SP87−100

時間遅れ神経回路網（ＴＤＮＮ）による音韻認識

アレックス ワイベル（ＡＴＲ）

１９８７年１２月１８日

社団法人 電子情報通信学会

時間遅れ神経回路網(TDNN)による音韻認識

Phoneme Recognition Using Time-Delay Neural Networks

アレックス　ワイベル(ATR自動翻訳電話研究所)

Alex Waibel (ATR Interpreting Telephony Research Laboratories)

あらまし　本論文では、時間遅れ神経回路網(Time Delay Neural Network)を提案し、音韻認識に適用する。TDNNは、次の特徴を持っている。　(1) 3層の神経回路網構成であるので、任意の判別境界をバックプロパゲーションのアルゴリズムによる学習により構成することが原理的に可能である。(2) TDNNは、学習により、音韻のスペクトログラムから、音韻特徴を発見して、それに基づいて音韻を認識することを意図した構成になっている。　(3) TDNNは、入力音韻の位置ずれに、影響されない構成をとっている。

認識性能の評価のために、日本語の重要語5240単語に含まれる有声破裂音 /b/,/d/,/g/の識別実験を、3名の発声した音声を用いて話者ごとに行った。偶数番目の単語に含まれる有声破裂音で学習を行い、奇数番目の単語に含まれる有声破裂音の識別実験を行った。TDNNで、3名の平均で98.5%の音韻認識率が達成された。同じ条件でのHidden Markov モデルの音韻認識率は、93.7%であり、TDNNが非常に優れていることが確かめられた。さらに、TDNNで用いられている特徴を調べた結果、音韻弁別特徴として知られている第2フォルマントの動きや母音立ち上がり位置の検出を学習によって自動的に発見して、音韻識別の特徴の一部として用いていることが解った。また、/g/の鼻音化の異音も、TDNN内部に適切に表現されていることが解った。

Abstract　In this paper we present a Time Delay Neural Network (TDNN) approach to phoneme recognition which is characterized by two important properties: 1.) Using a 3-layer arrangement of simple computing units, a hierarchy can be constructed that allows for the formation of arbitrary nonlinear decision surfaces. The TDNN learns these decision surfaces automatically using error back-propagation[1]. 2.) The time-delay arrangement enables the network to discover acoustic-phonetic features and the temporal relationships between them independent of position in time and hence not blurred by temporal shifts in the input. As a recognition task, the speaker-dependent recognition of the phonemes "B", "D" and "G" in varying phonetic contexts was chosen. For comparison, several discrete Hidden Markov Models (HMM) were trained to perform the same task. Performance evaluation over 1946 testing tokens from three speakers showed that the TDNN achieves a recognition rate of 98.5% correct while the rate obtained by the best of our HMMs was only 93.7%. Closer inspection reveals that the network "invented" well-known acoustic-phonetic features (e.g., F2-rise, F2-fall, vowel-onset) as useful abstractions. It also developed alternate internal representations to link different acoustic realizations to the same concept.

1 Introduction

In recent years, the advent of new learning procedures and the availability of high speed parallel supercomputers have given rise to a renewed interest in connectionist models of intelligence[1]. These models are particularly interesting for cognitive tasks that require massive constraint satisfaction, i.e., the parallel evaluation of many clues and facts and their interpretation in the light of numerous interrelated constraints. Because of the far-reaching implications to speech recognition, neural networks have recently been compared with other pattern recognition classifiers[2] and with other speech recognition techniques (see [3] for review). Although a number of studies report encouraging recognition performance, superior performance figures in comparison to existing techniques are needed before neural networks can be considered as a viable alternative for speech recognition systems. One possible explanation for the mixed performance results obtained so far might be given by the inability of most neural network architectures to deal properly with the dynamic nature of speech. Two important aspects of this are for a network to represent temporal relationships between acoustic events, while at the same time providing for invariance under translation

in time. The specific movement of a formant in time, for example, is an important cue to determining the identity of a voiced stop, but it is irrelevant whether the same set of events occurs a little sooner or later in the course of time. Without translation invariance a neural net requires precise segmentation, to align the input pattern properly. Since this is not always possible in practice, learned features tend to get blurred (in order to accommodate slight misalignments) and their performance deteriorates.

In the present paper, we describe a Time Delay Neural Network (TDNN), which addresses both of these aspects. We demonstrate through extensive performance evaluation that superior recognition results can be achieved.

