A COMPARISON OF PREPROCESSORS FOR THE CAMBRIDGE RECURRENT ERROR PROPAGATION NETWORK SPEECH RECOGNITION SYSTEM

Tony Robinson†, John Holdsworth‡, Roy Patterson‡ and Frank Fallside†

† Cambridge University Engineering Department, Trumpington Street, Cambridge, England.
‡ Medical Research Council – Applied Psychology Unit, 15 Chaucer Road, Cambridge, England.

ABSTRACT

This paper makes a comparison of several preprocessors for the task of speaker independent phoneme recognition from the TIMIT database using a recurrent error propagation network recogniser [?]

The paper evaluates FFT, filterbank, auditory model and LPC based techniques in the spectral and cepstral domains and adds some simple features such as estimates of the degree of voicing, formant positions and amplitudes. The paper concludes that the features do not make a significant contribution and that the spectral domain representations, independent of their derivation, are better suited to this task. However, we find that the recogniser was relatively insensitive to preprocessor and changes in the architecture and training of the recogniser are more significant.

The current recognition rate on the TIMIT database of 61 symbols is 69.5% correct (64.0% including insertion errors) and on a reduced 39 symbol set the recognition rate is 76.1% correct (70.4%). This compares favourably with the results of other methods, such as Hidden Markov Models, on the same task.

1 INTRODUCTION

The Cambridge Recurrent Error Propagation Network Speech Recognition System has been shown to be able to perform speaker independent phoneme recognition as well as the best Hidden Markov Models (HMMs) [?, ?].

Where as the issue of preprocessors for HMMs has been well researched, this issue has received far less attention from the connectionist viewpoint. However, there is a fundamental difference between the nearest neighbour decisions surfaces formed in the input space by a HMM vector quantiser and the hyperplanes formed by error propagation networks. One of the advantages of the connectionist approach is that the elments of the input vector can have different variances without giving undue bias to these input dimensions. Thus, while it is necessary to code different types of HMM input in different codebooks, the connectionist input may be treated as a single large vector.

This paper begins by describing the basic recogniser and proceeds to evaluate many commonly used preprocessors. These contain combinations of two forms of input, spectral representations and simple low dimensional features. The spectral representations are based on filterbank, Fast Fourier Transform (FFT) and Linear Predictive Coding (LPC), and in the case of FFT and LPC the cepstral representations are also used. The features used are zero crossing rate, energy and estimates of pitch frequency, degree of voicing, formant positions and amplitudes. In addition to an evaluation of preprocessors, other means of improving the performance of the recurrent net are also considered.

As with our previous work, the evaluation of these preprocessors is performed on the DARPA TIMIT Acoustic Phonetic Continuous Speech Database [?] (hereafter known as the TIMIT database). This is a well respected large database which is widely available to other researchers, thus enabling comparison between this work and the work of others.

2 THE BASIC SYSTEM

The phoneme recogniser presented here is derived from earlier work by two of the authors [?]. In all but the filterbank based preprocessors, the 16kHz digitised speech from the TIMIT database is Hamming windowed with a duration of 32ms and a frame separation of 16ms. This frame is passed to the various preprocessors to yield a vector of about 20 coefficients for the recurrent network. The net is trained on a 64 processor array of T800 transputers offering about 60 Mflops. The output from the net is interpreted as a vector of probabilities that the frame was labelled with a particular phoneme. This vector stream can be segmented using dynamic programming to yield the most likely string of phoneme symbols from the probability distribution. Greater accuracy can be achieved by using the durational and bigram transitional probabilities to constrain the phoneme string. Each of these steps will now be described in more detail pointing out where the current system deviates from that previously reported.

