

Neural Network Classifiers for Speech Recognition

Neural nets offer an approach to computation that mimics biological nervous systems. Algorithms based on neural nets have been proposed to address speech recognition tasks which humans perform with little apparent effort. In this paper, neural net classifiers are described and compared with conventional classification algorithms. Perceptron classifiers trained with a new algorithm, called back propagation, were tested and found to perform roughly as well as conventional classifiers on digit and vowel classification tasks. A new net architecture, called a Viterbi net, which recognizes time-varying input patterns, provided an accuracy of better than 99% on a large speech data base. Perceptrons and another neural net, the feature map, were implemented in a very large-scale integration (VLSI) device.

Neural nets are highly interconnected networks of relatively simple processing elements, or nodes, that operate in parallel. They are designed to mimic the function of neurobiological networks. Recent work on neural networks raises the possibility of new approaches to the speech recognition problem. Neural nets offer two potential advantages over existing approaches. First, their use of many processors operating in parallel may provide the computational power required for continuous-speech recognition. Second, new neural net algorithms, which could self-organize and build an internal speech model that maximizes performance, would perform even better than existing algorithms. These new algorithms could mimic the type of learning used by a child who is mastering new words and phrases.

The best existing non-neural speech recognizers perform well only in highly constrained tasks. For example, to achieve 99% accurate recognition on 105 words, a typical word recognizer must restrict the input speech to talker-dependent isolated words. That is, the recognizer must be trained with the speaker's voice prior to the speech recognition task and individual words must be spoken with pauses interjected between words. Accuracy can be as high as 95% for 20,000 words — when sentence context can be used as an aid in word recognition and when a pause is interjected between words. Even at 95% accuracy for individual

words, sentence accuracy only amounts to 50%.

Performance for recognizers that don't restrict speech to isolated-word talker-dependent speech is much worse. The best current algorithms for speech recognition use hidden Markov models (HMM) and therefore require computation rates of >100 million instructions per second (MIPS) for large vocabulary tasks. But computation rates of this magnitude are not available on conventional single-processor computers.

Building internal speech models and self-organization requires the development of algorithms that use highly parallel neuron-like architectures. Our approach to this problem is to design neural net architectures to implement existing algorithms and, then, to extend the algorithms and add advanced learning and model-building capabilities.

COMPONENTS OF AN ISOLATED-WORD SPEECH RECOGNIZER

An isolated-word speech recognizer must perform four major tasks. The structure of the speech recognizer shown in Fig. 1 reflects these tasks. The recognizer first includes a preprocessing section that extracts important information from the speech waveform. Typically, the preprocessor breaks the input waveform into 10-ms frames and outputs spectral pat-

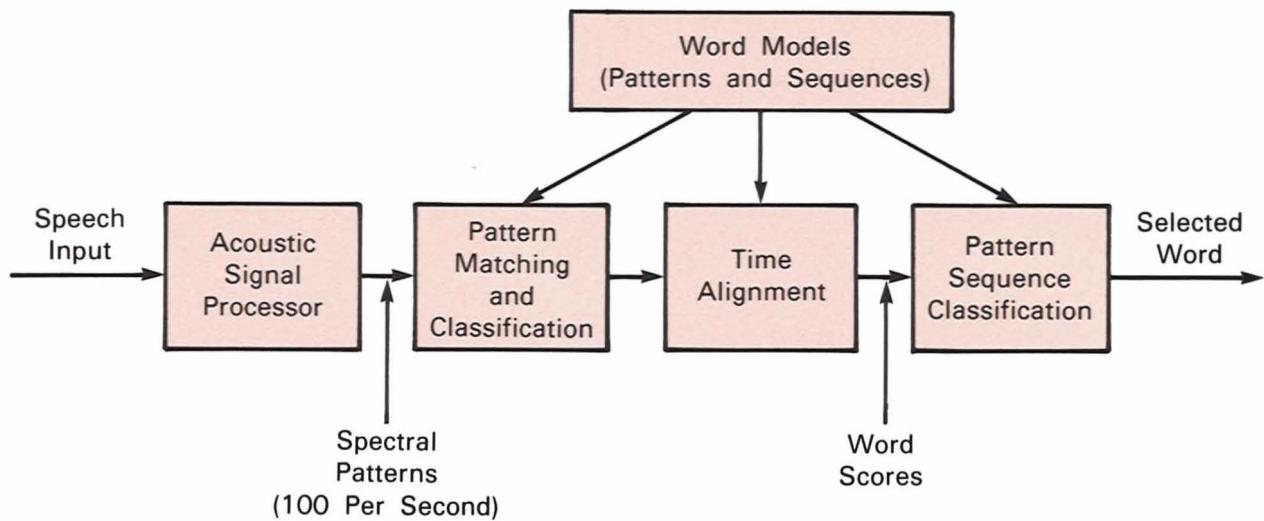


Fig. 1 — An isolated-word speech recognition system consists of preprocessor, pattern matching section, time alignment section, and pattern sequence classifier.

