Connectionist Glue: Modular Design of Neural Speech Systems

Alex Waibel
ATR Interpreting Telephony Research Laboratories, Osaka, Japan
Carnegie Mellon University, Pittsburgh, PA 15213

Abstract[1]

Scaling connectionist models to larger connectionist systems is difficult, because larger networks require increasing amounts of training time and data and the complexity of the optimization task quickly reaches computationally unmanageable proportions. In this paper, we train several small Time-Delay Neural Networks aimed at all phonemic subcategories (nasals, fricatives, etc.) and report excellent fine phonemic discrimination performance for all cases. Exploiting the hidden structure of these smaller phonemic subcategory networks, we then propose several techniques that allow us to "grow" larger nets in an incremental and modular fashion without loss in recognition performance and without the need for excessive training time or additional data. These techniques include *class discriminatory learning, connectionist glue, selective/partial learning and all-net fine tuning*. A set of experiments shows that stop consonant networks (BDGPTK) constructed from subcomponent BDG- and PTK-nets achieved up to 98.6% correct recognition compared to 98.3% and 98.7% correct for the component BDG- and PTK-nets. Extensions to other tasks are discussed.

1. Introduction

A number of studies have recently demonstrated that connectionist architectures capable of capturing some critical aspects of the dynamic nature of speech, can achieve superior recognition performance for small but difficult phonemic discrimination tasks [Waibel 87, Waibel 89, Watrous 88]. Encouraged by these results we would like to explore the question, how we might expand on these models to make them useful for the design of speech recognition systems. A problem that emerges, however, as we attempt to apply neural network models to the full speech recognition problem is the problem of scaling. Simply extending our networks to ever larger structures and retraining them soon exceeds the capabilities of even the fastest and largest of today's supercomputers. Moreover, the search complexity of finding an optimal solution in a huge space of possible network configurations quickly assumes unmanageable proportions. In an effort to extend our models from small recognition tasks to large scale speech recognition systems, we must therefore explore modularity and incremental learning as design strategies to break up a large learning task into smaller subtasks. Breaking up large tasks into subtasks to be tackled by individual black boxes interconnected in ad hoc arrangements, on the other hand, would mean to abandon one of the most attractive aspects of connectionism: the ability to perform complex constraint satisfaction tasks in a massively parallel and interconnected fashion, in view of an overall optimal performance goal. In this paper we demonstrate based on a set of experiments aimed at phoneme recognition that it is indeed possible to construct large neural networks by exploiting the hidden structure of smaller trained subcomponent networks. A set of successful techniques is developed that bring the design of practical large scale connectionist recognition systems within the reach of today's technology.

The present paper has five parts: In the next section we review Time-Delay Neural Networks as a technique to achieve accurate, reliable classification of phonemes in small but ambiguous phonemic subcategories (e.g., BDG, PTK, etc.). Excellent performance results are reported for *all* phonemic coarse classes found in a Japanese large vocabulary word database. In section 3, we then explore techniques for the modular extension of small networks to larger "connectionist systems". In section 4, we validate the usefulness of these techniques by applying them to the harder and larger tasks. We summarize our results in the last section of this paper.

2. Small Phonemic Classes by Time-Delay Neural Networks

To be useful for the proper classification of speech signals, a neural network must have a number of properties. First, it should have multiple layers and sufficient interconnections between units in each of these layers. This is to ensure that the network will have the ability to learn complex non-linear decision surfaces [Lippmann 87]. Second, the network should have the ability to represent relationships between events in time. These events could be spectral coefficients, but might also be the output of higher level feature detectors. Third, the actual features or abstractions learned by the

[1] A more detailed version of this paper has been submitted to the IEEE Transactions on Acoustics, Speech and Signal Processing.

network should be invariant under translation in time. Fourth, the learning procedure should not require precise temporal alignment of the labels that are to be learned. Fifth, the number of weights in the network should be small compared to the amount of training data so that the network is forced to encode the training data by extracting regularity. In the following, we review Time-Delay Neural Networks (TDNNs) as an architecture that satisfies all of these criteria and was designed explicitly for the classification of phonemes within small phonemic classes such as the voiced stops, "B", "D", "G", the voiceless stops "P", "T", "K", etc.