2.1 The TIMIT Database

fdmy0*	fsmm0	mrds0	msfh1	mtkd0*
fjlr0*	fspm0	mree0	msfv0	mtlb0
fkdw0*	fsrh0	mrfl0	msjk0	mtlc0
fmbg0	fsxa0	mrgm0	msjs1	mtmr0
fmcm0	ftaj0	mrjm0	mslb0*	mtmt0
fnkl0	ftbr0	mrlj0	msmc0	mtpf0
fntb0*	ftbw0	mrlr0*	msmr0	mtpg0
frew0	ftlh0	mrms1	msrg0	mtpp0
frll0	futb0*	mroa0	msrr0	mtrr0
fsah0	fvkb0	mrpc0	msvs0	mtwh0*
fsak0	fvmh0	mrpc1	mtaa0	mtwh1
fscn0	mbjv0*	mrre0	mtab0	mvlo0
fsdj0	mdem0*	mrtj0	mtas0	mwbt0
fsem0*	mdlm0*	mrtk0	mtat0	mwdk0
fsgf0	mdss0*	mrvg0	mtbc0	mwem0
fsjg0	mejs0*	mrws0	mtdb0	mwew0
fsjs0	mfwk0*	mrws1	mteb0	mwjg0
fsjw0	mjee0*	mrxb0	mter0	mwsh0
fskp0	mpam0*	msas0	mtjm0	mzmb0
fslb1	mpfu0*	mses0	mtjs0*	
fsma0	mrab1	msfh0	mtju0	

Table 1: Sentences used in the test set

At the time of writing, only the prototype TIMIT database was available. This contains all the training material of the full database but none of the test material. Thus, it was necessary to partition this database into training an testing portions. There are 420 speakers in total which were divided into 317 speakers for training and 103 speakers for testing. Eight sentences were used per speaker (the `si` and `sx` sentences). The identity of the test speakers are given in table **??**, those marked with an asterisk are believed to have been used by Lee and Hon for testing their Hidden Markov model recogniser [?]. The authors are very grateful to Vassilios Digalakis and Mari Ostendorf of Boston University for their help in compiling this list.

2.2 Automatic scaling of input

For practical reasons, (memory limitations on disk and in the transputer RAM), the preprocessed data was scaled to fit into 8 bits per channel. This was done by computing a histogram and scaling so that no more than one in 500 samples lies outside the central 15/16th of the range. Typically this meant that one sample in 1000 would be thresholded.

2.3 The Recurrent Network

The recurrent network falls into the framework described by Rumelhart, Hinton and Williams [?]. It may be viewed as a single layer error propagation (back propagation) network, part of whose output is fed back to the input after a single frame time delay. This is shown in figure ?? where the external input, $u(t)$, and the state input, $x(t)$, together form the input vector, the output vector being composed of the external output, $y(t+1)$, and the state output, $x(t+1)$. In practice, the external output is not trained to classify the current input vector, $u(t)$, but that of n frames previously, $u(t-n)$. This is to allow some forward context in the classification, backward context is already available through the state vector. For most of these experiments, a four frame delay was used which corresponds to 64ms.

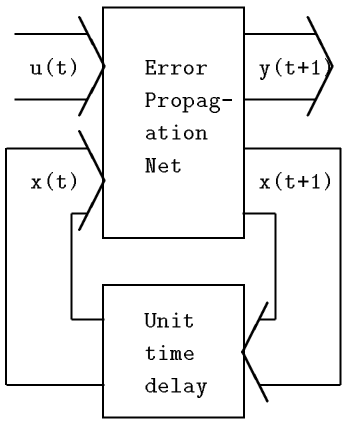

Figure 1: The recurrent network

The "time-expansion" or "batch" method of training recurrent networks is adopted for computational efficiency reasons. After 32 frames on each of the 64 transputers, the actual outputs are compared with the desired outputs and the contribution of these patterns to the gradient of the cross-entropy cost function is computed. Cross-entropy is used both because of the interpretation of the output units as probabilities and because it is found to reduce the training time needed.

An adaptive step size algorithm was necessary to achieve training in reasonable time. Each weight has a step size associated with it and the weight is changed by this amount in the direction of the locally computed gradient. If this gradient agrees in sign with the gradient when smoothed with a first order filter over the whole of the training set, then the step size is increased, otherwise it was decreased. In most experiments, the increase was multiplication by a factor of 1.116 and the decrease was a factor of 0.9. In two cases, pzc and pow, this proved to be unstable, and increases of 1.1155 and 1.113 were used, respectively.

