Phoneme Recognition Using Time-Delay Neural Networks

ALEXANDER WAIBEL, MEMBER, IEEE, TOSHIYUKI HANAZAWA, GEOFFREY HINTON,
KIYOHIRO SHIKANO, MEMBER, IEEE, AND KEVIN J. LANG

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 backpropagation [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 percent correct while the rate obtained by the best of our HMM's was only 93.7 percent. 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.

I. 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]. Sometimes also referred to as artificial neural networks or parallel distributed processing models, 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. Cognitive tasks, such as vision, speech, language processing, and motor control, are also characterized by a high degree of uncertainty and variability and it has proved difficult to achieve good performance for these tasks using standard serial programming methods. Complex networks composed of simple computing units are attractive for these tasks not only because of their "brain-like" appeal[1] but because they offer ways for automatically designing systems that can make use of multiple interacting constraints. In general, such constraints are too complex to be easily programmed and require the use of automatic learning strategies. Such learning algorithms now exist (for an excellent review, see Lippman [2]) and have been demonstrated to discover interesting internal abstractions in their attempts to solve a given problem [1], [3]–[5]. Learning is most effective, however, when used in an architecture that is appropriate for the task. Indeed, applying one's prior knowledge of a task domain and its properties to the design of a suitable neural network model might well prove to be a key element in the successful development of connectionist systems.

Naturally, these techniques will have far-reaching implications for the design of automatic speech recognition systems, if proven successful in comparison to already-existing techniques. Lippmann [6] has compared several kinds of neural networks to other classifiers and evaluated their ability to create complex decision surfaces. Other studies have investigated actual speech recognition tasks and compared them to psychological evidence in speech perception [7] or to existing speech recognition techniques [8], [9]. Speech recognition experiments using neural nets have so far mostly been aimed at isolated word recognition (mostly the digit recognition task) [10]–[13] or phonetic recognition with predefined constant [14], [15] or variable phonetic contexts [16], [14], [17].

A number of these studies report very encouraging recognition performance [16], but only few comparisons to existing recognition methods exist. Some of these comparisons found performance similar to existing methods [9], [11], but others found that networks perform worse than other techniques [8]. One might argue that this state of affairs is encouraging considering the amount of fine-tuning that has gone into optimizing the more popular, established techniques. Nevertheless, better comparative performance figures are needed before neural networks can be considered as a viable alternative for speech recognition systems.

Manuscript received November 3, 1987; revised January 3, 1988.

A. Waibel is with the Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA 15213.

T. Hanazawa and K. Shikano are with ATR Interpreting Telephony Research Laboratories, Osaka, Japan.

G. Hinton is with the Department of Computer Science and the Department of Psychology, University of Toronto, Toronto, Ont., Canada, and the Canadian Institute for Advanced Research.

K. J. Lang is with Carnegie-Mellon University, Pittsburgh, PA 15213.

IEEE Log Number 8825654.

[1] The uninitiated reader should be cautioned not to overinterpret the now-popular term "neural network." Although these networks appear to mimic certain properties of neural cells, no claim can be made that present exploratory attempts simulate the complexities of the human brain.

0096-3518/89/0300-0328$01.00 © 1989 IEEE

One possible explanation for the mixed performance results obtained so far may be limitations in computing resources leading to shortcuts that limit performance. Another more serious limitation, however, is 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 general, shift invariance has been recognized as a critically important property for connectionist systems and a number of promising models have been proposed for speech and other domains [18]–[21], [14], [17], [22].

In the present paper, we describe a Time-Delay Neural Network (TDNN) which addresses both of these aspects of speech and demonstrate through extensive performance evaluation that superior recognition results can be achieved using this approach. In the following section, we begin by introducing the architecture and learning strategy of a TDNN aimed at phoneme recognition. Next, we compare the performance of our TDNN's to one of the more popular current recognition techniques: Hidden Markov Models (HMM). In Section III, we start by describing an HMM, under development at ATR [23], [24]. Both techniques, the TDNN and the HMM, are then evaluated over a testing database and we report the results. We show that substantially higher recognition performance is achieved by the TDNN than by the best of our HMM's. In Section IV, we then take a closer look at the internal representation that the TDNN learns for this task. It discovers a number of interesting linguistic abstractions which we show by way of examples. The implications of these results are then discussed and summarized in the final section of this paper.

II. Time-Delay Neural Networks (TDNN)

To be useful for speech recognition, a layered feedforward 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 nonlinear decision surfaces [2], [6]. 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 sufficiently 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."

A. A TDNN Architecture 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 nonlinear 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 to the past history of events. The sigmoid function was chosen as the nonlinear output function F due to its convenient mathematical properties [18], [5].

For the recognition of phonemes, a three layer net is constructed.[2] Its overall architecture and a typical set of activities in the units are shown in Fig. 2.

At the lowest level, 16 normalized 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 ms. Melscale coefficients were computed from the power spectrum by computing log energies in each melscale energy band [25], where adjacent coefficients in frequency overlap by one spectral sample and are smoothed by reducing the shared sample by 50 percent [25].[3] Adjacent coefficients in time were collapsed for further data reduction resulting in an overall 10 ms frame rate. All coefficients of an input token (in this case, 15 frames of speech centered around the hand-labeled vowel onset) were then normalized. This was accomplished by subtracting from each coefficient the average coefficient energy computed over all 15 frames of an input token and then normalizing each coefficient to lie between -1 and $+1$. All tokens in our database were preprocessed in the same fashion. Fig. 2 shows the resulting coefficients for the speech token "BA" as input to the

[2] Lippmann [2], [6] demonstrated recently that three layers can encode arbitrary pattern recognition decision surfaces. We believe that complex nonlinear decision surfaces are necessary to properly perform classification in the light of considerable acoustic variability as reported in the experiments below.

[3] Naturally, a number of alternative signal representations could be used as input, but have not been tried in this study. Filterbank coefficients were chosen as they are simple to compute and readily-interpretable in the light of acoustic-phonetics. The melscale is a physiologically motivated frequency scale that provides better relative frequency resolution for lower frequency bands. Our implementation resulted in coefficients with a bandwidth of approximately 190 Hz up to 1400 Hz, and with increasing bandwidths thereafter.