2 Time Delay Neural Networks

To be useful for speech recognition, a layered feed forward 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[2]. 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 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 describe a TDNN architecture that satisfies all of these criteria and is designed explicitly for the recognition of phonemes, in particular, the voiced stops "B", "D" and "G".

2.1 A TDNN for Phoneme Recognition

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[2,1]. 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[1,4].

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.

At the lowest level, 16 melscale spectral coefficients

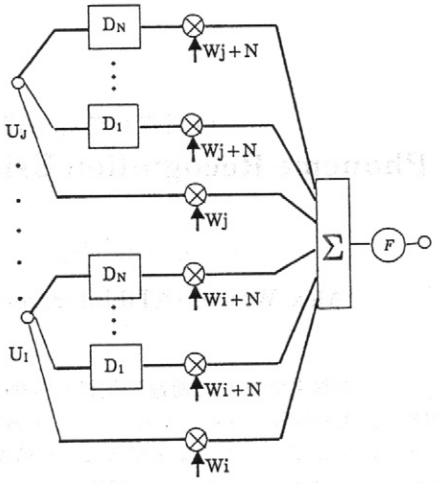

Figure 1: A Time Delay Neural Network (TDNN) unit

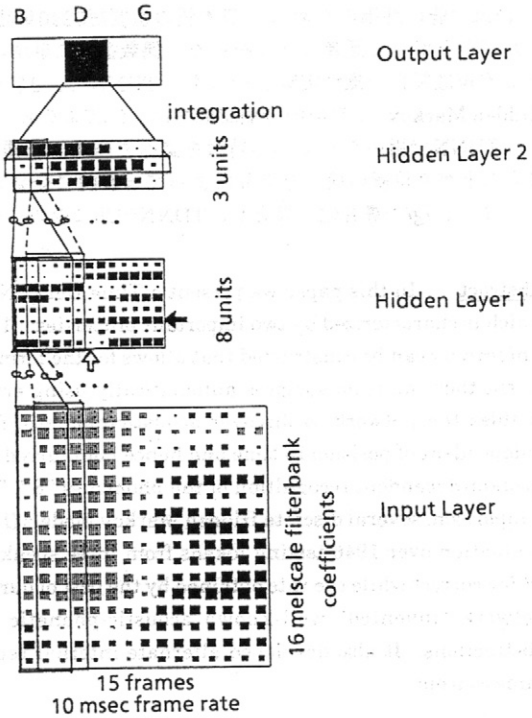

Figure 2: The TDNN architecture (input: "DA")

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[3] 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 "DA" 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 [5,6].

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 "D" 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.

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

2.2 Learning in a TDNN

Several learning techniques exist for optimization of neural networks[1,2]. For the present network we adopt the Back-propagation Learning Procedure[1,4]. This procedure iteratively adjusts all the weights in the network so as to decrease the error obtained at its output units. To arrive at a translation invariant network, 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[1]. 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.

The procedure described here is computationally rather expensive, due to the many iterations necessary for learning a complex multidimensional weight space and the number of learning samples. In our case, about 800 learning samples were used and between 20,000 and 50,000 iterations (step-size 0.002, momentum 0.1) of the back-propagation loop were run over all training samples. For greater learning speed, simulations were run on a 4 processor Alliant supercomputer and a staged learning strategy[3] was used to achieve faster convergence and good generalization. Learning still took about 4 days, but additional substantial increases in learning speed are possible[3]. Of course, this high computational cost applies only to learning. Recognition can easily be run in better than real-time.

3 Hidden Markov Models

As an alternative recognition approach we have implemented several Hidden Markov Models (HMM) aimed at phoneme recognition. HMMs are currently the most successful and promising approach [8,9,7] in speech recognition as they have been successfully applied to the whole spectrum of recognition tasks. HMMs' success is partially due to their ability to cope with the variability in speech by means of stochastic modeling. In the following sections, we describe the HMMs developed in our laboratory. They were aimed at phoneme recognition, more specifically the voiced stops "B", "D" and "G". More detail including results from experiments with variations on these models are given elsewhere[10,3] and we will restrict ourselves to a brief description of

[1]Note that in the experiments reported below these weight changes were actually carried out each time the error derivatives from all training samples had been computed[4].

our best configuration.