For the majority of the work presented in this paper, 96 state nodes were used, which yields about 20,000 weights to be trained. 32 passes through the training set were found to be sufficient and this could be achieved in 17 hours on the transputer array.

2.4 Overview of the preprocessor search

The space of commonly used preprocessors and combinations of these is too large to search exhaustively. As a simplifying assumption, some representation of the short term power spectrum was taken to be the most important feature. These representations were normalised and a separate power channel was added which was common to all preprocessors. A common feature of preprocessors for hidden Markov models is to use the difference of adjacent frames as extra input to the recognition system. This was thought to be unnecessary in the recurrent net case as the equivalent information may be computed by storing the previous input in the state vector.

The results will be presented in tabular form using a three letter acronym (tla) to refer to the preprocessor and giving the percentage correct and the percentages of insertion, substitution and deletion errors. In addition the accuracy defined as 100% minus the number of insertion, substitution and deletion errors is given. Table ?? gives the basic recognition rates when only the power channel is used (pow).

tla	correct	insert	subst.	delet.	accur.
pow	36.6%	12.9%	50.5%	12.9%	23.7%

Table 2: Recognition rates: Power alone

3 FFT BASED TECHNIQUES

The basis of the preprocessor used in previous work was the cube roots of a 20 channel bark scaled Fast Fourier Transform (FFT). The recognition rates for five different sets of initial weights are given in table ??. The mean of these will form the reference preprocessor, p20. The standard deviation (s.d.) of these results is about 0.3%, which provides a lower bound for significant difference between preprocessors.

tla	correct	insert	subst.	delet.	accur.
pr0	65.5%	6.1%	27.5%	7.0%	59.4%
pr1	65.4%	6.2%	27.6%	7.1%	59.2%
pr2	66.0%	6.2%	27.1%	7.0%	59.8%
pr3	65.4%	5.9%	27.5%	7.1%	59.5%
pr4	65.3%	6.2%	27.7%	7.0%	59.1%
p20	65.5%	6.1%	27.5%	7.0%	59.4%
s.d.	0.3%	0.1%	0.2%	0.1%	0.3%

Table 3: 20 channel bark scale FFT

Increasing the dimensionality of the preprocessor output may allow it to carry more information at the expense of increasing the computation needed in the preprocessor and the recogniser. More training data is needed if the added channels contain noise, to avoid the fitting of the model to this noise. Table ?? shows the effect of a factor of three in the number of number of channels used, there is no discernible trend.

tla	correct	insert	subst.	delet.	accur.
p16	65.5%	5.9%	27.4%	7.1%	59.6%
p20	65.5%	6.1%	27.5%	7.0%	59.4%
p36	66.3%	6.4%	26.8%	6.9%	59.9%
p48	65.6%	6.4%	27.3%	7.0%	59.2%

Table 4: Dimensionality of input

Table ?? shows the effect of varying the compression function applied to the channels. pc1 is no compression, p20 is the standard cube root compression, pc5 is a fifth root compression and pln is the conventional logarithmic compression. pc1 shows that it is important to use some form of compression to reduce the dynamic range, but there is little difference between the other three compression functions.

4 FILTERBANK AND THE AUDITORY MODEL

Two studies were performed to test whether a preprocessor modeled on the human auditory system would be particularly suited to the recurrent network.