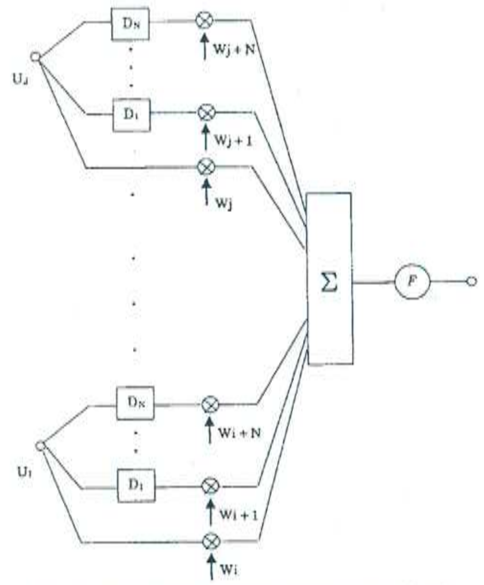

Fig. 1. A Time-Delay Neural Network (TDNN) unit.

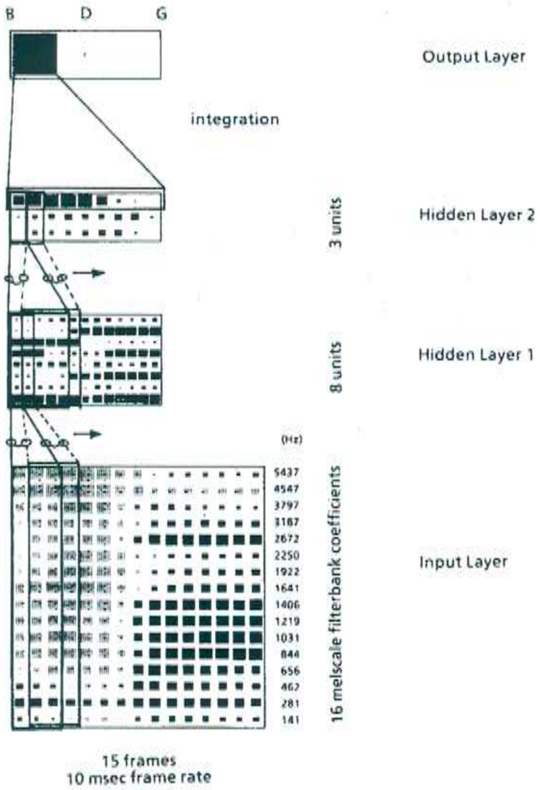

Fig. 2. The architecture of the TDNN.

network, where positive values are shown as black squares and negative values as gray 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 3 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 choice of 3 frames (30 ms) was motivated by earlier studies [26]–[29] that suggest that a 30 ms window might be sufficient to represent low level acoustic-phonetic events for stop consonant recognition. It was also the optimal choice among a number of alternative designs evaluated by Lang [21] on a similar task.

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 nonlinear (sigmoid function is applied here as well) TDNN unit which has fixed equal weights to a row of unit firings over time in hidden layer 2.[4]

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.[5] 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.

B. Learning in a TDNN

Several learning techniques exist for optimization of neural networks [1], [2], [30]. For the present network, we adopt the Backpropagation Learning Procedure [18],

[4] Note, however, that as for all units in this network (except the input units), the output units are also connected to a permanently active threshold unit. In this way, both an output unit's one shared connection to a row in hidden layer 2 and its dc-bias are learned and can be adjusted for optimal classification.

[5] Thus, 13 frames of activations in hidden layer 1 are generated when scanning the 15 frames of input speech with a 3 frame time delay window. Similarly, 9 frames are produced in hidden layer 2 from the 13 frames of activation in the layer below.

[5]. Mathematically, backpropagation is gradient descent of the mean-squared error as a function of the weights. The procedure performs two passes through the network. During the forward pass, an input pattern is applied to the network with its current connection strengths (initially small random weights). The outputs of all the units at each level are computed starting at the input layer and working forward to the output layer. The output is then compared to the desired output and its error calculated. During the backward pass, the derivative of this error is then propagated back through the network, and all the weights are adjusted so as to decrease the error [18], [5]. This is repeated many times for all the training tokens until the network converges to producing the desired output.

In the previous section, we described a method of expressing temporal structure in a TDNN and contrasted this method to training a network on a static input pattern (spectrogram), which results in shift sensitive networks (i.e., poor performance for slightly misaligned input patterns) as well as less crisp decision making in the units of the network (caused by misaligned tokens during training).

To achieve the desired learning behavior, we need to ensure 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 backpropagation 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 implement this, we first apply the regular backpropagation 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.[6] 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 error-prone preprocessing algorithms that otherwise would be needed for time alignment and/or segmentation. In Section IV-C, we will show examples of grossly misaligned patterns that are properly recognized due to this property.

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 of the backpropagation loop were run over all training samples. Two steps were taken to perform learning within reasonable time. First, we have implemented our learning procedure in C and Fortran on a 4 processor Alliant supercomputer. The speed of learning can be improved considerably by computing the forward and backward sweeps for several different training samples in parallel on different processors. Further improvements can be gained by vectorizing operations and possibly assembly coding the innermost loop. Our present implementation achieves about a factor of 9 speedup over a VAX 8600, but still leaves room for further improvements (Lang [21], for example, reports a speedup of a factor of 120 over a VAX11/780 for an implementation running on a Convex supercomputer). The second step taken toward improving learning time is given by a staged learning strategy. In this approach, we start optimizing the network based on 3 prototypical training tokens only.[7] In this case, convergence is achieved rapidly, but the network will have learned a representation that generalizes poorly to new and different patterns. Once convergence is achieved, the network is presented with approximately twice the number of tokens and learning continues until convergence.