2.1. Review of a Time-Delay Neural Network's Architecture

The basic unit used in many neural networks computes the weighted sum of its inputs and then passes this sum through a non-linear function, most commonly a threshold or sigmoid function [Lippmann 87, Rumelhart 86a]. In our TDNN, this basic unit is modified by introducing delays D_1 through D_N as shown in Fig.1. The J inputs of such a unit now will be multiplied by several weights, one for each delay and one for the undelayed input. For N = 2, and J = 16, for example, 48 weights will be needed to compute the weighted sum of the 16 inputs, with each input now measured at three different points in time. In this way a TDNN unit has the ability to relate and compare current input with the past history of events. The sigmoid function was chosen as the non-linear output function F due to its convenient mathematical properties [Rumelhart 86a, Rumelhart 86b].

For the recognition of phonemes, a three layer net is constructed. Its overall architecture and a typical set of activities in the units are shown in Fig.2 based on one of the phonemic subcategory tasks (BDG).

At the lowest level, 16 melscale spectral coefficients serve as input to the network. Input speech, sampled at 12 kHz, was hamming windowed and a 256-point FFT computed every 5 msec. Melscale coefficients were computed from the power spectrum [Waibel 87, Waibel 89] and adjacent coefficients in time collapsed resulting in an overall 10 msec frame rate. The coefficients of an input token (in this case 15 frames of speech centered around the hand labeled vowel onset) were then normalized to lie between -1.0 and +1.0 with the average at 0.0. Fig.2 shows the resulting coefficients for the speech token "BA" as input to the network, where positive values are shown as black and negative values as grey squares.

This input layer is then fully interconnected to a layer of 8 time delay hidden units, where J = 16 and N = 2 (i.e., 16 coefficients over three frames with time delay 0, 1 and 2). An alternative way of seeing this is depicted in Fig.2. It shows the inputs to these time delay units expanded out spatially into a 3 frame window, which is passed over the input spectrogram. Each unit in the first hidden layer now receives input (via 48 weighted connections) from the coefficients in the 3 frame window. The particular delay choices were motivated by earlier studies [Lang 87] [Waibel 87] [Waibel 89] [Makino 86] [Blumstein 79] [Blumstein 80] [Kewley-Port 83].

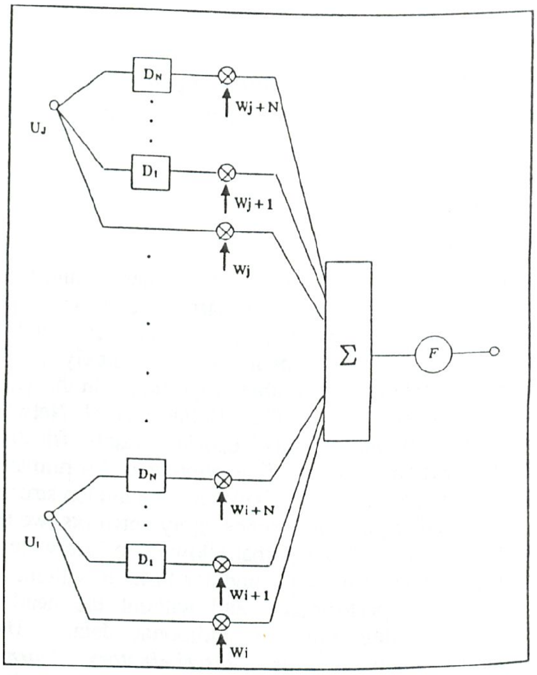

Figure 1: A Time Delay Neural Network (TDNN) unit

In the second hidden layer, each of 3 TDNN units looks at a 5 frame window of activity levels in hidden layer 1 (i.e., J = 8, N = 4). The choice of a larger 5 frame window in this layer was motivated by the intuition that higher level units should learn to make decisions over a wider range in time based on more local abstractions at lower levels.

