obtained using triowise classification with soft voting in either the single-layer perceptron network or the linear support vector machines.

For the linear discriminant systems, triowise classification with soft voting gives the best performance. This is to be expected since triowise classification employs the most discriminants, while soft voting makes fuller use of the information available.

The convergence behavior of the perceptron network is interesting; while E_{AVE} drops quite early in the training and slowly tapers off, E_{MAX} remains at 1 almost throughout the training and then drops sharply only at the last few epochs. Apparently, much of the error is due to only a few training samples.

The digits in this database already contain a fair amount of variation in skew and orientation. If these digits were to be deslanted, for example, the test error rates should be even lower. In fact, we shall find that this is indeed the case, as we explore this possibility in Section 4.2.

4.2. Experiments on deslanted images

To improve classification performance, we perform further preprocessing of the digit images prior to feature extraction. The objective is to reduce the amount of pattern variations within each class. One common way of doing this is by deslanting the digit images, which is performed as follows. First, the least-squares regression line passing through the centroid of the digit image (i.e., the center of mass of the pixels) is computed, weighting each pixel by its gray-level value. The digit image is then skewed about the centroid such that the regression line becomes vertical.

More formally, the equation for the weighted regression line is

$$X = \bar{X} + m \cdot (Y - \bar{Y}), \tag{25}$$

where X and Y are the pixel coordinates and

$$\bar{X} = \frac{\sum_{X,Y} X\, \mathbf{I}(X,Y)}{\sum_{X,Y} \mathbf{I}(X,Y)}, \tag{26}$$

Table 3
Error rates for various classifiers on deslanted MNIST digits

Feature classifier	Scheme	Voting option	Train error (%) (60 000 samples)	Test error (%) (10 000 samples)
Perceptron network	Pairwise	Hard	0.00	0.81
		Soft	0.00	0.73
	Triowise	Hard	0.00	0.63
		Soft	0.00	0.62
Linear SVMs	Pairwise	Hard	0.00	0.69
		Soft	0.00	0.68
	Triowise	Hard	0.00	0.65
		Soft	0.00	0.59

$$\bar{Y} = \frac{\sum_{X,Y} Y\, \mathbf{I}(X,Y)}{\sum_{X,Y} \mathbf{I}(X,Y)}, \tag{27}$$

$$m = \frac{\sum_{X,Y} X\, Y\, \mathbf{I}(X,Y) - \bar{X}\, \bar{Y} \sum_{X,Y} \mathbf{I}(X,Y)}{\sum_{X,Y} Y^2\, \mathbf{I}(X,Y) - \bar{Y}^2 \sum_{X,Y} \mathbf{I}(X,Y)}. \tag{28}$$

To preserve the position of the centroid at the center of the pixel map, skewing is performed about the centroid with linear interpolation:

$$\mathbf{I}_{deslanted}(X,Y) = (\lceil X' \rceil - X')\, \mathbf{I}_{original}(\lfloor X' \rfloor, Y) + (X' - \lfloor X' \rfloor)\, \mathbf{I}_{original}(\lceil X' \rceil, Y), \tag{29}$$

where $\lfloor\ \rfloor$ and $\lceil\ \rceil$ denote the integer floor and ceiling respectively, and

$$X' = X + m \cdot (Y - \bar{Y}). \tag{30}$$

We conduct the same experiments as before, except this time all digit images are deslanted in an additional preprocessing stage. In addition, we try only the pairwise and triowise linear discriminant systems as these have been shown to give superior results in the preliminary experiments (Section 4.1).

The error rates are shown in Table 3. In particular, by using triowise linear SVMs (with soft voting) on deslanted digit images, we have obtained the lowest ever error rate of 0.59% on the MNIST data set (the record in the literature was 0.7% [5]). Fig. 4 shows the 59 misclassified characters, while Table 4 shows the corresponding confusion matrix, from which we can tell the frequency with which a character is misclassified as another. For example, '2' is most often confused with '7', likewise '6' with '0', and '9' with '4'.