Fig. 3 shows the progress during a typical learning run. The measured error is 1/2 the squared error of all the output units, normalized for the number of training tokens. In this run, the number of training tokens used were 3, 6, 9, 24, 99, 249, and 780. As can be seen from Fig. 3, the error briefly jumps up every time more variability is introduced by way of more training data. The network is then forced to improve its representation to discover clues that generalize better and to deemphasize those that turn out to be merely irrelevant idiosyncracies of a limited sample set. Using the full training set of 780 tokens, this particular run was continued until iteration 35 000 (Fig. 3 shows the learning curve only up to 15 000 iterations). With this full training set, small learning steps have to be taken and learning progresses slowly. In this case, a step size of 0.002 and a momentum [5] of 0.1 was used. The staged learning approach was found to be useful to move the weights of the network rapidly into the neighborhood of a reasonable solution, before the rather slow fine tuning over all training tokens begins.

Despite these speedups, learning runs still take in the

[6] 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 [5].

[7] Note that for optimal learning, the training data are presented by always alternating tokens for each class. Hence, we start the network off by presenting 3 tokens, one for each class.

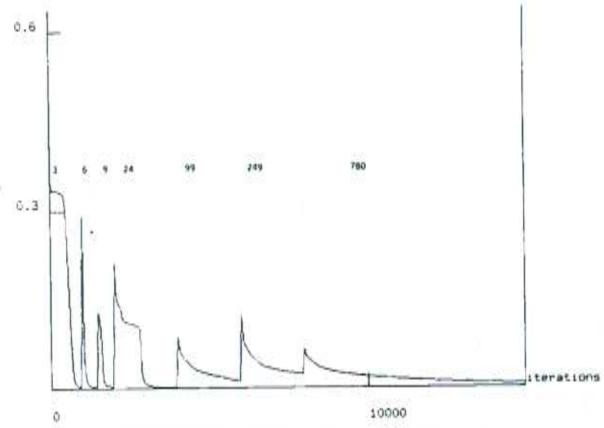

Fig. 3. TDNN output error versus number of learning iterations (increasing training set size).

order of several days. A number of programming tricks [21] as well as modifications to the learning procedure [31] are not implemented yet and could yield another factor of 10 or more in learning time reduction. It is important to note, however, that the amount of computation considered here is necessary *only for learning* of a TDNN and *not for recognition.* Recognition can easily be performed in better than real time on a workstation or personal computer. The simple structure makes TDNN's also well suited for standardized VLSI implementation. The detailed knowledge could be learned "off-line" using substantial computing power and then downloaded in the form of weights onto a real-time production network.

III. Recognition Experiments

We now turn to an experimental evaluation of the TDNN's recognition performance. In particular, we would like to compare the TDNN's performance to the performance of the currently most popular recognition method: Hidden Markov Models (HMM). For the performance evaluation reported here, we have chosen the best of a number of HMM's developed in our laboratory. Several other HMM-based variations and models have been tried in an effort to optimize our HMM, but we make no claim that an exhaustive evaluation of all HMM-based techniques was accomplished. We should also point out that the experiments reported here were aimed at evaluating two different *recognition philosophies.* Each recognition method was therefore implemented and optimized using its preferred representation of the speech signal, i.e., a representation that is well suited and most commonly used for the method evaluated. Evaluation of both methods was of course carried out using the same speech input data, but we caution the reader that due to the differences in representation, the exact contribution to overall performance of the recognition strategy as opposed to its signal representation is not known. It is conceivable that improved front end processing might lead to further performance improvements for either technique. In the following sections, we will start by introducing the best of our Hidden Markov Models. We then describe the experimental conditions and the database used for performance evaluation and conclude with the performance results achieved by our TDNN and HMM.

A. A Hidden Markov Model (HMM) for Phoneme Recognition

HMM's are currently the most successful and promising approach [32]–[34] in speech recognition as they have been successfully applied to the whole range of recognition tasks. Excellent performance was achieved at all levels from the phonemic level [35]–[38] to word recognition [39], [34] and to continuous speech recognition [40]. The success of HMM's is partially due to their ability to cope with the variability in speech by means of stochastic modeling. In this section, we describe an HMM developed in our laboratory that was aimed at phoneme recognition, more specifically the voiced stops "B," "D," and "G." The model described was the best of a number of alternate designs developed in our laboratory [23], [24].

The acoustic front end for Hidden Markov Modeling is typically a vector quantizer that classifies sequences of short-time spectra. Such a representation was chosen as it is highly effective for HMM-based recognizers [40].

Input speech was sampled at 12 kHz, preemphasized by $(1 - 0.97\, z^{-1})$, and windowed using a 256-point Hamming window every 3 ms. 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 [41], [42] augmented with power values (PWLR) [43], [42] was used as LPC distance measure for vector quantization.

A fairly standard HMM was adopted in this paper as shown in Fig. 4. It has four states and six transitions and was found to be the best of a series of alternate models tried in our laboratory. These included models with two, three, four, and five states and with tied arcs and null arcs [23], [24].

The HMM probability values were trained using vector sequences of phonemes according to the forward–backward algorithm [32]. 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 information, such as formant movement, and has lead to improvements over context insensitive models [23], [24]. Again, variations on these parameters have been tried for the discrimination of these three voiced stop consonants. In particular, we have used 10 and 15 frames (i.e., 30 and 45 ms) of the following vowel in a 5 state HMM, but no performance improvements over the model described were obtained.

The HMM was trained using about 250 phoneme tokens of vector sequences per speaker and phoneme (see details of the training database below). Fig. 5 shows for a typical training run the average log probability normalized by the number of frames. Training was continued

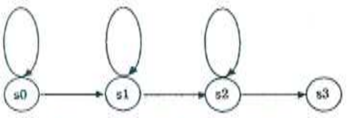

Fig. 4. Hidden Markov Model.

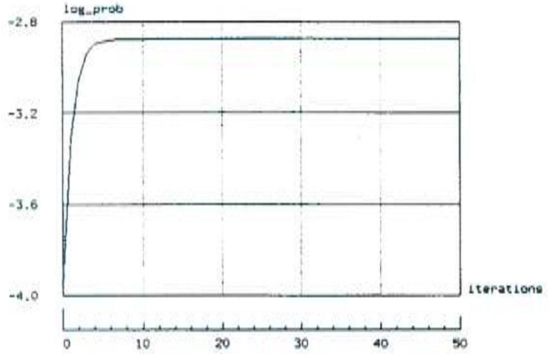

Fig. 5. Learning in the Hidden Markov Model.

until the increase of the average log probability between iterations became less than $2 * 10^{-3}$.