Finally, the output is obtained by integrating (summing) the evidence from each of the 3 units in hidden layer 2 over time and connecting it to its pertinent output unit (shown in Fig.2 over 9 frames for the "B" output unit). In practice, this summation is implemented simply as another TDNN unit which has fixed equal weights to a row of unit firings over time in hidden layer 2. While the network shown in Fig.2 was designed for a 3 class problem (e.g., BDG or PTK), variations to accommodate 2, 4 or 5 classes are easily implemented by allowing for 2, 4 or 5 units in hidden layer 2 and in the output layer.

When the TDNN has learned its internal representation, it performs recognition by passing input

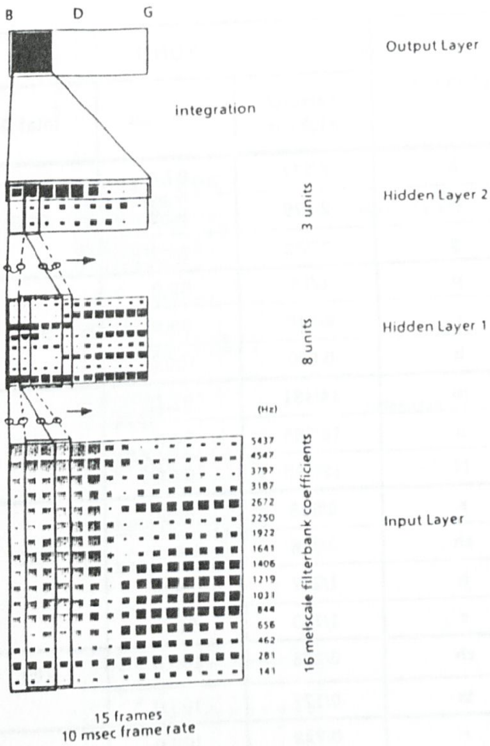

Figure 2: The TDNN architecture (input: "BA")

speech over the TDNN units. In terms of the illustration of Fig.2 this is equivalent to passing the time delay windows over the lower level units' firing patterns. At the lowest level, these firing patterns simply consist of the sensory input, i.e., the spectral coefficients.

Each TDNN unit outlined in this section has the ability to encode temporal relationships within the range of the N delays. Higher layers can attend to larger time spans, so local short duration features will be formed at the lower layer and more complex longer duration features at the higher layer. The learning procedure ensures that each of the units in each layer has its weights adjusted in a way that improves the network's overall performance.

The network described is trained using the Back-propagation Learning Procedure [Rumelhart 86a, Rumelhart 86b]. This procedure iteratively adjusts all the weights in the network so as to decrease the error obtained at its output units. For translation invariance, we need to ensure during learning that the network is exposed to *sequences* of patterns and that it is allowed (or encouraged) to learn about the most powerful cues and sequences of cues among them. Conceptually, the back-propagation procedure is applied to speech patterns that are stepped through in time. An equivalent way of achieving this result is to use a spatially expanded input pattern, i.e., a spectrogram plus some constraints on the weights. Each collection of TDNN-units described above is duplicated for each one frame shift in time. In this way the whole history of activities is available at once. Since the shifted copies of the TDNN-units are mere duplicates and are to look for the same acoustic event, the weights of the corresponding connections in the time shifted copies must be constrained to be the same. To realize this, we first apply the regular back-propagation forward and backward pass to all time shifted copies as if they were separate events. This yields different error derivatives for corresponding (time shifted) connections. Rather than changing the weights on time-shifted connections separately, however, we actually update each weight on corresponding connections by the same value, namely by *the average* of all corresponding time-delayed weight changes[2]. Fig.2 illustrates this by showing in each layer only two connections that are linked to (constrained to have the same value as) their time shifted neighbors. Of course, this applies to all connections and all time shifts. In this way, the network is forced to discover useful acoustic-phonetic features in the input, regardless of when in time they actually occurred. This is an important property, as it makes the network independent of errorprone preprocessing algorithms, that otherwise would be needed for time alignment and/or segmentation.