The acoustic front end for Hidden Markov Modeling is typically a vector quantizer that classifies sequences of short-time spectra. Input speech was sampled at 12kHz, preemphasized by (1 - 0.97 z^{-1}) and windowed using a 256-point Hamming window every 3 msec. Then a 12-order LPC analysis was carried out. A codebook of 256 LPC spectrum envelopes was generated from 216 phonetically balanced words. The Weighted Likelihood Ratio augmented with power values (PWLR)[11] was used as LPC distance measure for vector quantization. An HMM with four states and six transitions (the last state without a selfloop) was used in this study. The HMM probability values were trained using vector sequences of phonemes according to the forward-backward algorithm[7]. The vector sequences for "B", "D" and "G" include a consonant part and five frames of the following vowel. This is to model important transient informations, such as formant movement and has lead to improvements over context insensitive models[10]. The HMM was trained until convergence using about 250 phoneme tokens of vector sequences per speaker and phoneme. Typically, about 10 to 20 learning iterations were required for 256 tokens. A training run took about one hour on a VAX 8700. Floor values were set on the output probabilities to avoid errors caused by zero-probabilities. We have experimented with composite models, which were trained using a combination of context-independent and context-dependent probability values[9], but in our case no significant improvements were attained.

4 Recognition Experiments

We now turn to an experimental evaluation of the two techniques described in the previous sections. To provide a good framework for comparison, the same experimental conditions were given to both methods. For both, the same training data was used and both were tested on the same testing database as described below.

4.1 Experimental Conditions

For performance evaluation, we have used a large vocabulary database of 5240 common Japanese words[3]. These words were uttered in isolation by three male native Japanese speakers (MAU, MHT and MNM, all professional announcers). 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 phoneme recognition task chosen for this experiment was the recognition of the voiced stops, i.e., the phonemes "B", "D" and "G". The actual tokens were extracted from the utterances using manually selected acoustic-phonetic labels provided with the database[3]. For speaker MAU, for example, a total of 219 "B"s, 203 "D"s and 260 "G"s were extracted from the training and 227 "B"s, 179 "D"s and 252 "G"s from the testing data. Both recognition schemes, the TDNNs and the HMMs, were trained and tested speaker-dependently. Thus in both cases, separate networks were trained for each speaker.

In our database, no preselection of tokens was performed. All tokens labeled as one of the three voiced stops were included. It is important to note, that since the consonant tokens were extracted from entire utterances and *not* read in isolation, a significant amount of phonetic variability exists. Foremost, there is the variability introduced by the phonetic context out of which a token is extracted. Second, the position of a phonemic token within the utterance introduces additional variability[2]. Both of our recognition algorithms are only given the phonemic identity of a token and must find their own ways of representing the fine variations of speech. Since recognition results based on the training data are not meaningful, we report in the following only the results from open testing, i.e., from performance evaluation over the separate testing data set.

4.2 Results

Table 1 shows the results from the recognition experiments described above. As can be seen, for all three speakers, the TDNN yields considerable performance improvements over our HMM. Averaged over all three speakers, the error rate is reduced from 6.3% to 1.5%, a more than four fold reduction in error.

speaker	number of tokens	TDNN # errors	TDNN % correct	HMM # errors	HMM % correct
MAU	b(227)	4	98.8	18	92.9
	d(179)	3		6	
	g(252)	1		23	
MHT	b(208)	2	99.1	8	97.2
	d(170)	0		3	
	g(254)	4		7	
MNM	b(216)	11	97.5	27	90.9
	d(178)	1		13	
	g(256)	4		19	

Table 1: Recognition results for three speakers over test data using TDNN and HMM

Fig.3 and Fig.4 show scatter plots of the recognition outcome for the "D"s in the test data for speaker MAU, using the HMM and the TDNN. For the HMM (Fig.3), the log probability of the next best matching *incorrect* token is plotted against the log probability (normalized by number of frames) of the correct token, "D". In Fig.4, the activation levels from the TDNN's output units are plotted in the same fashion. The most striking observation that can be made from these plots is that the output units of a TDNN have a tendency to fire with high confidence as can be seen from the cluster of dots in the lower right hand corner of the scatter plots. Most output units tend to fire strongly for the correct phonemic class and not at all for any other, a property that is encouraged by the learning procedure. One possible consequence of this is that rejection thresholds could be introduced to improve recognition performance. If one were to eliminate among speaker MAU's tokens all those

[2]In Japanese, for example, a "G" is nasalized, when it occurs embedded in an utterance, but not in utterance initial position.

whose highest activation level is less than 0.5 and those which result in two or more closely competing activations (i.e., dots near the diagonal of the scatter plots; see[3] for complete set), 2.6% of all tokens would be rejected, while the remaining substitution error rate would be less than 0.46%.

