An Off-line Cursive Script Recognition System Using Recurrent Error Propagation Networks

A.W.Senior[*]   F.Fallside

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

Abstract

Computer handwriting recognition offers a new way of improving the human-computer interface and of integrating computers better into human society. This paper details a complete system for automatic, off-line recognition of handwriting which takes word images scanned from a handwritten page and identifies the words. The system is composed of normalisation and parametrisation operations which prepare word images for recognition by a recurrent network. After cross-validation training the network is able to model the handwritten data and can output letter probability estimates. These are accumulated in a Viterbi decoder which finds the best word match from a lexicon of permitted words. All of these processes are described in the paper, followed by results for experiments on two single-author cursive script databases and a discussion of these results and future work.

1 Introduction

Ever since the advent of computers, people have attempted to improve user interfaces to make them easier to use and hence accessible to a greater number of people. Recently great interest has been shown in handwriting recognition since it is a widely-used, convenient means of communication. So far, research into computer recognition of handwriting has concentrated on on-line recognition, which uses a stylus connected to the computer, and the important, but much harder, problem of off-line recognition, using scanned document images, has been largely ignored. Consequently few papers have been published in this area and results have so far been inadequate for most practical applications. There is a wide variety of potential commercial uses, such as reading addresses on envelopes, cheque verification, automatic fax handling and document processing which are waiting for a robust system of handwriting recognition to be developed, preferably independent of the writer and operating on a large vocabulary.

Previous approaches to the problem of off-line handwriting recognition (for instance those of Edelman, Ullman and Flash [3] and Božinović and Srihari [1]) have usually been based upon the spotting of features such as particular curve and stroke shapes in the

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

handwritten words. This paper presents an approach which relies on a much more basic feature — that of the line-segment, and builds up knowledge of the arrangements of such features in a recurrent error propagation network — a method previously unused for handwriting recognition. This approach avoids the need to write complex feature-spotting routines and the network builds up a representation of the permitted variation in, and complex patterns of, these line segments. The letter probability estimates which the network generates are combined in Markov models to choose the word in the lexicon which best matches the observed data.

2 Overview

The following sections describe in detail the complete off-line handwriting recognition system, which is shown in the block diagram of figure 1. First, the database on which the system is to operate is described, followed by a description of the preprocessing techniques needed to transform the data into a more usable form. Next, the recurrent network architecture, which forms the core of the recognition system, is detailed and variations to the conventional multi-layered perceptron are explained. Finally, the Viterbi decoder which produces word-level output is presented, with details of the enhanced durational modelling used. Results for this approach are presented and possible future improvements are discussed.

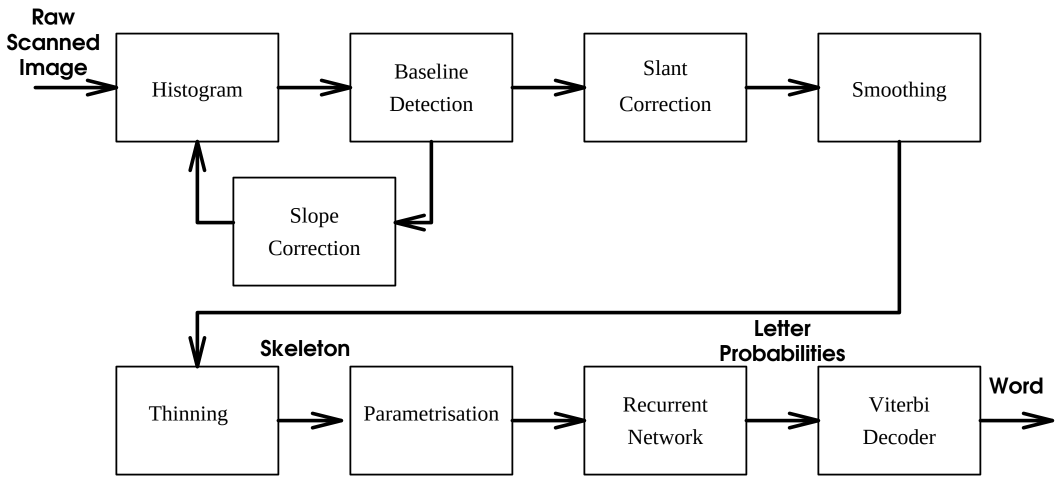


Figure 1: Schematic of the recognition system

3 Image acquisition and corpus choice

Lower case, cursive script words were written by a single author on a plain A4 sheet, each surrounded by white space to facilitate word extraction. These sheets were then scanned at 300 dpi resolution, in 8 bits to produce one file per page. These files were processed to extract each word into a separate file. Figure 2(a) shows one such word after thresholding.