2.1.1. Experimental Conditions, Database

For performance evaluation, we have used a large vocabulary database of 5240 common Japanese words [Waibel 87, Waibel 89]. The data used in this paper was uttered in isolation by one male native Japanese speaker (MAU). All utterances were recorded in a sound proof booth and digitized at a 12 kHz sampling rate. The database was then split into a training set and a testing set of 2620 utterances each, from which the actual phonetic tokens were extracted. The training tokens (up to 600 tokens per phoneme[3]) were randomized within each phoneme class. For a given training run they were then presented, alternating between each class to be learned. If a phoneme class was represented by an insufficient number of available training tokens, random tokens from its set were repeated, in order to preserve the alternating sequence of presentations among all training tokens. For performance evaluation, we have run all

[2] Note that weight changes were carried out after presentation of all training samples [Rumelhart 86b].

[3] Note, that for some phoneme categories an unnecessarily large number of tokens was found in the database (e.g., vowels), while for some others (e.g., "P") only few tokens were extracted. While excessive tokens are simply discarded at random to reduce the dataset size, a lack of tokens leads to poor generalization. The low recognition scores for "P" are therefore a result of the limited training data.

experiments on the testing tokens only, i.e., on tokens *not* included during training.

The entire database was phonetically handlabeled [Sagisaka 87]. These labels were used in the experiments reported below to center a given phoneme in the input range used for learning and evaluation. No attempt was made to correct for improper handlabels. Since all networks described here were trained in a translation invariant fashion, possible misalignments at the input are of no serious concern as long as all the critical features needed for discrimination are present *somewhere* in the input range. For consistency among our networks and efficiency of learning, we continued to employ a 150 msec input range. Note, however, that longer input ranges are possible and might in fact be preferable to extract all useful features of a given phoneme. All tokens in the database were included in the test set or the training set, respectively, and no preselection was done. The resulting data included a considerable amount of variability (see [Waibel 87, Waibel 89] for examples) due to its position within an utterance or phonetic context.

2.2. Discrimination Performance in Phonemic Subclasses

To evaluate our TDNNs on all phoneme classes (for an in depth discussion and comparative performance evaluation for voiced stops see [Waibel 87, Waibel 89]), recognition experiments have been carried out for seven phonemic subclasses found in the database. For each of these classes, TDNNs with an architecture similar to the one shown in Fig.2 were trained. A total of seven nets aimed at the major coarse phonetic classes in Japanese were trained, including voiced stops B, D, G, voiceless stops P,T,K, the nasals M, N and syllabic nasals, fricatives S, SH, H and Z, affricates CH, TS, liquids and glides R, W, Y and finally the set of vowels A, I, U, E and O. Each of these nets was given between two and five phoneme classes to distinguish and the pertinent input data was presented for learning. Note, that each net was trained only within each respective coarse class and has no notion of phonemes from other classes yet. Table 2-1 shows the recognition results for each of these major coarse classes.

phoneme	TDNN		
	#errors/ #tokens	%correct	total %
b	5/227	97.8	98.6
d	2/179	98.9	
g	2/252	99.2	
p	6/15	60.0	98.7
t	6/440	98.6	
k	0/500	100.0	
m	14/481	97.1	96.6
n	16/265	94.0	
N	12/488	97.5	
s	6/538	98.9	99.3
sh	0/316	100.0	
h	1/207	99.5	
z	1/115	99.1	
ch	0/123	100.0	100
ts	0/177	100.0	
r	0/722	100.0	99.9
w	0/78	100.0	
y	1/174	99.4	
a	0/600	100.0	98.6
i	1/600	99.8	
u	25/600	95.8	
e	8/600	98.7	
o	7/600	98.8	

Table 2-1: Recognition Results for 7 Phoneme Classes

3. Scaling TDNNs to Larger Phonemic Classes

We have seen in the previous section that TDNNs achieve superior recognition performance on difficult but small recognition tasks. To train these networks, however, substantial computational resources were needed. This raises the question of how our good but admittedly limited networks could be extended to encompass *all* phonemes or handle speech recognition in general. To shed light on this question of scaling, we consider first the problem of extending our networks from the task of voiced stop consonant recognition (hence the BDG-task) to the task of distinguishing among *all* stop consonants (the BDGPTK-task).