Typically, about 10–20 learning iterations are required for 256 tokens. A training run takes about 1 h on a VAX 8700. Floor values[8] 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 as suggested by Schwartz *et al.* [35], [36]. In our case, no significant improvements were attained.

B. Experimental Conditions

For performance evaluation, we have used a large vocabulary database of 5240 common Japanese words [44]. These words were uttered in isolation by three male native Japanese speakers (MAU, MHT, and MNM, all professional announcers) in the order they appear in a Japanese dictionary. 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 (the even numbered files as derived from the recording order) and a testing set (the odd numbered files). A given speaker's training and testing data, therefore, consisted 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 [44]. 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 TDNN's and the HMM's, 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. The actual signal of a "BA" will therefore look significantly different from a "BI" and so on. Second, the position of a phonemic token within the utterance introduces additional variability. In Japanese, for example, a "G" is nasalized, when it occurs embedded in an utterance, but not in utterance initial position. 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.

C. Results

Table I shows the results from the recognition experiments described above as obtained from the *testing data.* As can be seen, for all three speakers, the TDNN yields considerably higher performance than our HMM. Averaged over all three speakers, the error rate is reduced from 6.3 to 1.5 percent—a more than fourfold reduction in error.

While it is particularly important here to base performance evaluation on testing data,[9] a few observations can be made from recognition runs over the training data. For the training data set, recognition error rates were: 99.6 percent (MAU), 99.7 percent (MHT), and 99.7 percent (MNM) for the TDNN, and 96.9 percent (MAU), 99.1 percent (MHT), and 95.7 percent (MNM) for the HMM. Comparison of these results to those from the testing data in Table I indicates that both methods achieved good generalization from the training set to unknown data. The data also suggest that better classification rather than better generalization might be the cause of the TDNN's better performance shown in Table I.

Figs. 6–11 show scatter plots of the recognition outcome for the test data for speaker MAU, using the HMM and the TDNN. For the HMM (see Figs. 6–8), the log probability of the next best matching *incorrect* token is plotted against the log probability[10] of the correct token, e.g., "B," "D," and "G." In Figs. 9–11, the activation levels from the TDNN's output units are plotted in the same fashion. Note that these plots are not easily comparable, as the two recognition methods have been trained in quite different ways. They do, however, represent the

[8] Here, once again, the optimal value out of a number of alternative choices was selected.

[9] If the training data are insufficient, neural networks can in principle learn to *memorize* training patterns rather than finding generalization of speech.

[10] Normalized by number of frames.

TABLE I
RECOGNITION RESULTS FOR THREE SPEAKERS OVER TEST DATA USING TDNN AND HMM

speaker	number of tokens	number of errors	recognition rate	TDNN	number of errors	recognition rate	HMM
MAU	b(227)	4	98.2	98.8	18	92.1	92.9
	d(179)	3	98.3		6	96.7	
	g(252)	1	99.6		23	90.9	
MHT	b(208)	2	99.0	99.1	8	96.2	97.2
	d(170)	0	100		3	98.2	
	g(254)	4	98.4		7	97.2	
MNM	b(216)	11	94.9	97.5	27	87.5	90.9
	d(178)	1	99.4		13	92.7	
	g(256)	4	98.4		19	92.6	

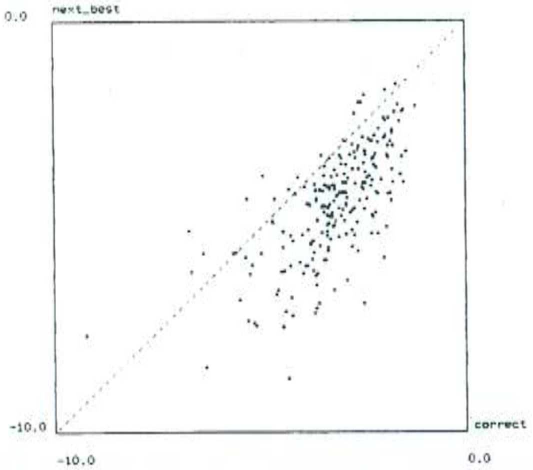

Fig. 6. Scatter plot showing log probabilities for the best matching incorrect case versus the correctly recognized "B's" using an HMM.

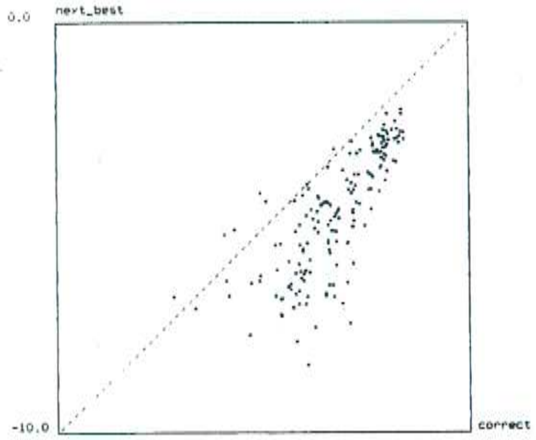

Fig. 7. Scatter plot showing log probabilities for the best matching incorrect case versus the correctly recognized "D's" using an HMM.

numerical values that each method's decision rule uses to determine the recognition outcome. We present these plots here to show some interesting properties of the two techniques. 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

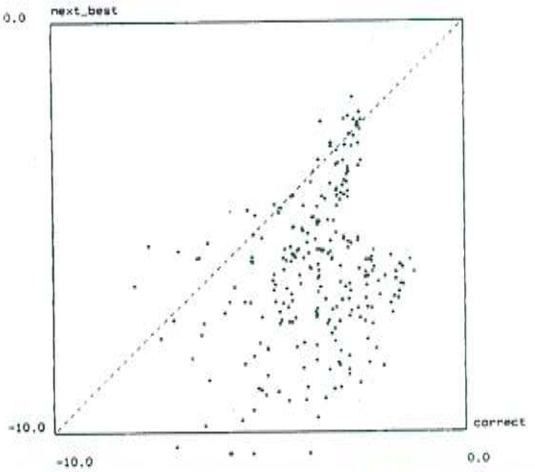

Fig. 8. Scatter plot showing log probabilities for the best matching incorrect case versus the correctly recognized "G's" using an HMM.