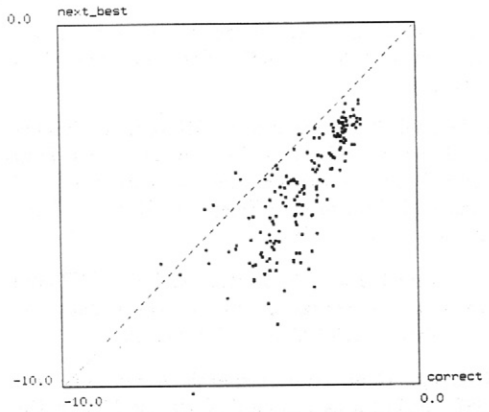

Figure 3: Scatter plot showing log probabilities for the best matching incorrect case vs. the correctly recognized "D"s using a HMM

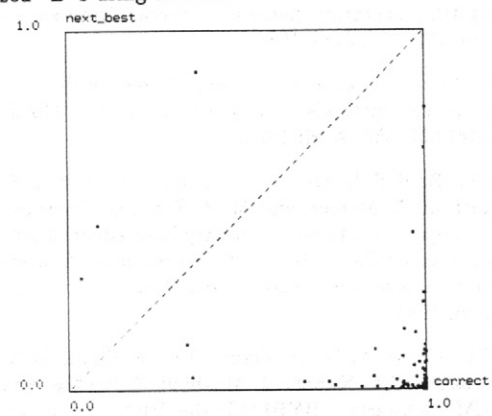

Figure 4: Scatter plot showing activation levels for the best matching incorrect case vs. the correctly recognized "D"s using a TDNN

4.3 The Learned Internal Representations of a TDNN

Given the encouraging performance of our TDNNs, a closer look at the learned internal representation of the network is warranted. Additional examples illustrating the observations in the following can be found in [3]. Fig.2 and the left side of Fig.5 show two typical instances of a "D" out of two different phonetic contexts ("DA" and "DO", respectively). In both cases, only the correct unit, the "D-output unit" fires strongly, despite the fact that the two input spectrograms differ considerably from each other. If we study the internal firings in these two cases we can see that the network has learned to use alternate internal representations to link variations in the sensory input to the same higher level concepts. A good example is given by the firings of the third and fourth hidden unit in the first layer above the input layer. As can be seen from Fig.5, the fourth hidden unit fires particularly strongly after vowel onset in the case of "DO", while the third unit shows stronger activation after vowel onset in the case of "DA".

Fig.6 shows the significance of these different firing patterns. Here the connection strengths for four of the eight moving TDNN units are shown, where white and black blobs represent positive and negative weights, respectively, and the magnitude of a weight is indicated by the size of the blob. In this figure, the time delays are displayed spatially as a 3 frame window of 16 spectral coefficients. Conceptually, the weights in this window form a moving acoustic-phonetic feature detector, that fires when the pattern for which it is specialized is encountered in the input speech. In our example, we can see that hidden unit number 4 (which was activated for "DO") has learned to fire when a falling (or rising) second formant starting at around 1600 Hz is found in the input (see filled arrow in Fig.6). As can be seen in Fig.5, this is the case for "DO" and hence the firing of hidden unit 4 after voicing onset (see row pointed to by the

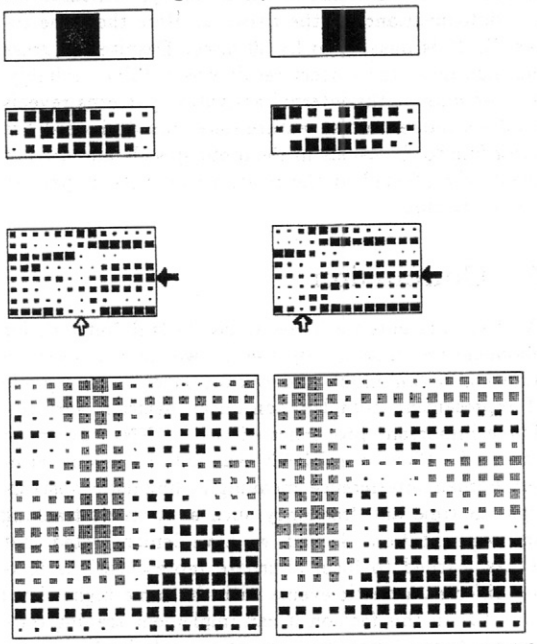

Figure 5: TDNN Activation patterns for centered and misaligned (30 msec) "DO"