3.1. The Problem of Training Time

For a network aimed at the discrimination of the voiced stops (a BDG-net), approximately 6000 connections had to be trained over about 800 training tokens. An identical net (also with approximately 6000 connections to be trained[4]) can achieve discrimination

[4] Note, that these are connections over which a back-propagation pass is performed during each iteration. Since many of them share the same weights, only a small fraction (about 500) of them are actually free parameters.

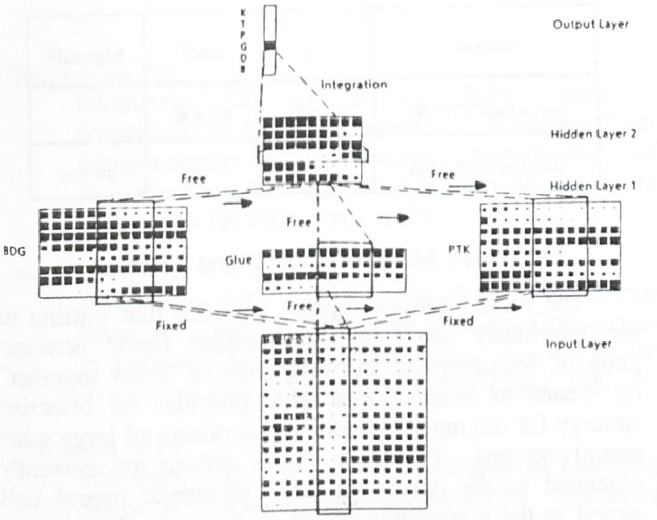

Figure 6: Combination of a BDG-net and a PTK-net using 4 additional units in hidden layer 1 as free "Connectionist Glue".

class distinctive features that were missing in our second experiment. In a fourth experiment, we have now examined an approach that allows for the network to be free to discover *any* additional features that might be useful to merge the two component networks. In stead of previously training a class distinctive network, we now add four units to hidden layer 1, whose connections to the input are free to learn any missing discriminatory features to supplement the 16 frozen BDG and PTK features. We call these units the " *connectionist glue*" that we apply to merge two distinct networks into a new combined net. This network is shown in Fig.6. The hidden units of hidden layer 1 from the BDG-net are shown on the left and those from the PTK-net on the right. The connections from the moving input window to these units have been trained individually on BDG- and PTK-data, respectively and -as before- remain fixed during combination learning. In the middle on hidden layer 1 we show the 4 free "Glue" units. Combination learning now finds an optimal combination of the existing BDG- and PTK-features and also supplements these by learning additional interclass discriminatory features. In doing so we have raised the number of connections to be trained to 8,000, which is only a small increase in number of connections (and learning time) over the original component nets. Performance evaluation of this network over the BDGPTK test database yielded a recognition rate of 98.4%.

3.2.5. All-Net Tuning

In addition to the techniques described so far, it may be useful to free *all* connections in a large modularly constructed network for an additional small amount of fine tuning. This has been done for the BDGPTK-net shown in Fig.6 yielding some additional perforn improvements. The resulting network finally act (over testing data) a recognition score of 98.6%.

3.3. Steps for the Design of Large Scale Neur Nets

Method	bdg	ptk	bdgp
Individual TDNNs	98.3 %	98.7 %	
TDNN:Max. Activation			60.5
Retrain BDGPTK			98.3
Retrain Combined Higher Layers			98.1
Retrain with V/UV-units			98.4
Retrain with Glue			98.4
All-Net Fine Tuning			98.6

Table 3-1: From BDG to BDGPTK; Modular Scaling Methods.