The first stage of auditory processing is spectral decomposition which is commonly simulated by a bank of band pass filters. Data from psycho-physical experiments on human subjects indicate the

tla	correct	insert	subst.	delet.	accur.
pc1	62.1%	7.3%	29.8%	8.1%	54.8%
p20	65.5%	6.1%	27.5%	7.0%	59.4%
pc5	65.4%	6.1%	27.5%	7.1%	59.4%
pln	65.0%	6.7%	28.1%	6.9%	58.3%

Table 5: Compression functions

equivalent rectangular bandwidth of these filters vary roughly in proportion to the center frequency of the filter in accordance with the following equation modified from [?].

$$erb(f) = 24.7 + f/9.265 \quad (1)$$

In order to provide an input frame for the network every 16ms, the output of each filter in a bank of such filters is Hamming windowed and then the cube root taken of the energy integrated across time. Though this is a rather crude method of collapsing the filter output, rows c20 and c36 in table ?? show satisfactory results for a 20 and 36 channel implementation of this spectral decomposition stage using 4th order Butterworth filters. These results confirm that a front end based on auditory parameters can perform very well. This is as would be expected given the theoretical similarity to the bark scaled FFT preprocessor which is already based on parameters from the human auditory system.

The second stage of auditory processing involves some form of magnitude compression and then adaptation with respect to signal level both across time and across frequency. This adaptation is required in order to cope with the very great variation in signal level encountered in the environment. The results fed in table ?? are from a implementation of such a system including filtering, pure logarithmic compression and rapid level adaptation. These results though reasonable were disappointing when compared to other preprocessors.

tla	correct	insert	subst.	delet.	accur.
c20	65.5%	6.0%	27.3%	7.2%	59.5%
c36	65.4%	6.5%	27.5%	7.1%	58.9%
fed	61.3%	7.2%	31.0%	7.7%	54.1%

Table 6: Filterbank and Auditory model recognition results

This poor performance could be due to two factors. Firstly adaptation combined with the pure logarithmic compression could be performing an excessive normalisation of the input signal giving rise to an over emphasis of small features in the signal. To prevent excessive normalisation we propose to implement a limit on adaptation determined by the recent signal level. Psycho-physical data suggest such a time varying adaptation limit exists in the the human auditory system.

The second cause of the poor performance could be the spectral sharpening included in this more complete simulation to model the effect known as two-tone suppression in hearing. Current networks are limited to a small number of input channels. In this case the sharpening can result in a spectral feature being entirely contained in one channel of network input. As a result of this, small variations in the position of spectral features such as formants can result in a feature moving completely from one channel to another. This again brings about an over-emphasis of small variations in the input. Further tests will be necessary with either this sharpening removed, or perhaps the input vector to the network smoothed. In the longer term, however, when using this type of spatially organised input, the network should probably be adapted to somehow associate points adjacent in the input frame.

5 LPC BASED TECHNIQUES

Linear Predictive Coding (LPC) is a very popular front end for speech recognition systems because of the low computational cost compared with FFT or filterbank methods. All the preprocessors considered in this section will use 16th order LPC computed using the autocorrelation method. acf is just the autocorrelation values, lpf is the resulting linear predictor filter coefficients, lpa is the log area ratios of the equivalent lossless tube, cep are the cepstral coefficients derived from LPC and l20 is a the cube rooted 20 sample bark scale spectrum derived from the LPC filter. In addition, the Smoothed Group delay model of Singer, Umezalia and Itakura was included because of its good quantisation properties [?].

It is interesting to note that although all these representations are derived from the same set of autocorrelation coefficients, there is a significant range in performance. lpf and acf which are linearly related to the power spectrum (pc1), show poorer performance even though this mapping can be incorporated into the first layer of weights at no additional cost. l20 shows slightly worse recognition than p20, presumably because of the smoothing imposed on the power spectrum by the LPC representation.

tla	correct	insert	subst.	delet.	accur.
lpf	54.6%	8.1%	37.3%	8.1%	46.6%
acf	58.9%	7.4%	33.2%	8.0%	51.4%
lpa	62.0%	7.3%	30.8%	7.2%	54.7%
cep	63.4%	6.6%	29.3%	7.3%	56.8%
sgd	64.4%	6.3%	28.3%	7.3%	58.1%
l20	64.7%	6.3%	28.0%	7.3%	58.4%
p20	65.5%	6.1%	27.5%	7.0%	59.4%