4.3. Comparison with other models

Our model is somewhat similar to the convolutional neural network [3–5], which is among the best classifiers in the domain of handwritten digit recognition. The convolutional network model is characterized by local receptive fields, weight sharing, subsampling, a hierarchical structure

and automated discovery of features through backpropagation. The first version of this model, LeNet-1, is a relatively small and simple network. Later incarnations of the model, such as LeNet-4 and LeNet-5, are much larger and more complex.

We also compare our model with the tangent distance classifier [40] as well as the Virtual Support Vector Machine [41]. In Table 5, we reproduce the best results reported in Ref. [5] as well as our best result for comparison. All results are based on the MNIST data set.

Clearly, based on test error alone, our model (i.e., vision-based features from deslanted images + triowise linear SVMs with soft voting) outperforms all the other models. Moreover, there is no evidence that the other models achieve perfect classification on the training data as our model does. Indeed, we cannot apply boosting [42] to our model precisely because it has no training errors. Despite this, and contrary to conventional wisdom, overfitting is not a problem as evidenced by our model's excellent test performance. We believe this is because we use a linear discriminant system instead of a non-linear one.

Fig. 4. The 59 characters that are misclassified by the triowise linear SVMs with soft voting.

Table 5
Classification error rates for various models on the MNIST data set[a]

Classifier model	Test error (%)
LeNet-4	1.10
LeNet-4, boosted [distort]	0.70
LeNet-5	0.95
LeNet-5 [distort]	0.80
Tangent distance	1.10
Virtual SVM	0.80
⟨Our model⟩ [deslant]	0.59

[a][distort] indicates that artificially distorted sampleshave been added to the training set, while [deslant] indicatesthat the model has been trained and tested on deslanted digitimages.

5. Conclusion

We have developed a vision-based handwritten digit recognition system, which extracts features that are biologically plausible, linearly separable and semantically clear. With good features, we need only a relatively simple feature classifier that trains fast and gives excellent classification performance.

Possible future enhancements include yet further preprocessing for increased normalization, adding other types of features, and extension to multi-resolution modeling. The last is especially interesting as it allows the extraction of features at multiple spatial scales, which is also performed by the biological visual system [11,14].

Although we have so far only experimented on handwritten digits, our model is generic enough to be applied to the other types of characters and images. Indeed, it would be interesting to try our model on the alphabet, punctuation, Kanji or Chinese characters, etc., both handwritten and typewritten. We can also apply our model to the recognition of both geometric and natural shapes. All these remain for future work.

Table 4
Confusion matrix for the recognition of deslanted images by triowise linear SVMs with soft voting

Class	0	1	2	3	4	5	6	7	8	9	#errors
0	977	0	0	0	0	0	2	1	0	0	3
1	0	1134	1	0	0	0	0	0	0	0	1
2	1	0	1023	1	1	0	0	5	1	0	9
3	0	0	1	1005	0	4	0	0	0	0	5
4	0	0	0	0	975	0	1	1	1	4	7
5	1	0	0	3	0	887	1	0	0	0	5
6	5	2	1	0	1	1	948	0	0	0	10
7	0	1	2	1	0	0	0	1022	0	2	6
8	0	0	1	0	0	1	0	0	972	0	2
9	0	0	0	0	7	2	0	1	1	998	11

Each row corresponds to a target class, while each column corresponds to a recognition output class.

References

[1] A.K. Jain, D. Zonker, Representation and recognition of handwritten digits using deformable templates, IEEE Trans. Pattern Anal. Mac. Intell. 19 (12) (1997) 1386–1391.

[2] H. Kang, S. Lee, Combining classifiers based on minimization of a bayes error rate, Proceedings of the Fifth International Conference on Document Analysis and Recognition (ICDAR'99), 1999, pp. 398–401.

[3] Y. Lecun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, L. Jackel, Handwritten digit recognition with a backpropagation network, in: D. Touretzky (Ed.), Advances in Neural Information Processing Systems, Vol. 2, Morgan Kaufmann, San Mateo, CA, 1990, pp. 396–404.

[4] Y. Lecun, L.D. Jackel, L. Bottou, C. Cortes, J.S. Denker, H. Drucker, I. Guyon, U.A. Muller, E. Sackinger, P. Simard, V. Vapnik, Learning algorithms for classification: a comparison on handwritten digit recognition, in: J.H. Oh, C. Kwon, S. Cho (Eds.), Neural Networks: The Statistical Mechanics Perspectives, World Scientific, Singapore, 1995, pp. 261–276.