The corpus chosen was the set of numbers written out as words (‘one’ to ‘nineteen’, tens from ‘twenty’ to ‘hundred’, plus ‘thousand’, ‘million’ and ‘zero’). These words were chosen because they form a corpus useful for an application such as cheque verification, but the small vocabulary enabled a reasonable study to be made in a short time and facilitated data collection. Seventeen exemplars of each of these words were taken, giving over 500 images, split into training, test and validation sets (see section 6). Further experiments were carried out on a database of 1330 words (520 word vocabulary) taken from the LOB natural language corpus [5], likewise split into training, test and validation sets.

4 Preprocessing

From the original scanned image, containing 8MB of information, all that is ultimately desired is the identity of the words on the page, which is in the order of bytes. One way of looking at recognition is as a process of information filtering with the ultimate aim of deriving this word information. In order to process the data effectively with a recognition technique such as a connectionist network, they must be reduced in number and transformed into a form more appropriate than a grey scale image, by the application of certain preprocessing techniques. In the same manner as filters, cepstra, Mel scale binning and vector quantisation are used to encode speech before attempting recognition, useful, invariant information must be extracted from the written words while discarding the vast majority of redundant variation.

A number of operations are involved in this sifting process. Each operation is described below, and figure 1 illustrates the whole process.

4.1 Histograms and baseline detection

Normalisation is of prime concern in preprocessing handwritten words. To normalise an image, it must be scaled so that letters are a standard height, and sheared so that letters are oriented vertically on a horizontal base. The character height is determined by finding the intuitively important lines which are shown running along the top and bottom of lower case letters in figure 2(b) — the half and base lines respectively, with a centre line between the two. With these lines, the ascenders and descenders which are used by human readers in determining word shape can also be identified.

The heuristic used for base line detection consists of the following steps:

1 Calculate the 'vertical density histogram' by counting the number of black pixels in each horizontal line in the image.

2 Reject the part of the image likely to be a descender.

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

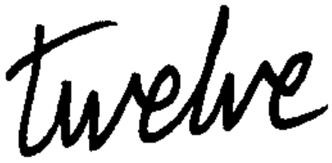
4 Carry out moment analysis to find the line of best fit.

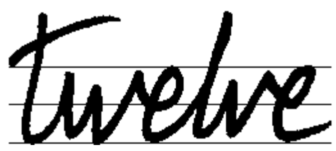
5 Reject all outliers, and calculate the new line of best fit. This is now considered to be the base line of the character.

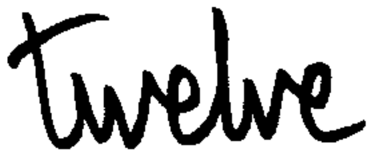
4.2 Slope and slant correction

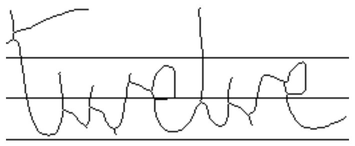
Given the estimate of the base line position, from section 4.1, the writing can be straightened to make the baseline horizontal. This straightening is carried out by application of a shear transform parallel to the y axis. (See figure 2b.) Next, the base line and the half line can be recalculated, using the knowledge that the baseline is now horizontal, by scanning a newly calculated histogram above and below the peak value, to find the 40%

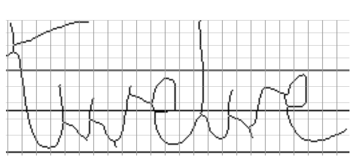
points of the slope. Slope correction can be carried out on whole lines to remove rotation in the scanned image or skewed writing, and then carried out on individual words to remove local tranformations.