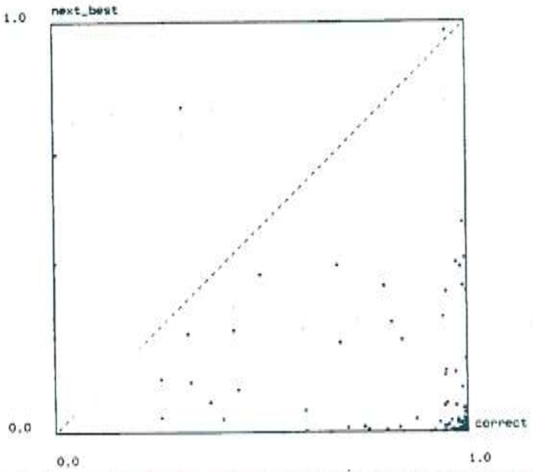

Fig. 9. Scatter plot showing activation levels for the best matching incorrect case versus the correctly recognized "B's" using a TDNN.

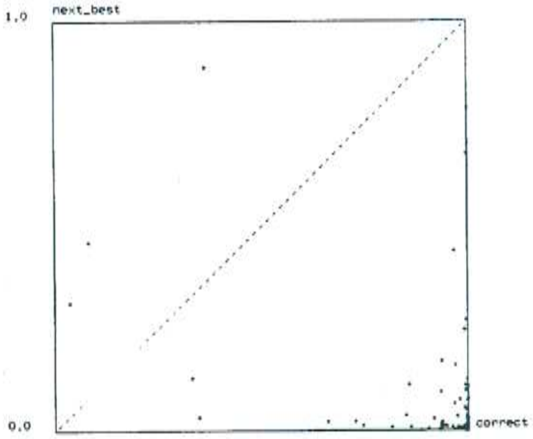

Fig. 10. Scatter plot showing activation levels for the best matching incorrect case versus the correctly recognized "D's" using a TDNN.

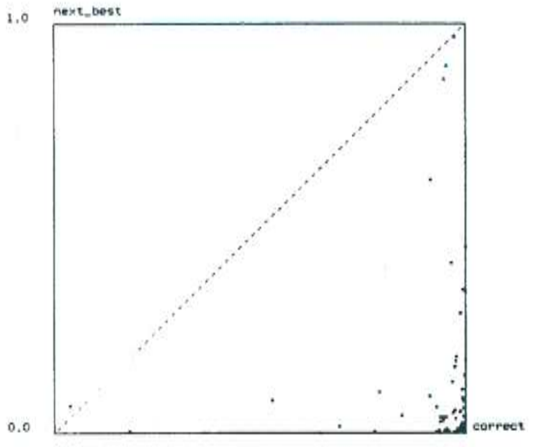

Fig. 11. Scatter plot showing activation levels for the best matching incorrect case versus the correctly recognized "G's" using a TDNN.

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 whose highest activation level is less than 0.5 and those which result in two or more closely competing activations (i.e., are near the diagonal in the scatter plots), 2.6 percent of all tokens would be rejected, while the remaining substitution error rate would be less than 0.46 percent.

IV. The Learned Internal Representations of a TDNN

Given the encouraging performance of our TDNN's, a closer look at the learned internal representation of the network is warranted. What are the properties or abstractions that the network has learned that appear to yield a very powerful description of voiced stops? Figs. 12 and 13 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. 13, 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."

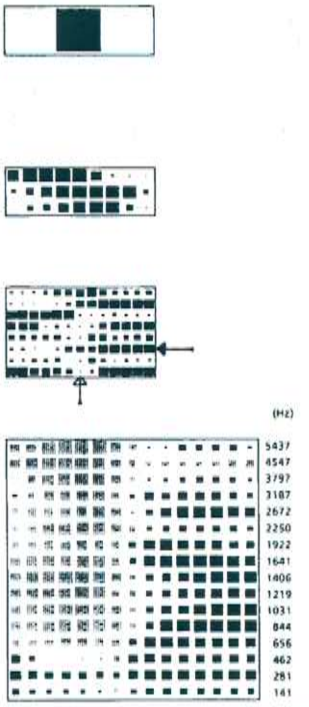

Fig. 12. TDNN activation patterns for "DA."

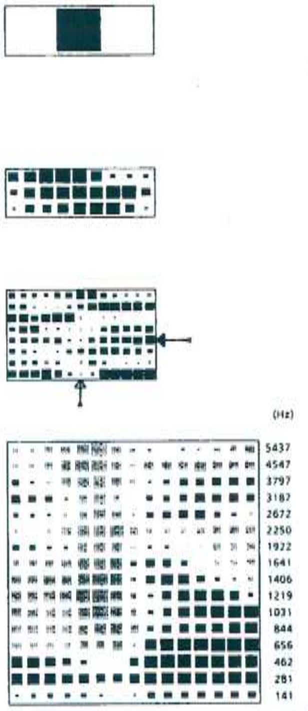

Fig. 13. TDNN activation patterns for "DO."

Fig. 14 shows the significance of these different firing patterns. Here the connection strengths for 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

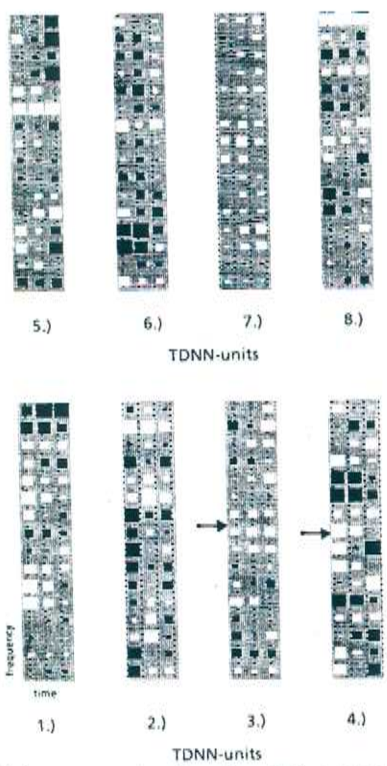

Fig. 14. Weights on connections from 16 coefficients over 3 time frames to each of the 8 hidden units in the first layer.

