Off-line Handwriting Recognition by Recurrent Error Propagation Networks

A.W.Senior* F.Fallside

Cambridge University Engineering Department
Trumpington Street,
Cambridge,
CB2 1PZ.

Abstract

Recent years have seen an upsurge of interest in computer handwriting recognition as a means of making computers accessible to a wider range of people. A complete system for off-line, automatic recognition of handwriting is described, which takes word images scanned from a handwritten page and produces word-level output. Normalisation and preprocessing methods are described and details of the recurrent error propagation network and Viterbi decoder used for recognition are given. Results are reported and compared with those presented by researchers using other methods.

1 Introduction

As computers become more powerful, and take a greater rôle in everyday life, the search continues for ways of integrating them into workplaces by making them conform to existing conventions of human communication. Though automatic speech recognition holds out some hope as a natural method of human to computer communication, for many applications it is unsuited and some of its promises are slow to be fulfilled. To fill these gaps, much interest is being shown in handwriting recognition, which offers applications as a direct input medium and in automatic document processing. Accordingly it is convenient to divide handwriting recognition systems into *on-line*, in which the writing is input directly via an electronic pen, and *off-line* in which words are written normally on paper and subsequently input using an optical scanner. This paper investigates the latter, more general, problem in which the data is in the form of a noisy image written with wide, overlapping pen strokes, rather than a time-ordered sequence of precisely known pen co-ordinates.

So far, most researchers have concentrated on on-line recognition (now seeing commercial application in pen computers) or the recognition of isolated characters, written separately and usually in capitals (as used in automatic postcode readers). Božinović and Srihari [1, 2] have described a system for off-line cursive script word recognition which uses a feature-based approach, segmenting the word at potential character boundaries and decoding words by

*E-mail: aws@eng.cam.ac.uk

a probabilistic method. Edelman, Ullman and Flash [3] describe a more recent system, based on 'alignment of letter prototypes', which seeks to parametrise the strokes as splines and compare these with letter prototypes, allowing affine transformations of the models.

Here we have used a recurrent error propagation network to tackle word recognition, a technique which avoids the difficult problem of finding a valid segmentation of the word into letters, and naturally fits a dynamic programming procedure which constrains the search to a lexicon of permitted words. Real, scanned handwritten data is used in a whole word, single writer system. Earlier authors who have studied the off-line problem have in fact used data collected from a digitising tablet.

2 Database Used

The images used were written by a single writer, scanned on a 300dpi, 8 bit flatbed scanner and transferred to a workstation which was used for all of the subsequent processing. Words are written surrounded by white space so that a simple program can search for them before extraction into separate files. Figure 2 shows one such word.

For this study, the corpus chosen was the set of numbers 'one' to 'twenty', tens from 'thirty' to 'hundred' plus 'thousand', 'million' and 'zero' all written in words. These thirty-one words were chosen because, though constituting a limited vocabulary, they might be considered to form a useful corpus for an application such as cheque amount verification. Seven exemplars of each of these words were taken: three to serve as a training set and four as test data. Subsequently to achieve consistent training of the network, a validation set was collected which consisted of three exemplars of each word.

3 Preprocessing

To prepare the data for recognition by a network, extensive preprocessing is required to normalise the data and present it in an appropriate form. A number of operations are involved in this sifting process. Figure 1 illustrates the whole process for which each operation is described below.

3.1 Histograms

An important tool used throughout the preprocessing is that of the density histogram. These are formed by counting the number of black pixels in each vertical or horizontal line in the image, and plotting these counts as histograms which are used to identify the baselines of the word. (See figure 2.)

3.2 Baseline Detection

Among the most fundamental features extracted from the raw image data are the base lines (the lines which run along the top and bottom of a lower case '*o*', above an '*l*' or below a '*y*'). It is with these lines that one identifies the ascenders and descenders so important in determining word shape. We shall