(a) Initial image


(b) Slope corrected


(c) Slant corrected


(d) Thinned


(e) Boxes for parametrisation

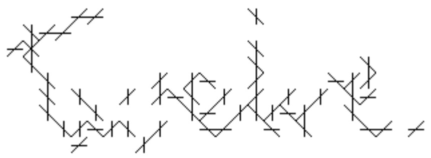

(f) The parametrised data

Figure 2: Successive stages in the preprocessing.

Božinović and Srihari [1] detail a method for letter slant correction (slant is the tilting of tall strokes away from the vertical). This involves isolating areas of the text which are near-vertical strokes and estimating the slant of each by calculating the centroids of the upper and lower halves. These slant estimates are averaged, and a shear carried out to remove the slant. This procedure has been found to be very sensitive to the thickness of the writing and is unreliable when the writing is thinner than expected. Figure 2d shows a slant-(over-)corrected word.

4.3 Smoothing and thinning

The word image is next convolved with a 2-dimensional Gaussian filter to remove noise, due either to scanning defects, or to applying transforms to discrete images. It has been found that there is negligible noise on a scanned image when using a black fibre-tip pen on plain white paper. Having normalised and smoothed the image, it is thresholded to leave every pixel black or white. Next 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. (See figure 2b.)

5 Parametrisation

Now that the image has been reduced to a standard form, which highlights invariants of the words and suppresses spurious variations, the preprocessed image needs to be parametrised in an appropriate manner for input to the network which is to carry out the recognition process. The scheme chosen for this parametrisation is described below.

The area covered by the word is first divided into rectangles. (See figure 2e.) 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, four bins are allocated to represent different line angles (vertical, horizontal, and the lines 45 degrees from these). Given this framework, the lines of the skeleton image are coarse coded:

Working from left to right in the thinned image, each stroke (a one-pixel wide line in the skeleton between junctions or end points) is divided into equal length segments (the length being proportional to the character height) for which the centroid position and angle are calculated. The box associated with this segment's (x, y, θ) values is now 'filled'. 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 2f shows the input pattern schematically. Each line represents a full bin and its position and orientation correspond roughly to the position and orientation of the section of skeleton which gave rise to it. 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 2f.

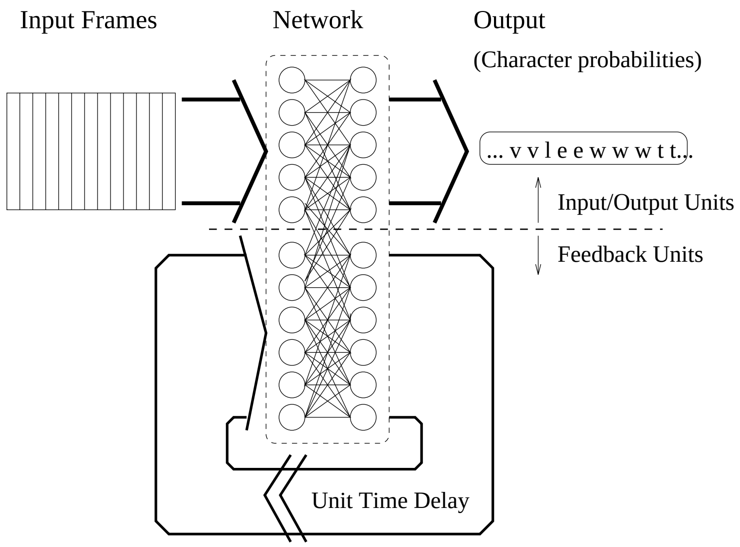
The training data for the networks must be labelled to allow the network to associate input patterns with the correct letter outputs, one per frame. To do this, a segmentation scheme is required. The scheme used initially is an 'equal length' scheme, where each letter in any word is assumed (though this is clearly inaccurate) to occupy the same number of frames of input. Thus in an n letter word which requires k frames, the first $\frac{k}{n}$ frames are labelled with the first letter of the word. Once the network is partially trained, a different scheme is employed, which uses the network's knowledge of letter lengths to label frames automatically and the approximate, equal-length segmentation is ignored. Alternatively the data could be hand-segmented according to what is actually present in the data. This more accurate labelling could reasonably be expected to give more accurate training and improved results, but with the penalty of a large amount of work.