terns that represent the input waveform. The spectral patterns are then compared with stored reference spectral patterns. This comparison yields the local distance between the static pattern from the preprocessor and the stored patterns, which is a measure of the match or similarity between these patterns.

The sequence of spectral patterns from the preprocessor must be time aligned to the nodes in word models. Time alignment compensates for variations in talking rate and pronunciation. Word models contain the sequence of patterns expected for each word and use internal nodes to time align input pattern sequences to expected pattern sequences. Following the temporal alignment and matching functions, a decision logic section selects the word model with the best matching score as the recognizer output.

Neural nets have been applied to the four tasks required of a speech recognizer. Two of these applications — pattern matching or classification, and time alignment — are the focus of this paper.

NEURAL NETS AS CLASSIFIERS

A taxonomy of eight neural nets that can be used to classify static patterns is presented in

Fig. 2. These classifiers are grouped on the basis of whether they accept continuous or binary inputs, and on the basis of whether they employ supervised or unsupervised training.

Nets trained with supervision, such as perceptrons [1], are used as classifiers (see box, "Supervised Training"). These nets require labeled training data. During training, the number of classes desired and the class identification of each training sample are known. This information is used to determine the desired net output and to compute an error signal. The error signal shows the discrepancy between the actual and the desired outputs; it is used to determine how much the weights should change to improve the performance for subsequent inputs.

Nets trained without supervision [see box, "Unsupervised Training (Self-Organization)"], such as Kohonen's feature map-forming nets [2], can be used as vector quantizers. No information concerning correct classes is given to these nets. They self-organize training samples into classes or clusters without intervention from a "teacher." Self-organized, or unsupervised, training offers the advantage of being able to use any large body of available unlabeled data for training.

Classical algorithms that most closely resemble equivalent neural net algorithms are listed

Supervised Training

Supervised training requires labeled training data and a "teacher" that presents examples and provides the desired output. In typical operation, examples are presented and classified by the network; then error information is fed back and weights are adapted, until weights converge to final values.

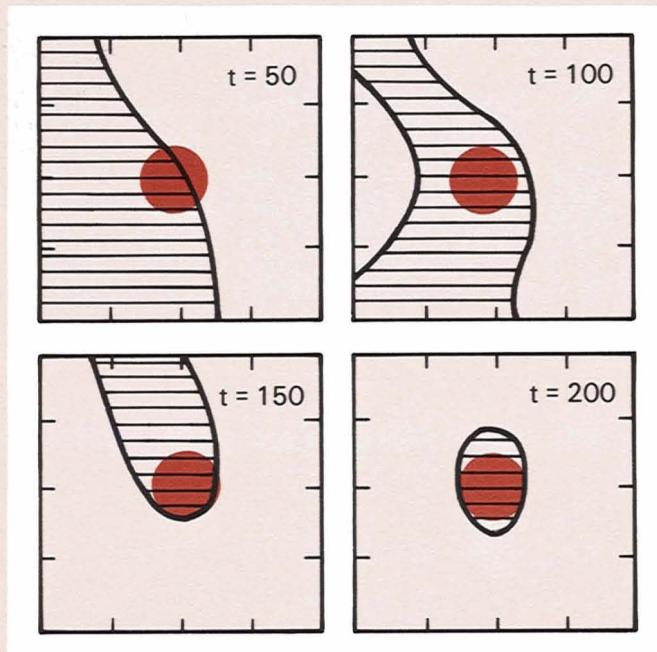
Multilayer perceptrons use supervised training and overcome many of the limitations of single-layer perceptrons. But they were generally not used in the past because effective training algorithms were not available. The development of a new training algorithm, called back propagation, has permitted multilayer perceptrons to come into increasing use [3]. The illustration shows decision regions formed, using back propagation, after 50, 100, 150 and 200 training examples were presented to a two-layer perceptron with eight hidden nodes and two outputs [4]. The shaded areas in the plots are the actual decision regions for Class A and the colored circular regions are the desired minimum error decision regions. The circular decision region for Class A formed after 200 iterations. Further iterations form a more perfect circle.