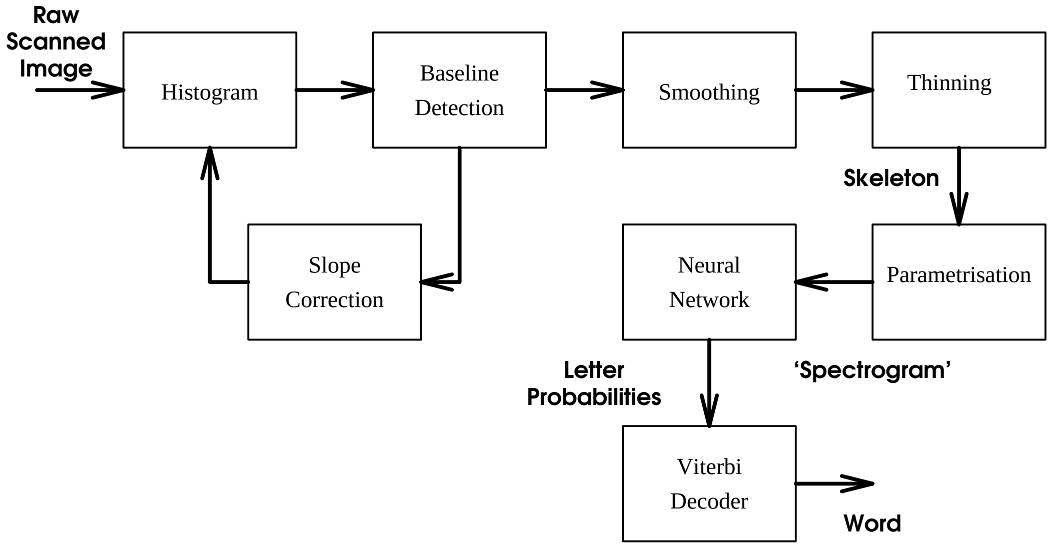


Figure 1: Schematic of the Recognition System

refer to these as the half, base, upper and lower lines, with a centre line midway between the half and base lines, as shown in figure 2.

The heuristic used for base line detection, is :

1. Calculate the vertical density histogram.

2. Reject that part of an image likely to be a descender.

3. Find the lowest pixel in each vertical scan line.

4. Carry out moment analysis to find the line of best fit.

5. Reject all outliers, and calculate the new line of best fit, which is considered to be the base line of the character.

3.3 Slope Correction

Given the best estimate of the base line, we can attempt to straighten the writing to give it a horizontal baseline. The image is straightened by application of a shear transform parallel to the y axis. (See figure 3b.) Now, we can re-calculate the base line (using a more accurate histogram) and the half line. These two lines are fundamental to the normalisation used in subsequent processes since the distance between the two is taken to be the character height, and for a given writer assumed to be proportional to the character width.

Figure 2: Histograms and Half, Centre and Base Lines.

3.4 Thinning

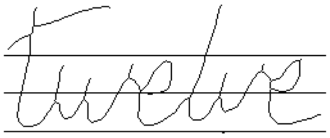
Having de-skewed and smoothed the image (by convolution with a 2-dimensional filter), it is thresholded to leave every pixel in one of two states. Then, an iterative thinning algorithm is applied to reduce the lines in the writing to one-pixel width so that the strokes can be followed later. Algorithms due to Gonzalez and Wintz [4] and Davies [5] were used to generate lines one pixel wide. (See figure 3c.)

4 Recognition

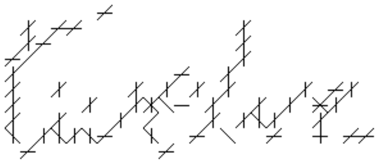
Having reduced the image to a standard form, which highlights invariants of the words and suppresses irrelevant variations, we come to the main task — that of actually extracting the word information from the image. As described earlier, a number of approaches have been taken to this problem, but the one used here is that of error propagation networks, using variations of the basic backpropagation algorithm [6] to learn the shapes of letters and to give probabilistic estimates of letter and word identities. We have investigated two architectures, but here we shall only describe the more successful recurrent error propagation network whose design as a time-invariant pattern recogniser makes it suitable for this application, with the the x axis substituted for the time axis.

(a) Initial image

(b) Slope corrected



(c) Thinned



(d) The parametrised data

Figure 3: Successive stages in the preprocessing.

4.1 Parametrisation

Having chosen this recognition method, the preprocessed image needs to be parametrised in an appropriate manner for input to the network. The parametrisation scheme chosen is as follows:

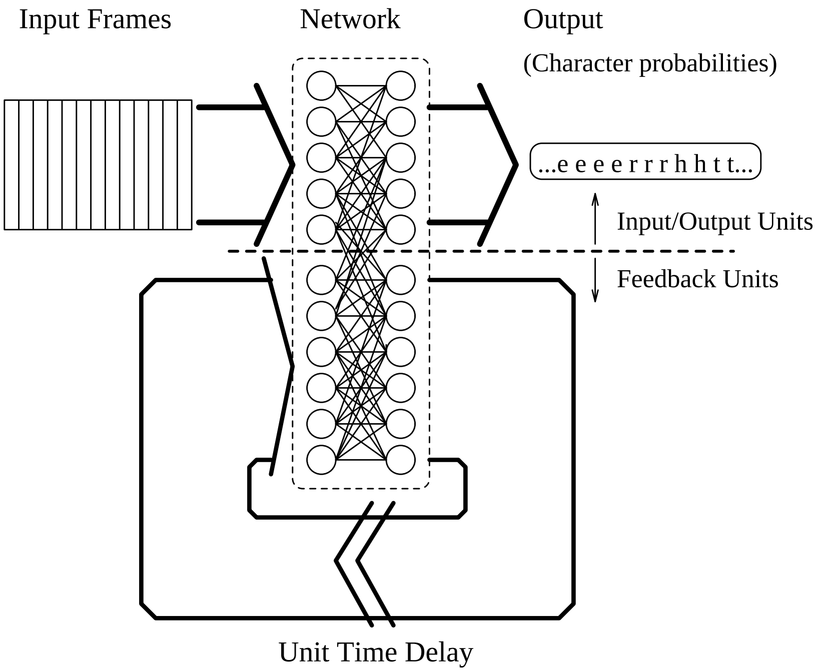
We first divide the area covered by the word into rectangles. There are sixteen horizontal rows and a variable number of vertical frames of a width proportional to the estimated character width. Thus long words have more frames than short words, but, a given character will always occupy approximately the same number. For each of these rectangles, we allocate four bins representing line angles (vertical, horizontal, and the lines 45 degrees from these.) Given this framework, we coarse code the lines of the skeleton image:

Working from left to right in the thinned image, we take each stroke (a stroke being taken to mean a one-pixel wide line between junctions or end points) and divide it into equal length segments (the length being determined by the character size) for which the centroid position and angle are calculated. We now 'fill' the box associated with this segment's (x, y, θ) values. Segments which are not perfectly aligned with the angles of the bins, are coarse coded — that is they contribute to the two bins representing the closest orientations.

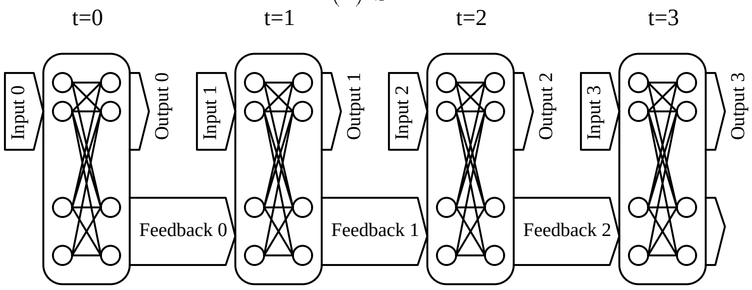
Figure 3d shows the input pattern schematically. In each square, a line indicates the presence of a line segment at approximately that angle in that bin. Because of the coarse coding, some line segments contribute to two bins and this is seen on the '*l*' stroke which is between the vertical and 45 degrees so both these lines are shown in the corresponding boxes in Figure 3d.

4.2 Recurrent Networks

Recurrent error propagation networks have been successfully applied to speech recognition and other dynamic problems which treat a time-varying signal [7, 8]. A recurrent network is well suited to time-invariant pattern recognition because the same processing is performed on each section of the input



(a) Schematic



(b) Net unfolded for training

Figure 4: Recurrent Error Propagation Network

stream, but hidden units are available to encode context information about the preceding signal.

The network is in the form of a single layer of standard perceptrons with sigmoid activation functions (as described by Rumelhart, Hinton and Williams [6]). However some of the output units are connected one-to-one to some of the input units with a unit time-delay. (See figure 4a.) The remaining input units accept the parametrised input described above, one frame at a time, and the 27 output units give letter probabilities for the 26 lower case letters plus a space character. During the forward pass, successive frames of input data are presented to the network and results are fed back through the time delay, while letter probabilities are read off from the other outputs. To allow the network to assimilate context information, a delay of several frames is permitted before reading off the probabilities.

To be able to obtain the correct outputs from the network, we have to determine the correct values of the internal parameters ('weights') by a training procedure. Training the network requires 'unfolding' it in time. During training on a word, the inputs, outputs and feedback activations are stored for each frame. Then the errors are propagated back using the generalised delta rule [6], treating the network at different times as different layers of a multi-layer network (Figure 4b). To give target values for the training, we require a segmentation of the word into characters. For simplicity, an 'equal length' segmentation is used initially, where letters are assumed to be of the same length, though this is not true in practice. When the network performs reasonably well with this segmentation, we can use its own estimation of the optimal segmentation, which will be more accurate and enhances subsequent performance.