Given the importance of features demonstrated by studies of reading [12], further improvements are to be expected if this parametrisation approach is combined with that of feature spotting. To derive a reliable set of rules for a complete set of features would be very difficult, but there are certain important features such as *'i'* dots, ascenders or turning points which are easy to identify reliably with simple rules. By spotting these features with rules and indicating them in the data representation passed on to the network, more relevant information can be preserved, and the burden on the network's processing is reduced. This extra data increases the size of a frame from 64 bins to 161. Hence, the number of links in a network with 40 feedback units increases from 7035 to 13,534 (a 90% increase) and the training time is correspondingly increased. Results with this enhanced preprocessing are compared to the basic method in section 8.

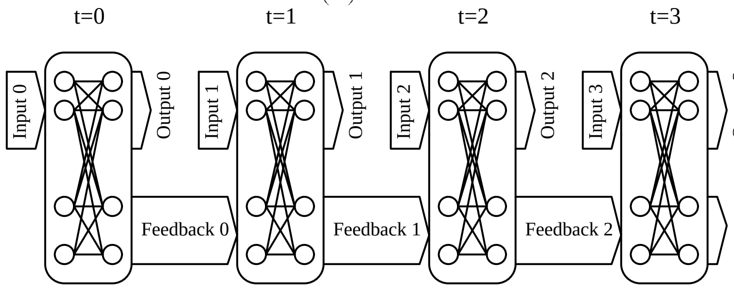
6 Recurrent networks

This section describes the 'recurrent networks' which are used to recognise the patterns of line segments present in the preprocessed data and to interpret them as letters. These networks are a type of connectionist (often termed 'neural') network which is to say that they are composed of a large number of very simple processing units with many interconnecting links. Each unit outputs a function of the weighted sum of its inputs, but the usefulness of such networks resides in the existence of training algorithms which can adjust the weight of each of the links to converge towards a desired function approximation — in this case the network is taught to recognise letters.

Recurrent networks have been successfully applied to speech recognition and other dynamic problems where a series of time-varying signals needs to be classified [7, 9]. A recurrent network is well suited to the recognition of patterns occurring in a time-series because the same processing is performed on each section of the input stream, but internal 'state' units are available to encode multi-frame context information about the preceding signal. By repeated presentation of training examples, the network parameters can be adjusted automatically so that the network recognises letter shapes in a sequence of preprocessed frames of data. It may be noted that recurrent networks were found to perform better than Time-Delay Neural Networks [14] for this task because of their better performance with time-distorted data.



(a) Schematic



(b) Net unfolded for training

Figure 3: The recurrent error propagation network

The recurrent network architecture used here is a single layer of standard perceptrons with sigmoid activation functions (as described by Rumelhart, Hinton and Williams [11]), however some of the output units are connected one-to-one to some of the input units with a unit time-delay, and these input units receive no external input (see figure 3a).

The remaining input units accept the parametrised data described above and the other 27 output units estimate letter probabilities for the 26 lower case letters plus a space character. During the forward pass, successive frames are presented at the input and results are fed back through the time delay, while output letter probabilities are read off from the other outputs. To allow the network to assimilate context information, several frames of data are passed through the network before the probabilities corresponding to the first frame are read off. An interval of two frames has been found to be most satisfactory.

Training the network requires 'unfolding' it in time. During training on a word, the inputs, outputs and feedback activations are stored for each frame. At the end of a word, the errors are propagated back using the generalised delta rule [11], treating the network at different times as different layers of a multi-layer network (figure 3b). One problem with network training is to know when to finish training — too much training can cause the network to over-specialise and to adapt too closely to the particular features of the training set. This would impair generalisation to unseen test data, but if training is stopped too early, the optimum performance will not be achieved. This problem is solved by creating a third data set which is used for validation. After training the network for a short time, the network's performance is tested on the validation set. This train and validate cycle is repeated until the recognition rate on the validation set starts to deteriorate. At this point training is stopped and the network is tested on the test data which has still not been put through the network.