Table 7: LPC based preprocessors

6 ADDING ADDITIONAL FEATURES

The previous sections have established that the bark scaled FFT, p20, was as good as any of the preprocessors tried. This section adds additional features to this preprocessor in the hope that new information will be added which will increase the recognition rate. The frame rate of 16ms is towards the high end of those used in speech recognition so preprocessors pp2, pp4 and pp8 divide the frame into 2, 4, and 8 sections respectively and calculate the power in each section so that an energy contour through the frame is available. These results are given in table ?? which show no trend of increasing accuracy with more energy channels, even though examination of the weight matrix reveals that some units detect changes in amplitude within a frame.

tla	correct	insert	subst.	delet.	accur.
p20	65.5%	6.1%	27.5%	7.0%	59.4%
pp2	66.0%	6.2%	27.0%	6.9%	59.8%
pp4	65.4%	6.2%	27.8%	6.8%	59.2%
pp8	65.8%	6.1%	27.3%	6.9%	59.7%

Table 8: Differing numbers of energies per frame

For Hidden Markov Model recognition, the bark (or equivalently mel) scaled cepstrum, bsc, is more often used than the frequency domain representation. Table ?? shows a slightly worse performance in the cepstral domain, but this figure is very close to the pln entry of table ?? to which it is linearly related.

p20 was augmented with several types of feature: pzc adds zero crossing information; pf0 adds the position of the highest peak in the cepstrum corresponding to a pitch frequency; pf3 adds the positions of the first three formants measured as peaks in the bark scaled spectrum; ppa adds the amplitudes as well as the positions of these peaks. Finally pre is the bark scaled spectrum with four energies per frame (as pp4), and all the above features with the exception of the amplitudes of the formants, which corresponds to the preprocessor of previous work. All these results are given in table ??, but unfortunately no preprocessor offers significantly better results than p20.

7 TUNING THE RECURRENT NET

The number of state units was set to 96 in order to achieve training of many preprocessors at reasonable speed. Table ?? shows

tla	correct	insert	subst.	delet.	accur.
p20	65.5%	6.1%	27.5%	7.0%	59.4%
bsc	64.8%	6.4%	28.2%	7.0%	58.4%
pzc	65.2%	6.4%	27.8%	7.1%	58.8%
pfv	65.1%	6.2%	27.8%	7.1%	58.9%
pf0	65.8%	6.0%	27.1%	7.1%	59.8%
pf3	66.0%	6.1%	27.1%	6.9%	59.9%
ppa	65.7%	6.4%	27.4%	6.9%	59.3%
pre	65.6%	6.3%	27.6%	6.8%	59.3%

Table 9: Additional features

the effect of increasing this to 128, 192 and 256. The number of weights, and hence the amount of computation needed per training pair, increases as the square of the number of state units. This limited the maximum dimensionality to 256. Also shown in table ?? is the effect of preprocessing the data with multiple offsets and training on all the slightly different sets of data. Four offsets were chosen and the number of presentations of all the training data was increased from 32 to 96. `pd4` has the standard four frame delay and `pd6` has a six frame delay, both with 192 state units.

tla	correct	insert	subst.	delet.	accur.
p20	65.5%	6.1%	27.5%	7.0%	59.4%
128	67.1%	6.1%	26.2%	6.7%	61.0%
192	68.9%	6.0%	24.8%	6.2%	63.0%
256	69.4%	5.7%	24.5%	6.2%	63.6%
pd4	69.5%	5.5%	24.0%	6.4%	64.0%
pd6	69.0%	5.1%	24.5%	6.5%	63.9%

Table 10: More state units and more training data

It is clear that increasing the number of state units and using multiple offsets both provide significant increase in performance.