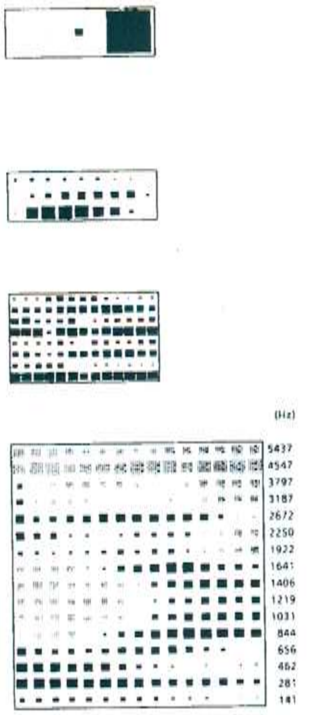

Fig. 15. TDNN activation patterns for "GA" embedded in an utterance.

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. 14). As can be seen in Fig. 13, this is the case for "DO" and hence the firing of hidden unit 4 after voicing onset (see row pointed to by the filled arrow in Fig. 13). In the case of "DA" (see Fig. 12), in turn, the second formant does not fall significantly, and hidden unit 3 (pointed to by the filled arrow) fires instead. From Fig. 14 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 Figs. 12 and 13, and compare them to the weights of hidden unit 1 displayed in Fig. 14). 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 Figs. 13 and 12). 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.

In the previous example, we have seen that the TDNN can account for variations in phonetic context. Figs. 15 and 16 show examples of variability caused by the relative position of a phoneme within a word. In Japanese, a "G" embedded in a word tends to be nasalized as seen in the spectrum of a "GA" in Fig. 15. Fig. 16 shows a word initial "GA." Despite the striking differences between these two input spectrograms, the network's internal alternate representations manage to produce in both cases crisp output firings for the right category.

Figs. 17 and 18, finally, demonstrate the shift invariance of the network. They show the same token "DO" of Fig. 13, misaligned by +30 ms and −30 ms, respectively. Despite the gross misalignment (note that significant transitional information is lost by the misalignment in Fig. 18), 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. Naturally, error rates will gradually increase when the tokens are artificially shifted to an extent that important features begin to fall outside the 15 frame data window considered here. We have observed, for example, a 2.6 percent increase in error rate when all tokens from

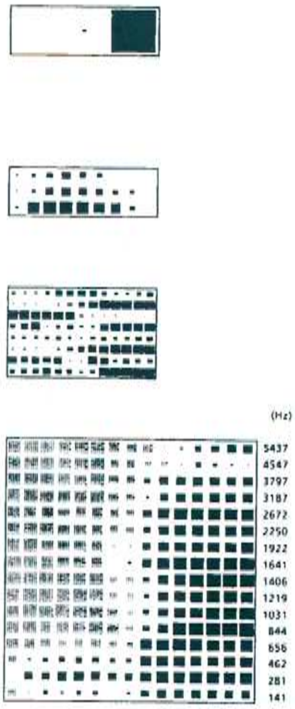

Fig. 16. TDNN activation patterns for "GA" in utterance initial position.

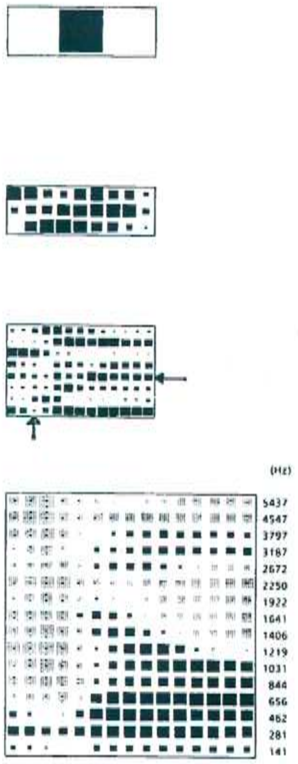

Fig. 17. TDNN activation patterns for "DO" misaligned by +30 ms.

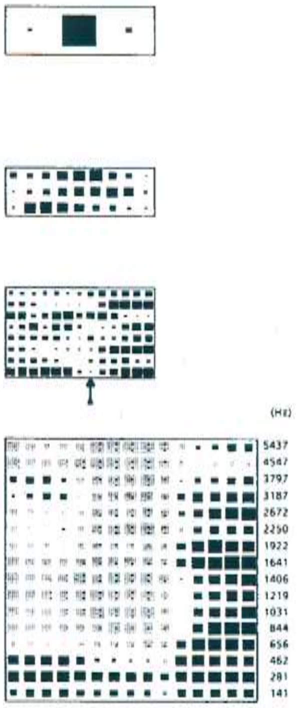

Fig. 18. TDNN activation patterns for "DO" misaligned by −30 ms.

the training data were artificially shifted by 20 ms. Such residual time-shift sensitivities are due to the edge effects at the token boundaries and can probably be removed by training the TDNN using randomly shifted training tokens.[11] We also consider the formation of shift-invariant *internal* features to be the important desirable property we observe in the TDNN. Such internal features could be incorporated into larger speech recognition systems using more sophisticated search techniques or a syllable or word level TDNN, and hence could replace the simple integration layer we have used here for training and evaluation.

Three important properties of the TDNN's have been observed. First, our TDNN was able to learn, without human interference, meaningful linguistic abstractions 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. Third, we have seen that the network is shift invariant and does not rely on precise alignment or segmentation of the input.

V. Conclusion and Summary

In this paper we have presented a Time-Delay Neural Network (TDNN) approach to phoneme recognition. We

[11] We gratefully acknowledge one of the reviewers for suggesting this idea.

have shown that this TDNN has two desirable properties related to the dynamic structure of speech. First, it can learn the temporal structure of acoustic events and the temporal relationships between such events. Second, it is translation invariant, that is, the features learned by the network are insensitive to shifts in time. Examples demonstrate that the network was indeed able to learn acoustic-phonetic features, such as formant movements and segmentation, and use them effectively as internal abstractions of speech.