Back propagation training begins with setting node weights and offsets to small random values. Then a training example and desired outputs are presented to

the net. The actual outputs of the net are calculated, error signals are formed, and an iterative algorithm is used to adapt the node weights — starting at the output nodes and working back to the first hidden layer. An example and desired outputs are again presented to the net and the entire process is repeated until weights converge.

If the decision regions or desired mappings are complex,

back propagation can require many repeated cyclical presentations of the entire training data to converge. This difficulty increases training time, but it does not affect response time of the network during use. It does, however, set a practical limit to the size of nets that can be trained. For single-layer nets or for multi-layer nets with restricted connectivity, this restriction is not a severe problem.



Decision regions after 50, 100, 150 and 200 trials generated by a two-layer perceptron classifier trained with back propagation training algorithm.

at the bottom of Fig. 2. The Hopfield, Hamming, and adaptive resonance theory classifiers are binary-input nets: they are most appropriate when exact binary representations of input data, such as ASCII text or pixel values, are available. (These nets can also be used to solve minimization problems, in data compression applications, and as associative memories.) The single and multilayer perceptrons, feature

map classifiers and reduced Coulomb energy (RCE) nets are classifiers that can be used with continuous-value inputs. The perceptron and RCE classifiers require supervised training; the feature map classifier does not (see box, "Supervised Training").

Block diagrams of conventional and of neural net classifiers are presented in Fig. 3. Both types of classifiers determine which one of M

classes best represents an unknown static input pattern containing N input elements.

The conventional classifier contains two stages. The first stage computes matching scores, which indicate how closely an input pattern matches the exemplar pattern of each class. (The exemplar is the pattern that is most representative of a class.) The second stage of the classifier selects the class with the best matching score and thus finds the class that is closest to the input pattern.

In contrast with the preceding method of classification, the neural network classifier (Fig. 3b) accepts the N input values through N parallel input connections. The first stage of the classifier computes matching scores, in parallel. It then transmits the scores, again in parallel, to the next stage over M analog lines. Here the maximum of these values is selected and enhanced. At the end of the classification process, only one of the M -class outputs is high. Information about classification errors can be fed back to the first stage of the classifier, and connection weights between nodes adapted, thereby adjusting the algorithm and

making a correct identification in the future more likely. Adaptation is typically incremental and requires minimal computation and storage.

The classifiers displayed in the taxonomy in Fig. 2 can identify which internal class best represents an input pattern. This capability can be used, for instance, to identify speech sounds that can vary substantially between utterances. Similarly, the classifiers can identify sounds that have been corrupted by noise or by interference from other, competing, talkers.

Neural classifiers can also be used as associative, or content-addressable, memories. An associative memory is one that, when presented with a fragment of a desired pattern, returns the entire pattern. Classifiers can also serve in speech and image recognition applications to vector-quantize analog inputs. Vector quantization compresses data without losing important information, and can be used to reduce storage and computation requirements.

Neural nets differ from conventional classifiers in four important ways. The hardware has a