[5] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.

[6] K.M. Mohiuddin, J. Mao, A comparative study of different classifiers for handprinted character recognition, in: E.S. Gelsema, L.N. Kanal (Eds.), Pattern Recognition in Practice IV, Elsevier, Amsterdam, 1994, pp. 437–448.

[7] R. Plamondon, S.N. Srihari, On-line and off-line handwriting recognition: a comprehensive survey, IEEE Trans. Pattern Anal. Mac. Intell. 22 (1) (2000) 63–84.

[8] A.A. Verikas, M.I. Bachauskene, A.V. Lashas, Investigation of a number of character recognition algorithms, in: L. Shapiro, A. Rosenfeld (Eds.), Computer Vision and Image Processing, Academic Press, New York, 1992, pp. 231–243.

[9] P.A. Devijver, J. Kittler, Pattern Recognition: A Statistical Approach, Prentice-Hall, London, 1982.

[10] L.-N. Teow, K.-F. Loe, Handwritten digit recognition with a novel vision model that extracts linearly separable features, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-2000), Vol. 2, 2000, pp. 76–81.

[11] S. Coren, L.M. Ward, J.T. Enns, Sensation and Perception, 4th Edition, Harcourt Brace, New York, 1994.

[12] D.H. Hubel, T.N. Wiesel, Receptive fields, binocular interaction and functional architecture in the cat's visual cortex, J. Physiol. 160 (1962) 106–154.

[13] D.H. Hubel, T.N. Wiesel, Receptive fields and functional architecture of monkey striate cortex, J. Physiol. 195 (1968) 215–243.

[14] R. Sekuler, R. Blake, Perception, 3rd Edition, McGraw-Hill, New York, 1994.

[15] H.R. Wilson, D. Levi, L. Maffei, J. Rovamo, R. DeValois, The perception of form: retina to striate cortex, in: L. Spillmann, J.S. Werner (Eds.), Visual Perception: The Neurophysiological Foundations, Academic Press, New York, 1990, pp. 231–272 (Chapter 10).

[16] A. Fiorentini, G. Baumgartner, S. Magnussen, P.H. Schiller, J.P. Thomas, The perception of brightness and darkness: relations to neuronal receptive fields, in: L. Spillmann, J.S. Werner (Eds.), Visual Perception: The Neurophysiological Foundations, Academic Press, New York, 1990, pp. 129–161 (Chapter 7).

[17] P.H. Schiller, J.H. Sandell, J.H.R. Maunsell, Functions of the ON and OFF channels of the visual systems, Nature 322 (1986) 824–825.

[18] I. Biedermann, Recognition-by-components: a theory of human image understanding, Psychol. Rev. 94 (2) (1987) 115–147.

[19] K. Fukushima, S. Miyake, Neocognitron: a new algorithm for pattern recognition tolerant of deformations and shifts in position, Pattern Recognition 15 (1982) 455–469.

[20] B. Dreher, Hypercomplex cells in the cat's striate cortex, Invest. Ophthalmol. 11 (1972) 355–356.

[21] O.D. Creutzfeldt, M. Ito, Functional synaptic organization of primary visual cortex neurons in the cat, Exp. Brain Res. 6 (1968) 324–352.

[22] A.M. Sillito, Inhibitory circuits and orientation selectivity in the visual cortex, in: D. Rose, V.G. Dobson (Eds.), Models of the Visual Cortex, Wiley, New York, 1985, pp. 396–407.

[23] K.P. Hoffmann, J. Stone, Conduction velocity of afferents to cat visual cortex: a correlation with cortical receptive field properties, Brain Res. 32 (1971) 460–466.

[24] L. Maffei, Complex cells control simple cells, in: D. Rose, V.G. Dobson (Eds.), Models of the Visual Cortex, Wiley, New York, 1985, pp. 334–340.

[25] J. Hertz, A. Krogh, R.G. Palmer, Introduction to the Theory of Neural Computation, Addison-Wesley, Reading, MA, 1991.

[26] F. Rosenblatt, Principles of Neurodynamics, Spartan, New York, 1962.