The back-propagation algorithm (introduced as the modified delta rule in Rumelhart, Hinton & Williams [11]) used to train error-propagation networks has been modified in this work to speed up convergence and to make convergence to a good local minimum more likely. In addition to the incorporation of a momentum term in the weight update formulae, Jacobs' delta-bar-delta update rule [4] and Bridle's Softmax [2] have been used. The former provides for individual learning rates for each weight and the latter a different transfer function on the output units of the network, ensuring that the outputs are all probabilities (between 0 and 1 and summing to 1). Because of difficulties in training stability, modifications suggested by Robinson [10] were incorporated and gave much improved convergence. These changes use multiplicative learning rate changes and prevent the learning rates from deviating too far from the mean. For this work the geometric mean was used, and the additional measure of zeroing momentum terms when the network energy increased was taken.

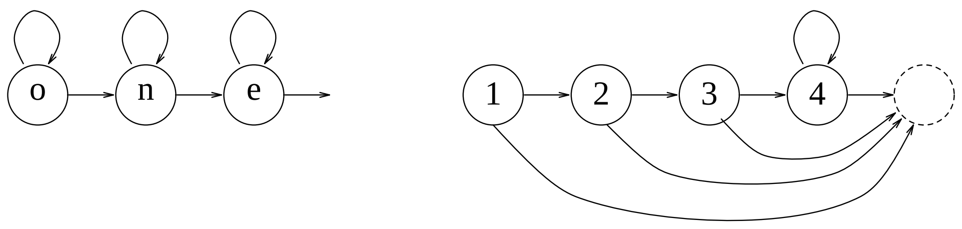
6.1 Prior probabilities

In any corpus of text, the occurrence frequencies $f(\mathcal{L})$ of different letters $\mathcal{L}$ will vary. In a typical section of text, the letter *'e'* may outnumber the letter *'q'* by a factor of several hundred. This discrepancy will severely distort the network's behaviour, since it will rarely be presented with *'q'*s and these will be swamped. A network trained on such data will recognise nearly anything as an *'e'* and is unlikely to recognise anything as a *'q'*. To overcome this problem, these biases are removed in the training. In the backpropagation algorithm, the objective function, or energy, E_p is defined by $E_p = \frac{1}{2}\sum_j(o_{pj} - t_{pj})^2$ for pattern p in class $\mathcal{L}(p)$, with output o_{pj} and target value t_{pj}. This determines the error signal propagated back through the network. If this error function is weighted so that more weight is given to rare letters, then all letter classes will contribute roughly the same amount to the error signal and the learning will be improved. The new objective function used is:

$$E_p = \frac{1}{2} \frac{\sum_j (o_{pj} - t_{pj})^2}{f(\mathcal{L}(p))} \tag{1}$$

7 Post-processing

The output of the network is in the form of probability estimates for each of the 27 letter classes in each of the frames. From this data, the word that generated the data must be determined. This is achieved using Dynamic Programming [13] in the form of the Viterbi algorithm. A Markov model [8] is created for each word, with one state per letter, each state containing the probability that the data so far were generated by the letters in that model. For every new frame, each state's probability is multiplied by the network's estimate of the probability of the letter to which that state corresponds. Transitions (transferring one state's probability to another state) are allowed only from one state to itself or to the next state. Figure 4a shows the model for the word 'one' with two arrows from each state marking the permitted transitions. After the last frame of data has been processed, the probability of the final state of any word model is the probability that the observed data was produced by that model. By choosing the maximum of these likelihood estimators, if the models are good, a good estimate of the identity of the original word is obtained.