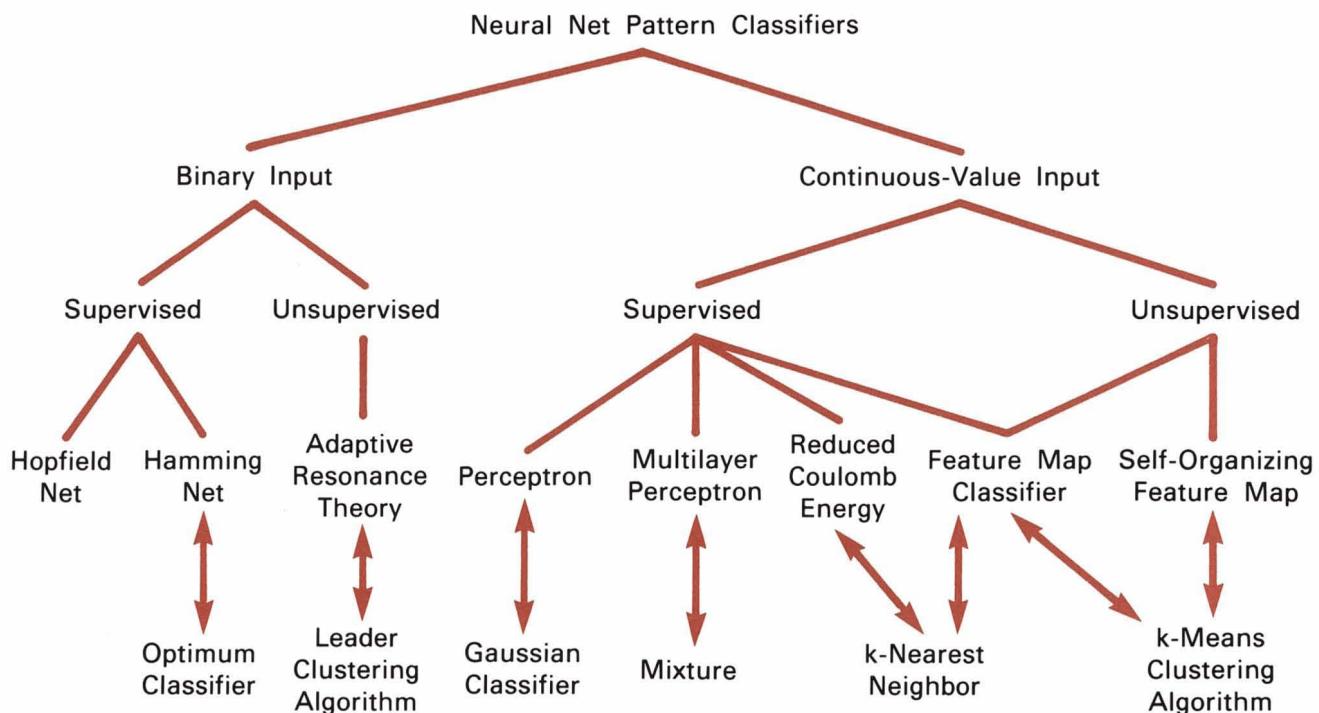


Fig. 2 — A taxonomy of eight neural net classifiers; the conventional pattern classification algorithms most similar to corresponding neural net algorithms are listed at the bottom.

Unsupervised Training (Self-Organization)

Unsupervised training, sometimes called self-organization, uses unlabeled training data; it requires no external teacher to feed error signals back to the network during training. Data are entered into the network, which then forms internal categories or clusters. The clusters compress the amount of input data that must be processed without losing important information. Unsupervised clustering can be used in such applications as speech and image recognition to perform data compression and to reduce the amount of computation required for classification, as well as the space required for storage.

Self-organized training is well represented by self-organizing feature maps. Kohonen's self-organizing feature maps [2] use an algorithm that creates a vector quantizer by adjusting weights from common input nodes to M output nodes, arranged in a two-dimensional grid (Fig. A). Output nodes are extensively interconnected with many local connections. During training, continuous-value input vectors are

presented sequentially in time, without specifying the desired output. After input vectors have been presented to the net, weights will specify cluster or vector centers that sample the input space so that the point density function of the vector centers tends to approximate the probability density function of the input vectors. This algorithm requires a neighborhood to be defined around each node that decreases in size with time.

Net training begins with setting random values for the weights from the N inputs to the M output nodes. Then a new input is presented to the net and the distance between the input and all nodes is computed. The output node with the least distance is then selected and the weights to that node and all nodes in its neighborhood are updated, which makes these nodes

more responsive to the current input. This process is repeated for further inputs until the weights converge and are fixed.

The self-organized training process is illustrated in Fig. B. Crosspoints of the lines in Fig. B represent values of weights between two inputs and each of 100 nodes in a feature map net. Lines connect points for physically nearest neighbors in the grid. The crosspoints grow with time and cover the rectangular region representing the distribution from which the input training examples were drawn. Lines don't cross, indicating that nearby nodes are sensitive to similar inputs and that a topological mapping between inputs and nodes has been achieved.

A VLSI chip that implements the feature map algorithm has been fabricated (see box, "Implementing Neural Nets in VLSI").

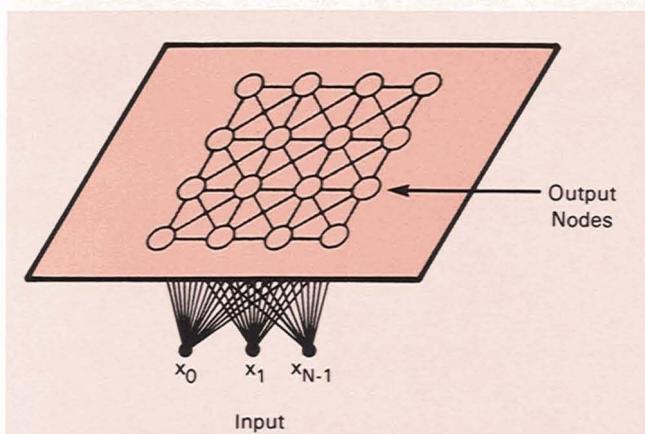


Fig. A — Using variable connection weights, this feature map connects every input node to every output node.