Since the network is being used to estimate probabilities, Bridle's [9] Softmax formulation is used on the output units. To improve performance and speed of convergence of the network we have made use of Jacobs' delta-bar-delta update rule [10], which gives each weight a variable training rate. Further improvements were obtained when Robinson and Fallside's [11] modifications to this rule were used. In order to achieve the optimum training time for the network, a validation procedure is used, whereby the network is tested on the validation set between periods of training. When recognition rates start to tail off, the network is considered trained and is tested on the test set.

5 Post-processing

The output of the network is in the form of probability estimates for each of the 27 categories in each of the frames. From this data we need to decide which word was written. This is achieved using dynamic programming [12] which relies on the principal of optimality, in the Viterbi algorithm to find a best match. We create a Markov model for each word in our vocabulary, with one state per letter, each state containing the probability that the data so far was generated by the preceding letters in that model. For each new frame, we take the network's probability estimates and multiply each state's probability by the probability of the corresponding letter. Transitions are allowed only from one state to itself or to the next state. After the last frame of data has been processed, the final state of each word model contains the probability that the observed data was produced by that model. By choosing the maximum of these likelihood estimators, if our model is good, we have a good estimate of the identity of the original word.

The Viterbi decoder can be improved by allocating transition probabilities which affect the average length of each letter. A simple durational model is included in the Viterbi decoder used, by giving each letter two successive identical states and thus forcing the minimum duration to be two frames.

6 Results

The system performance is judged by the number of words in the test set which are correctly identified, and these are quoted as a percentage of the 124 test words used. Using a network with 40 feedback units and 64 inputs/frame, a 73.4% recognition was obtained. Though the corpus used so far has only a small (31 word) vocabulary, we can investigate the system's scaling performance by testing the same data with a larger lexicon. This extended lexicon is created using the most common words in the LOB (Lancaster–Oslo/Bergen) corpus [13] to supplement the numbers lexicon. Using this lexicon the recognition rate was 60.4%.

7 Conclusions

The above results show that the method of recurrent error propagation networks can be applied successfully to the task of off-line cursive script recognition. Though comparison of results with other researchers is difficult (because of the difference in experimental details, the actual handwriting used and the method of data collection) we might note here the results which have been published for this problem . Božinović and Srihari achieved a 78% recognition rate with a 780 word lexicon [1] and Edelman, Ullman and Flash [3] quote a 50% recognition rate for whole words on a 30,000 word vocabulary.

Further improvements can be expected by adjustments in the preprocessing and parametrisation and also by the introduction of a language model, such as those already used successfully in the domain of speech recognition.

References

[1] R.M. Božinović and S.N. Srihari. Off-line cursive word recognition. *IEEE PAMI*, 11(1):68–83, January 1989.

[2] Sargur N. Srihari and Radmilo M. Božinović. A multi-level perception approach to reading cursive script. *Artificial Intelligence*, 33:217–255, 1987.

[3] S. Edelman, T. Flash, and S. Ullman. Reading cursive script by alignment of letter prototypes. *International Journal of Computer Vision*, 5(3):303–331, 1990.

[4] Rafael C. Gonzalez and Paul A. Wintz. *Digital Image Processing.* Addison Wesley, Reading, Mass., 1977.

[5] E.R. Davies. *Machine Vision: theory algorithms, practicalities.* Microelectrics and signal processing Number 9. London Academic, 1990.

[6] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In D.E. Rumelhart and J.L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, chapter 8, pages 318–362. Bradford Books, 1986.

[7] Barak A. Pearlmutter. Dynamic recurrent neural networks. Technical Report CMU-CS-88-191, CMU, School of Computer Science, Pittsburgh, PA15213, December 1990.

[8] Tony Robinson and Frank Fallside. A recurrent error propagation network speech recognition system. *Computer Speech and Language*, 5:259–274, 1991.

[9] John S. Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. *Neurocomputing*, F 68:227–236, 1990.

[10] Robert A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1:295–307, 1988.

[11] A.J. Robinson and F. Fallside. Phoneme recognition from the TIMIT database using recurrent error propagation networks. Technical Report TR 42, Cambridge University Engineering Department, Cambridge, UK., March 1990.

[12] H.F. Silverman and D.P. Morgan. The application of dynamic programming to connected speech recognition. *IEEE ASSP Magazine*, pages 6–25, July 1990.

[13] Stig Johansson, Roger Garside, Knut Hofland, and Geoffrey Leech. The tagged LOB corpus vertical/horizontal version. Technical report, Norwegian Computing Centre for The Humanities, 1986.