Table 3-1 summarizes the major results fro experiments. In the first row it shows the recog performance of the two initial TDNNs individually to perform the BDG- and the PTK respectively. Underneath, we show the results fr Hidden Markov Model, as discussed in the pr section. The third row shows that simply adding T and selecting the unit with the largest output act does not lead to acceptable performance (only correct). We have observed before that this is in negative consequence of inhibition in these net While inhibition of incorrect output categories le good, robust and confident performance, it erroneous results when additional networks are added without consideration of the interaction b them. We have then retrained a complete BDGP which achieves good recognition performance (correct), but found that it requires excessive amou training time. As an alternative, we have then ex three methods that exploit the hidden structu previously learned subcomponent networks, e.g BDG- and PTK-networks. With small additional t at the higher layers these networks could be merg achieve good recognition performance (98.1%). additional hidden units from a class dist voiced/unvoiced TDNN were added, recognition results improve to 98.4%. Similarly, through the application of "connectionist glue", a 98.4% performance score is achieved. Finally, when all the connections in the latter network are freed to perform small additional adjustments over a few additional training iterations, recognition results improve further to 98.6%.

The results indicate, that larger TDNNs can indeed be trained *incrementally*, without requiring excessive amounts of training and without loss in performance. In fact, the resulting incrementally trained networks appear to perform slightly better than the monolithically trained BDGPTK-net. Moreover, they achieve performance as high as the subcomponent BDG- and PTK-nets alone. As a strategy for the efficient construction of larger networks we have found the following concepts to be extremely effective: *modular,incremental learning, class distinctive learning, connectionist glue, partial and selective learning and all-net fine tuning*.

4. Extensions to Harder and Larger Tasks

To verify the general usefulness of the techniques described in the previous sections, we have now begun to experiment with tasks other than the stop consonants. What, for example, is the outcome when two subcategories are not as clearly separable by a potentially easily detectable and independent acoustic feature as might have been the case with the voicing distinction in our stop consonant experiments ? To answer this question, we have applied our techniques to the task of merging a voiced stop network (BDG) and a nasal network (M, N and syllabic nasals). It has been observed elsewhere [Waibel 87, Waibel 89], that the voiced stops "G" found in our Japanese database include numerous nasalized phoneme tokens (NG) depending on their position in the uttereance. During learning, our BDG-net successfully developed a complex non-linear decision surface to allow for both acoustic realizations as legal pronounciations of the voiced stop "G". In doing so the BDG-net has developed nasal features as cues to help discriminte a "G" from other stop consonants. When we attempt to combine a BDG-net with a nasal net, however, the nasal features of the BDG-net are then likely to conflict with those of an all nasal net. The burden of suitably merging these two nets therefore lies predominantly on hidden units acting as connectionist glue and their ability to fill in missing information and/or resolve conflicting information. This experiment has actually been carried out and we report its results in table 4-1. The top row shows again the recognition rate achieved by either network over testing data from the corresponding subclasses (voiced stop, nasal). The second row shows the recognition rate achieved by a merged net that employed connectionist glue as described in the previous section. A recognition score of 96.7% was achieved, which is again comparable to the performance of the original subcomponent nets.

Method	bdg	mnN	bdgmnN
Individual TDNNs	98.6 %	96.6 %	
Retrain with Glue			96.7 %

Table 4-1: Merging a Nasal- and a BDG-net

The results further support the idea that putting to use previously discovered abstractions (here: acoustic phonetic features) and allowing them to "grow together" by means of connectionist glue provides an effective strategy for the modular incremental design of large scale neural systems. In speech, these notions are currently extended to the design of large phonemic neural nets aimed at the recognition of *all* consonants. Preliminary results indicate that superior performance can be achieved for these systems as well [Waibel 88].

5. Conclusion

We summarize the major technical results from this work:

We have reported further experimental results from the use of Time Delay Neural Networks (TDNNs) for recognition in all major phonemic categories in a large vocabulary speech database and have measured *excellent recognition performance*. We believe, that the good performance results are due to the key properties of TDNNs, including: *shift invariance*, the proper representation of the *dynamic time-varying properties* of speech and the automatic discovery of *alternate, complementary internal features* of speech. These properties have been extensively documented elsewhere [Waibel 87, Waibel 89].