The TDNN presented here has two hidden layers and has the ability to learn complex nonlinear decision surfaces. This could be seen from the network's ability to use alternate internal representations and trading relations among lower level acoustic-phonetic features, in order to arrive robustly at the correct final decision. Such alternate representations have been particularly useful for representing tokens that vary considerably from each other due to their different phonetic environment or their position within the original speech utterance.

Finally, we have evaluated the TDNN on the recognition of three acoustically similar phonemes, the voiced stops "B," "D," and "G." In extensive performance evaluation over testing data from three speakers, the TDNN achieved an average recognition score of 98.5 percent. For comparison, we have applied various Hidden Markov Models to the same task and only been able to recognize 93.7 percent of the tokens correctly. We would like to note that many variations of HMM's have been attempted, and many more variations of both HMM's and TDNN's are conceivable. Some of these variations could potentially lead to significant improvements over the results reported in this study. Our goal here is to present TDNN's as a new and successful approach for speech recognition. Their power lies in their ability to develop shift-invariant internal abstractions of speech and to use them in trading relations for making optimal decisions. This holds significant promise for speech recognition in general, as it could help overcome the representational weaknesses of speech recognition systems faced with the uncertainty and variability in real-life signals.

Acknowledgment

The authors would like to express their gratitude to Dr. A. Kurematsu, President of ATR Interpreting Telephony Research Laboratories, for his enthusiastic encouragement and support which made this research possible. We are also indebted to the members of the Speech Processing Department at ATR and Mr. Fukuda at Apollo Computer, Tokyo, Japan, for programming assistance in the various stages of this research.

References

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

[2] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Mag.*, vol. 4, Apr. 1987.

[3] D. C. Plaut, S. J. Nowlan, and G. E. Hinton, "Experiments on learning by back propagation," Tech. Rep. CMU-CS-86-126, Carnegie-Mellon Univ., June 1986.

[4] T. J. Sejnowski and C. R. Rosenberg, "NETtalk: A parallel network that learns to read aloud," Tech. Rep. JHU/EECS-86/01, Johns Hopkins Univ., June 1986.

[5] D. E. Rumelhart, G.E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533-536, Oct. 1986.

[6] W. Y. Huang and R. P. Lippmann, "Comparison between neural net and conventional classifiers," in *Proc. IEEE Int. Conf. Neural Networks*, June 1987.

[7] J. L. McClelland and J. L. Elman, *Interactive Processes in Speech Perception: The TRACE Model*. Cambridge, MA: M.I.T. Press, 1986, ch. 15, pp. 58-121.

[8] S. M. Peeling, R. K. Moore, and M. J. Tomlinson, "The multi-layer perceptron as a tool for speech pattern processing research," in *Proc. IoA Autumn Conf. Speech Hearing*, 1986.

[9] H. Bourlard and C. J. Wellekens, "Multilayer perceptrons and automatic speech recognition," in *Proc. IEEE Int. Conf. Neural Networks*, June 1987.

[10] B. Gold, R. P. Lippmann, and M. L. Malpass, "Some neural net recognition results on isolated words," in *Proc. IEEE Int. Conf. Neural Networks*, June 1987.

[11] R. P. Lippmann and B. Gold, "Neural-net classifiers useful for speech recognition," in *Proc. IEEE Int. Conf. Neural Networks*, June 1987.

[12] D. J. Burr, "A neural network digit recognizer," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 1986.

[13] D. Lubensky, "Learning spectral-temporal dependencies using connectionist networks," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Apr. 1988.

[14] R. L. Watrous and L. Shastri, "Learning phonetic features using connectionist networks: An experiment in speech recognition," in *Proc. IEEE Int. Conf. Neural Networks*, June 1987.

[15] R. W. Prager, T. D. Harrison, and F. Fallside, "Boltzmann machines for speech recognition," *Comput., Speech, Language*, vol. 3, no. 27, Mar. 1986.

[16] J. L. Elman and D. Zipser, "Learning the hidden structure of speech," Tech. Rep., Univ. Calif., San Diego, Feb. 1987.

[17] R. L. Watrous, L. Shastri, and A. H. Waibel, "Learned phonetic discrimination using connectionist networks," in *Proc. Euro. Conf. Speech Technol.*, Edinburgh, Sept. 1987, pp. 377-380.

[18] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning Internal Representations by Error Propagation*. Cambridge, MA: M.I.T. Press, 1986, ch. 8, pp. 318-362.

[19] J. S. Bridle and R. K. Moore, "Boltzmann machines for speech pattern processing," in *Proc. Inst. Acoust. 1984*, 1984, 315-322.

[20] D. W. Tank and J. J. Hopfield, "Neural computation by concentrating information in time," in *Proc. Nat. Academy Sci.*, Apr. 1987, pp. 1896-1900.

[21] K. Lang, "Connectionist speech recognition," Ph.D. dissertation proposal, Carnegie-Mellon Univ., Pittsburgh, PA.

[22] K. Fukushima, S. Miyake, and T. Ito, "Neocognitron: A neural network model for a mechanism of visual pattern recognition," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, pp. 826-834, Sept./Oct. 1983.

[23] T. Hanazawa, T. Kawabata, and K. Shikano, "Discrimination of Japanese voiced stops using Hidden Markov Model," in *Proc. Conf. Acoust. Soc. Japan*, Oct. 1987, pp. 19-20 (in Japanese).

[24] —, "Recognition of Japanese voiced stops using Hidden Markov Models," IEICE Tech. Rep., Dec. 1987 (in Japanese).

[25] A. Waibel and B. Yegnanarayana, "Comparative study of nonlinear time warping techniques in isolated word speech recognition systems," Tech. Rep., Carnegie-Mellon Univ., June 1981.

[26] S. Makino and K. Kido, "Phoneme recognition using time spectrum pattern," *Speech Commun.*, pp. 225-237, June 1986.