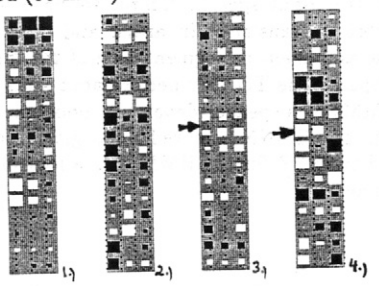

Figure 6: Weights on connections from 16 coefficients over 3 time frames to four of the 8 hidden units in the first layer

filled arrow in Fig.5). In the case of "DA" (see Fig.2) in turn, the second formant does not fall significantly, and hidden unit 3 (pointed to by the filled arrow) fires instead. From Fig.6 we can verify that TDNN-unit 3 has learned to look for a steady (or only slightly falling) second formant starting at about 1800 Hz. The connections in the second and third layer then link the different firing patterns observed in the first hidden layer into one and the same decision.

Another interesting feature can be seen in the bottom hidden unit in hidden layer number 1 (see Fig.2 and Fig.5, and compare with the weights of hidden unit 1 displayed in Fig.6). This unit has learned to take on the role of finding the segment boundary of the voiced stop. It does so in reverse polarity, i.e., it is always on *except* when the vowel onset of the voiced stop is encountered (see unfilled arrow in Fig.5(left) and Fig.2). Indeed, the higher layer TDNN-units subsequently use this "segmenter" to base the final decision on the occurrence of the right lower features at the right point in time. The right side of Fig.5, finally, demonstrates the shift-invariance of the network. Here the same token "DO" is misaligned by 30 msec. Despite the gross misalignment, the correct result was obtained reliably. A close look at the internal activation patterns reveals that the hidden units' feature detectors do indeed fire according to the events in the input speech, and are not negatively affected by the relative shift with respect to the input units.

5 Conclusion

We have presented a Time Delay Neural Network for phoneme recognition. By use of two hidden layers in addition to an input and output layer it is capable of representing complex non-linear decision surfaces. Three important properties of the TDNNs have been observed. First, our TDNN was able to invent without human interference meaningful linguistic abstractions in time and frequency such as formant tracking and segmentation. Second, we have demonstrated that it has learned to form alternate representations linking different acoustic events with the same higher level concept. In this fashion it can implement trading relations between lower level acoustic events leading to robust recognition performance despite considerable variability in the input speech. Third, we have seen that the network is translation-invariant and does not rely on precise alignment or segmentation of the input. We have compared the TDNN's performance with the best of our HMMs on a speaker-dependent phoneme recognition task. The TDNN achieved a recognition of 98.5% compared to 93.7% for the HMM, i.e., a fourfold reduction in error.

References

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

[2] R.P. Lippmann. An introduction to computing with neural nets. *IEEE ASSP Magazine*, 4–22, April 1987.

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

[4] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, October 1986.

[5] K. Lang. Connectionist speech recognition. July 1987. PhD thesis proposal, Carnegie-Mellon University.

[6] S. Makino and K. Kido. Phoneme recognition using time spectrum pattern. *Speech Communication*, 225–237, June 1986.

[7] F. Jelinek. Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64(4):532–556, April 1976.

[8] L. R. Bahl, S. K. Das, P. V. de Souza, F. Jelinek, S. Katz, R. L. Mercer, and M. A. Picheny. Some experiments with large-vocabulary isolated-word sentence recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, April 1984.

[9] Y.L. Chow, M.O. Dunham, O.A. Kimball, M.A. Krasner, G.F. Kubala, J. Makhoul, S. Roucos, and R.M. Schwartz. BYBLOS: the BBN continuous speech recognition system. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 89–92, April 1987.

[10] T. Hanazawa, T. Kawabata, and K. Shikano. Discrimination of japanese voiced stops using hidden markov model. In *Conference of the Acoustical Society of Japan*, pages 19–20, October 1987. (in Japanese).

[11] K Shikano. *Evaluation of LPC Spectral Matching Measures for Phonetic Unit Recognition.* Technical Report, Carnegie-Mellon University, May 1985.