8 COMPARISON WITH HMM

The best Hidden Markov Model phoneme recognition results are currently obtained by using multiple codebooks and smoothed triphone models and the best published results to date for the TIMIT task come from the SPHINX recognition system [?]. By simply mapping the 61 TIMIT symbol set onto the 39 CMU/SPHINX symbol set for the `pd4` configuration it is possible to make a comparison between the Hidden Markov Model (HMM) and the Artificial Neural Network (ANN) techniques. These results are shown in table ??, the difference in performance between the two techniques is believed to be significant.

tla	test	size	correct	insert	subst.	delet.	accur.
ANN	large	61	69.5%	5.5%	24.0%	6.4%	64.0%
ANN	large	39	76.1%	5.7%	17.4%	6.5%	70.4%
ANN	small	61	69.3%	6.3%	24.6%	6.1%	62.9%
ANN	small	39	76.3%	6.3%	17.6%	6.1%	69.9%
HMM	small	39	73.8%	7.7%	19.6%	6.6%	66.1%

Table 11: Comparison with the SPHINX phone recogniser

9 CONCLUSION

From the previous sections it can be seen that most power spectrum based preprocessors give about the same performance. It would have been possible to run each preprocessor several times with different starting weights, so eliminating this source of variance and so obtaining a more accurate ranking. However, the difference between preprocessors was found to be small and changes to the network were found to yield far more significant improvements.

Is is perhaps not surprising that the conventional signal processing front ends perform better for speech recognition by machine than the auditory model, since signal processing methods have been intensively studied and optimised for just this purpose, whereas auditory models have been pursued mainly for psychophysical modelling results.

It is interesting that the autocorrelation function (`acf`) and linear predictive filter (`lpf`) performed worse than the other LPC techniques, even though the latter preprocessors could all be derived from the former. This demonstrates that although within the connectionist framework it is theoretically possible to perform any arbitrary mapping, the data representation, in the form of the choice of preprocessor, is important.

Unfortunately, the best preprocessor for phoneme recognition is not necessarily the best for word recognition, as has been demonstrated by a number of researchers including Russell et. al. [?]. As a result, this work is currently being extended to word recognition and preliminary results from the 1000 word DARPA Resource Management task will soon be presented [?].

This paper has presented the results of training 35 networks at about 17 hours each on a 64 processors array. This represents over 4 CPU years and it is disappointing that no significant improvement in preprocessors was found. However, changes to the recurrent network have yielded increases in performance, and the authors believe that the resulting technique yields the best results on this task to date.

10 ACKNOWLEDGEMENTS

The work described in this paper was carried out as part of an ESPRIT Basic Research Action project (3207). The authors would like to acknowledge NIST for the provision of the TIMIT database and the ParSiFal project IKBS/146 which developed the transputer array.

REFERENCES

[1] Tony Robinson and Frank Fallside. Phoneme recognition from the TIMIT database using recurrent error propagation networks. Technical Report CUED/F-INFENG/TR.42, Cambridge University Engineering Department, March 1990.

[2] Kai-Fu Lee and Hsiao-Wuen Hon. Speaker-independent phone recognition using hidden Markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(11):1641–1648, November 1989.

[3] John S. Garofolo. *Getting Started with the DARPA TIMIT CD-ROM: An Acoustic Phonetic Continuous Speech Database*. National Institute of Standards and Technology (NIST), Gaithersburgh, MD, 1988.

[4] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical Report ICS-8506, University of California, San Diego, September 1985.

[5] Brian R. Glasberg and Brian C. J. Moore. Derivation of filter shapes from notched-noise data. *Hearing Research*.

[6] H. Singer, T. Umezaki, and F. Itakura. Low bit quantization of the smoothed group delay spectrum for speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 761–764, Albuquerque, 1990.

[7] M. J. Russell, K. M. Ponting, S. M. Peeling, S. R. Browning, J. S. Bridle, R. K. Moore, I. Galiano, and P. Howell. The ARM continuous speech recognition system. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 69–72, Albuquerque, 1990.

[8] Tony Robinson and Frank Fallside. Word recognition from the DARPA resource management database with the Cambridge recurrent error propagation network speech recognition system. In *Third Australian International Conference on Speech Science and Technology*, Melbourne, November 1990.