[27] S. E. Blumstein and K. N. Stevens, "Acoustic invariance in speech production: Evidence from measurements of the spectral characteristics of stop consonants," *J. Acoust. Soc. Amer.*, vol. 66, pp. 1001-1017, 1979.

[28] —, "Perceptual invariance and onset spectra for stop consonants in different vowel environments," *J. Acoust. Soc. Amer.*, vol. 67, pp. 648-662, 1980.

[29] D. Kewley-Port, "Time varying features as correlates of place of articulation in stop consonants," *J. Acoust. Soc. Amer.*, vol. 73, pp. 322-335, 1983.

[30] G. E. Hinton, "Connectionist learning procedures," *Artificial Intelligence*, 1987.

[31] M. A. Franzini, "Speech recognition with back propagation," in *Proc. 9th Annu. Conf. IEEE/Eng. Med. Biol. Soc.*, Nov. 1987.

[32] F. Jelinek, "Continuous speech recognition by statistical methods," *Proc. IEEE*, vol. 64, pp. 532–556, Apr. 1976.

[33] J. K. Baker, "Stochastic modeling as a means of automatic speech recognition," Ph.D. dissertation, Carnegie-Mellon Univ., Apr. 1975.

[34] 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 *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Apr. 1984.

[35] R. Schwartz, Y. Chow, O. Kimball, S. Roucos, M. Krasner, and J. Makhoul, "Context-dependent modeling for acoustic-phonetic recognition of continuous speech," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Apr. 1985.

[36] A.-M. Derouault, "Context-dependent phonetic Markov models for large vocabulary speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Apr. 1987, pp. 360–363.

[37] K. F. Lee and H. W. Hon, "Speaker-independent phoneme recognition using hidden Markov models," Tech. Rep. CMU-CS-88-121, Carnegie-Mellon Univ., Pittsburgh, PA, Mar. 1988.

[38] P. Brown, "The acoustic-modeling problem in automatic speech recognition," Ph.D. dissertation, Carnegie-Mellon Univ., May 1987.

[39] L. R. Rabiner, B. H. Juang, S. E. Levinson, and M. M. Sondhi, "Recognition of isolated digits using hidden Markov models with continuous mixture densities," *AT&T Tech. J.*, vol. 64, no. 6, pp. 1211–1233, July–Aug. 1985.

[40] 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 *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Apr. 1987, pp. 89–92.

[41] M. Sugiyama and K. Shikano, "LPC peak weighted spectral matching measures," Inst. Elec. Commun. Eng. Japan, vol. 64-A, no. 5, pp. 409–416, 1981 (in Japanese).

[42] K. Shikano, "Evaluation of LPC spectral matching measures for phonetic unit recognition," Tech. Rep., Carnegie-Mellon Univ., May 1985.

[43] K. Aikawa and K. Shikano, "Spoken word recognition using vector quantization in power-spectrum vector space," Inst. Elec. Commun. Eng. Japan, vol. 68-D, no. 3, Mar. 1985 (in Japanese).

[44] Y. Sagisaka, K. Takeda, S. Katagiri, and H. Kuwabara, "Japanese speech database with fine acoustic-phonetic transcriptions," Tech. Rep., ATR Interpreting Telephony Res. Lab., May 1987.

Alexander Waibel (S'79–M'86) was born in Heidelberg, West Germany, on May 2, 1956. He received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer science in 1979, 1980, and 1986 from the Massachusetts Institute of Technology, Cambridge, and from Carnegie-Mellon University, Pittsburgh, respectively.

From 1980 to 1985 he was a member of the Computer Science Research Staff at Carnegie-Mellon University. In 1986 he joined the Faculty of the Computer Science Department as Research Associate and is now a Research Computer Scientist with joined responsibilities in the Center for Machine Translation at Carnegie-Mellon and at the ATR Interpreting Telephony Research Laboratories in Osaka, Japan. From May 1987 to July 1988, he worked as Invited Research Scientist at the ATR Interpreting Telephony Research Laboratories in Osaka, Japan. He has published and lectured extensively on speech recognition and related areas. His current research interests include speech recognition and synthesis, neurocomputing, machine learning, and machine translation.

Dr. Waibel is a member of the IEEE Acoustics, Speech, and Signal Processing Society, the IEEE Computer Society, the Acoustical Society of America, the Association for Computational Linguistics, and the International Neural Network Society.

Toshiyuki Hanazawa was born in Japan on March 25, 1962. He received the B.S. degree in physics from Tokyo Metropolitan University, Tokyo, Japan, in 1985.

In 1985 he joined the Information Systems and Electronics Development Laboratory, Mitsubishi Electric Corporation, Kamakura, Japan, where he was engaged in speech recognition research. Since 1987 he has been with the ATR Interpreting Telephony Research Laboratories, Osaka, Japan, where he is currently engaged in speech recognition research.

Geoffrey Hinton received the B.A. degree in psychology from the University of Cambridge in 1970 and the Ph.D. degree in artificial intelligence from the University of Edinburgh in 1978.

He is a Professor in the Departments of Computer Science and Psychology at the University of Toronto, and a Fellow of the Canadian Institute for Advanced Research. His research interests include methods of using connectionist networks for learning, memory, perception, symbol processing, and motor control.

Kiyohiro Shikano (M'84) was born in Japan on October 12, 1947. He received the B.E., M.E., and the Ph.D. degrees in electrical engineering from Nagoya University, Nagoya, Japan, in 1970, 1972, and 1980, respectively.

From 1972 to 1986 he worked at the Electrical Communication Laboratories, Nippon Telegraph and Telephone Corporation, Tokyo, Japan, where he was engaged in speech recognition research. During 1984–1986, he was a Visiting Scientist in the Computer Science Department at Carnegie-Mellon University, Pittsburgh, PA, and was working on speech recognition. Since 1986 he has been with the ATR Interpreting Telephony Research Laboratories, Osaka, Japan, where he is currently Head of the Speech Processing Department.

Kevin J. Lang received the B.A. degree in mathematics from Grinnell College.

He is presently a Ph.D. candidate at Carnegie-Mellon University. His research interests include programming language design and connectionist architectures for perception.