[27] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representations by error propagation, in: D.E. Rumelhart, J.L. McClelland et al. (Eds.), Parallel Distributed Processing: Explorations in the Micro-structure of Cognition, Vol. 1: Foundations, MIT Press, Cambridge, MA, 1986, pp. 318–362 (Chapter 8).

[28] C.J.C. Burges, A tutorial on support vector machines for pattern recognition, Data Mining Knowledge Discovery 2 (1998) 121–167.

[29] C. Cortes, V. Vapnik, Support vector networks, Mac. Learning 20 (1995) 273–297.

[30] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.

[31] T. Hastie, R. Tibshirani, Classification by pairwise coupling. Technical Report, University of Toronto, 1996.

[32] M. Moreira, E. Mayoraz, Improved pairwise coupling with correcting classifiers, Proceedings of the Tenth European Conference on Machine Learning (ECML '98), Springer, Berlin, 1998, pp. 160–171.

[33] D. Price, S. Knerr, L. Personnaz, G. Dreyfus, Pairwise neural network classifiers with probabilistic outputs, in: G. Tesauro, D. Touretzky, T. Leen (Eds.), Advances in Neural Information Processing Systems, Vol. 7, MIT Press, Cambridge, MA, 1995, pp. 1109–1116.

[34] E.B. Baum, F. Wilczek, Supervised learning of probability distributions by neural networks, in: D.Z. Anderson (Ed.), Neural Information Processing Systems, 1988, pp. 52–61.

[35] G.E. Hinton, Connectionist learning procedures, Artif. Intell. 40 (1989) 185–234.

[36] B. Scholkopf, K. Sung, C. Burges, F. Girosi, P. Nigogi, T. Poggio, V. Vapnik, Comparing support vector machines with Gaussian kernels to radial basis function classifiers, IEEE Trans. Signal Process. 45 (1997) 2758–2765.

[37] Y. Yang, X. Liu, A re-examination of text categorization methods, Proceedings of the 22nd International Conference on R& D in Information Retrieval (SIGIR'99), 1999, pp. 42–49.
[38] P.J. Grother, NIST Special Database 19—Handprinted Forms and Characters Database, National Institute of Standards and Technology, 1995.
[39] T. Joachims, Making large-scale SVM learning practical, in: B. Scholkopf, C.J.C. Burges, A.J. Smola (Eds.), Advances in Kernel Methods—Support Vector Learning, MIT Press, Cambridge, MA, 1999.
[40] P. Simard, Y. LeCun, J. Denker, Efficient pattern recognition using a new transformation distance, in: S.J. Hanson, J.D. Cowan, C.L. Giles (Eds.), Advances in Neural Information Processing Systems, Vol. 5, Morgan Kaufmann, San Mateo, CA, 1993, pp. 50–58.
[41] C.J.C. Burges, Scholkopf, Improving the accuracy and speed of support vector machines, in: M. Mozer, M. Jordan, T. Petsche (Eds.), Advances in Neural Information Processing Systems, Vol. 9, 1997, pp. 375–381.
[42] H. Drucker, R. Schapire, P. Simard, Improving performance in neural networks using a boosting algorithm, in: S.J. Hanson, J.D. Cowan, C.L. Giles (Eds.), Advances in Neural Information Processing Systems, Vol. 5, Morgan Kaufmann, San Mateo, CA, 1993, pp. 42–49.

About the Author—LOO-NIN TEOW is a doctoral student at the National University of Singapore, having obtained his B.Sc. and M.Sc. degrees in Computer Science from the same university in 1992 and 1997, respectively. Prior to his doctoral studies, he worked for 6 years at Kent Ridge Digital Labs, a research institute in Singapore; his last-held appointment there being a Research Associate. His areas of interest include pattern classification, machine learning, computer vision, and uncertainty reasoning.

About the Author—KIA-FOCK LOE is an associate professor in the Department of Computer Science at National University of Singapore. He received his Bachelor Degree and M.Sc. from Nanyang University in 1973 and 1977, respectively. He completed his Doctorate of Science from Tokyo University in 1985. His research interests are neural network, machine learning, pattern recongnition and geometric modeling.