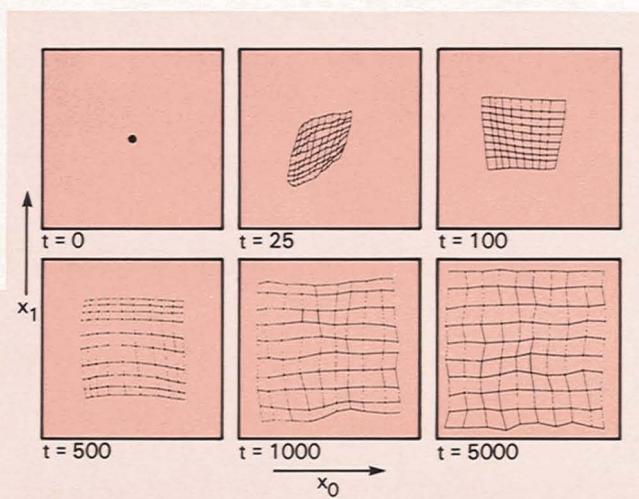


Fig. B — The vector weights to 100 output nodes from two input nodes generated by Kohonen's feature map algorithm [2]. The horizontal axis represents the value of the weight from input x_0 and the vertical axis represents the value from input x_1 . Line intersections specify two weights for each node. Lines connect weights for nearest-neighbor nodes. An orderly grid indicates that topologically close nodes are responsive to similar inputs. Inputs were random, independent, and uniformly distributed over the area shown.