The serious problems associated with scaling smaller phonemic subcomponent networks to larger phonemic tasks are overcome by careful modular design. Modular design is achieved by several important strategies: *selective and incremental learning* of subcomponent tasks, *exploitation of previously learned hidden structure*, the application of *connectionist glue* or *class distinctive features* to allow for separate networks to "grow" together, *partial training* of portions of a larger net and finally, *all-net fine tuning* for making small additional adjustments in a large net.

Our findings suggest, that judicious application of a number of connectionist design techniques could lead to the successful design of high performance large scale connectionist speech recognition systems.

References

[Blumstein 79]
Blumstein, S.E. and Stevens, K.N. Acoustic Invariance in Speech Production: Evidence from Measurements of the Spectral Characteristics of Stop Consonants. *Journal of the Acoustical Society of America* 66:1001-1017, 1979.

[Blumstein 80]
Blumstein, S.E. and Stevens, K.N. Perceptual Invariance and Onset Spectra for Stop Consonants in Different Vowel Environments. *Journal of the Acoustical Society of America* 67:648-662, 1980.

[Fahlman 88]
Fahlman, S.E. *An Emprirical Study of Learning Speed in Back-Propagation Networks.* Technical Report CMU-CS-88-162, Carnegie-Mellon University, June, 1988.

[Haffner 88]
Haffner, P., Waibel, A. and Shikano, K. Fast Back-propagation Learning Methods for Neural Networks in Speech. In *Proceedings of the Fall Meeting of the Acoustical Society of Japan.* October, 1988. A more detailed version is in press as ATR-technical report.

[Kewley-Port 83]
Kewley-Port, D. Time Varying Features as Correlates of Place of Articulation in Stop Consonants. *Journal of the Acoustical Society of America* 73:322-335, 1983.

[Lang 87]
Lang, K. Connectionist Speech Recognition. July, 1987.PhD thesis proposal, Carnegie-Mellon University.

[Lippmann 87]
Lippmann, R.P. An Introduction to Computing with Neural Nets. *IEEE ASSP Magazine* :4-22, April, 1987.

[Makino 86]
Makino, S. and Kido, K. Phoneme Recognition Using Time Spectrum Pattern. *Speech Communication* :225-237, June, 1986.

[Rumelhart 86a]
Rumelhart, D.E. and McClelland, J.L. *Parallel Distributed Processing; Explorations in the Microstructure of Cognition.* MIT Press, Cambridge, MA, 1986.

[Rumelhart 86b]
Rumelhart, D.E., Hinton, G.E. and Williams, R.J. Learning Representations by Back-Propagating Errors. *Nature* 323:533-536, October, 1986.

[Sagisaka 87]
Sagisaka, Y., Takeda, K., Katagiri, S. and Kuwabara, H. *Japanese Speech Database with Fine Acoustic-Phonetic Transcriptions.* Technical Report, ATR Interpreting Telephony Research Laboratories, May, 1987.

[Waibel 87]
Waibel, A., Hanazawa, T., Hinton, G., Shikano, K. and Lang K. *Phoneme Recognition Using Time-Delay Neural Networks.* Technical Report TR-1-0006, ATR Interpreting Telephony Research Laboratories, October, 1987.

[Waibel 88]
Waibel, A., Sawai, H. and Shikano, K. *Modularity and Scaling in Large Phonemic Neural Networks.* Technical Report TR-I-0034, ATR Interpreting Telephony Research Laboratories, July, 1988. also under review for publication in the IEEE Transactions on Acoustics, Speech and Signal Processing.

[Waibel 89]
Waibel, A., Hanazawa, T., Hinton, G., Shikano, K. and Lang K. Phoneme Recognition Using Time-Delay Neural Networks. *IEEE, Transactions on Acoustics, Speech and Signal Processing* , March, 1989.

[Watrous 88]
Watrous, R. *Speech Recognition Using Connectionist Networks.* PhD thesis, University of Pennsylvania, September, 1988.