(a) A simple Markov model (b) A complex durational model for one letter

Figure 4: Markov models

7.1 Durational modelling

An important consideration when using Markov models is to model the duration characteristics of the data correctly. The variations to be modelled are both inter-class and intra-class. A model of the first of these must reflect the fact that some letters take more frames (are wider) than others. The simple model adopted is to assume that 'm' and 'w' are roughly 50% wider than most letters, and 'i' and 'l' are half that width. These assumptions are reflected in the segmentation, and in the transition parameters of the Markov models as described in the next paragraphs. Some experiments on modelling intra-class duration variation with Markov models like that shown in figure 4b have been carried out [12] and show a significant improvement in recognition rate.

8 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 129 (330 for LOB) test words used. All results below are for networks with 40 feedback units trained using the cross-validation technique described in section 6. The results shown are averaged across four test runs, and are presented with standard deviations to indicate the sensitivity to initial conditions. The four test runs were identical except for the seed used by the random number generator which randomises the initial weights of the network to break symmetry. Experiments under other conditions used the same initial weights (the same seeds) so paired data analysis can be conducted to assess performance improvement. The first two columns of results are the means and sample standard deviations for models trained with equal-length segmentation, and the next two show the results after the network has been retrained using its own segmentation estimates.

Method	Corpus	Equal		Auto	
		Mean (%)	σ_{n-1}	Mean (%)	σ_{n-1}
Original preprocessing	Numbers	80.2	2.2	79.2	2.2
New preprocessing	Numbers	91.8	1.6	91.4	1.9
Tested on training data	Numbers	*97.6*	*1.2*	*99.1*	*0.7*
New preprocessing	LOB	68.6	1.5	69.0	1.9
Tested on training data	LOB	*84.0*	*4.6*	*84.6*	*4.3*

Table 1: Percentage recognition rates for networks with 40 feedback units.

The newer preprocessing method which incorporates the feature spotting mentioned in section 5 is clearly much better than the original. Automatic segmentation retraining does not seem to give significant improvements. The training carried out on this segmentation is probably *over-training* here as shown by the improvement in the test set performance. The difference in training and test performances indicate that some advantage could be derived from increasing the training set size for both corpora.

9 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. Comparison of results with other researchers is difficult because of differences in experimental details, the actual handwriting used and the method of data collection. Some other results are: Božinović and Srihari [1] 78% on a 780 word vocabulary, using data collected on-line; Edelman, Ullman & Flash [3] 47-53% with various authors, using a 30,000 word vocabulary, also using data collected on-line; Lecolinet & Crettez [6] 53% on a multi-writer, small-vocabulary database.

These experiments show that a 91.8% recognition rate can be achieved on the simpler task attempted, and 69.0% on the larger, more useful task. It can also be seen that the enhancements in preprocessing have significantly improved this performance. Further improvement can be expected with a future increase in the training set sizes. Improvements might also be achieved by making the preprocessing more robust and more detailed (for instance by spotting more features).

This paper has only addressed the problem of single-writer handwriting recognition, so it has limited applicability. Future work will continue to aim for higher recognition rates on the tasks presented here, but must also continue to increase the vocabulary size and begin to look at a writer independent task. Future work is also expected to include the investigation of the effect of grammatical constraints on recognition rates.

References

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

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

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

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

[5] S. Johansson, E. Atwell, R. Garside, and G. Leech. The tagged LOB corpus. Technical report, Norwegian Computing Centre for The Humanities, Bergen, 1986.

[6] Eric Lecolinet and Jean-Pierre Crettez. A grapheme-based segmentation technique. In *International Conference on Document Analysis and Recognition*, volume 2, pages 740–748, 1991.

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

[8] L.R. Rabiner and B.H. Juang. An introduction to hidden Markov models. *IEEE ASSP magazine*, 3(1):4–16, January 1986.

[9] A.J. Robinson. *Dynamic Error Propagation Networks.* PhD thesis, Cambridge University Engineering Department, February 1989.

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

[11] 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.

[12] A.W. Senior. Off-line handwriting recognition: A review and experiments. Technical Report TR105, Cambridge University Engineering Department, UK, December 1992. Available by anonymous ftp from svr-ftp.eng.cam.ac.uk.

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

[14] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K.J. Lang. Phoneme recognition using time delay neural networks. *IEEE Acoustics, Speech and Signal Processing*, 37(3):328 –339, March 1989.