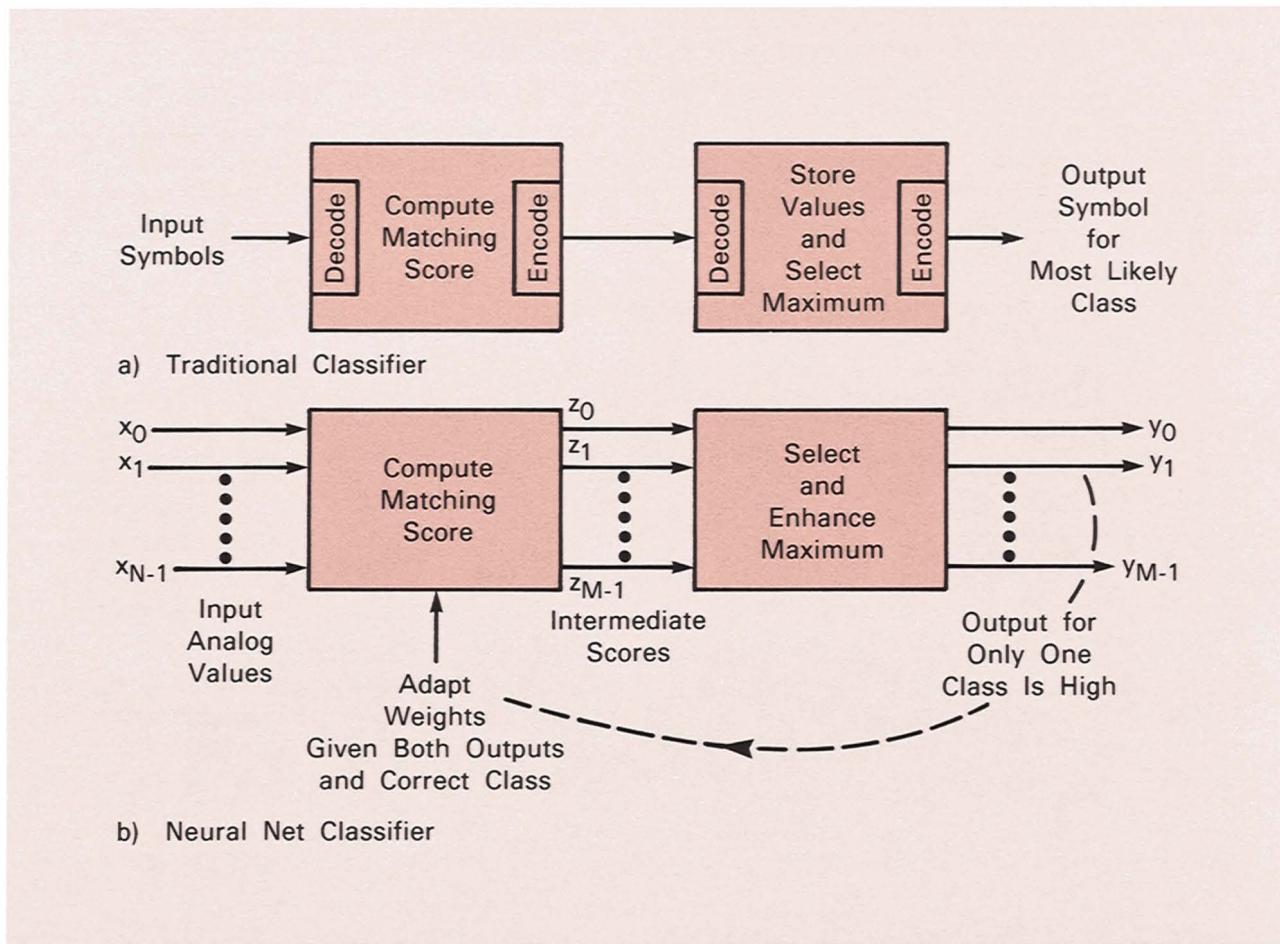


Fig. 3 — a) The inputs and outputs of conventional classifiers are passed serially and processing is serial. b) Inputs and outputs of neural net classifier are in parallel and processing is in parallel.

parallel architecture; inputs and outputs are analog and in parallel; computations are performed in parallel; and connection weights are adapted incrementally over time, which improves performance yet uses simple calculations and limited storage.

CLASSIFICATION WITH MULTILAYER PERCEPTRONS

A single-layer perceptron (Fig. 4) is the simplest possible feed-forward neural net classifier. It classifies an input pattern applied at the input into one of two classes, denoted A and B in Fig. 4. This net forms half-plane decision regions by dividing the space that the input spans, using a hyperplane decision boundary. Decision regions are used to determine which class label should be attached to an unknown

input pattern. If the values of the analog inputs for a pattern fall within a certain decision region, then the input is considered to be a member of the class represented by that decision region.

An example of the evolution of a single-layer perceptron's decision boundaries during 80 trials of training is shown on the right side of Fig. 4. This example uses a training algorithm, called the perceptron convergence procedure, which modifies weights only on trials where an error occurs.

Multilayer perceptrons [3] are nets that include layers of hidden nodes not directly connected to both inputs and outputs. The capabilities of perceptrons with one, two and three layers and step nonlinearities are summarized in Fig. 5. Single-layer perceptrons cannot sep-

rate inputs from classes A and B in the exclusive-or problem shown in the third column of Fig. 5. This problem includes two classes with disjoint decision regions that cannot be separated by a single straight line. Two-layer perceptrons typically form open- or closed-convex decision regions and thus can separate inputs for the exclusive-or problem, but they typically cannot separate inputs when classes are meshed.

Three-layer perceptrons with enough nodes in the first layer of hidden nodes can not only separate inputs when classes are meshed, they can form decision regions as complex as required by any classification algorithm [4]. Three-layer perceptrons with sigmoidal nonlinearities can also form arbitrary continuous nonlinear mappings between continuous-valued analog inputs and outputs [4].

SPOKEN DIGIT CLASSIFICATION

A new training algorithm, called back propagation, was recently developed for multilayer perceptrons [3]. Multilayer perceptron classifiers trained with back propagation were

compared with conventional classifiers using a digit-classification task involving speech from a variety of speakers. This comparison was designed to gain experience with the performance of multilayer perceptrons trained with back propagation on a difficult classification problem — not to develop a digit recognizer.

Classifiers were evaluated using the first seven single-syllable digits ("one," "two," "three," "four," "five," "six" and "eight") of the Texas Instruments (TI) Isolated Word Data Base [5]. A version of this data base that had been sampled at 12 kHz was processed to extract 11 mel cepstra from every 10-ms frame. Mel cepstral parameters are derived from a spectral analysis of a short segment of the input speech waveform; they have been found to be useful for speech recognition [6]. Each classifier was presented with 22 input coefficients for every test or training utterance word token. Eleven of these coefficients were obtained from the highest-energy frame for the word and 11 from the frame that preceded the highest-energy frame by 30 ms (Fig. 6). The additional coefficients from the earlier frame provided information about spec-

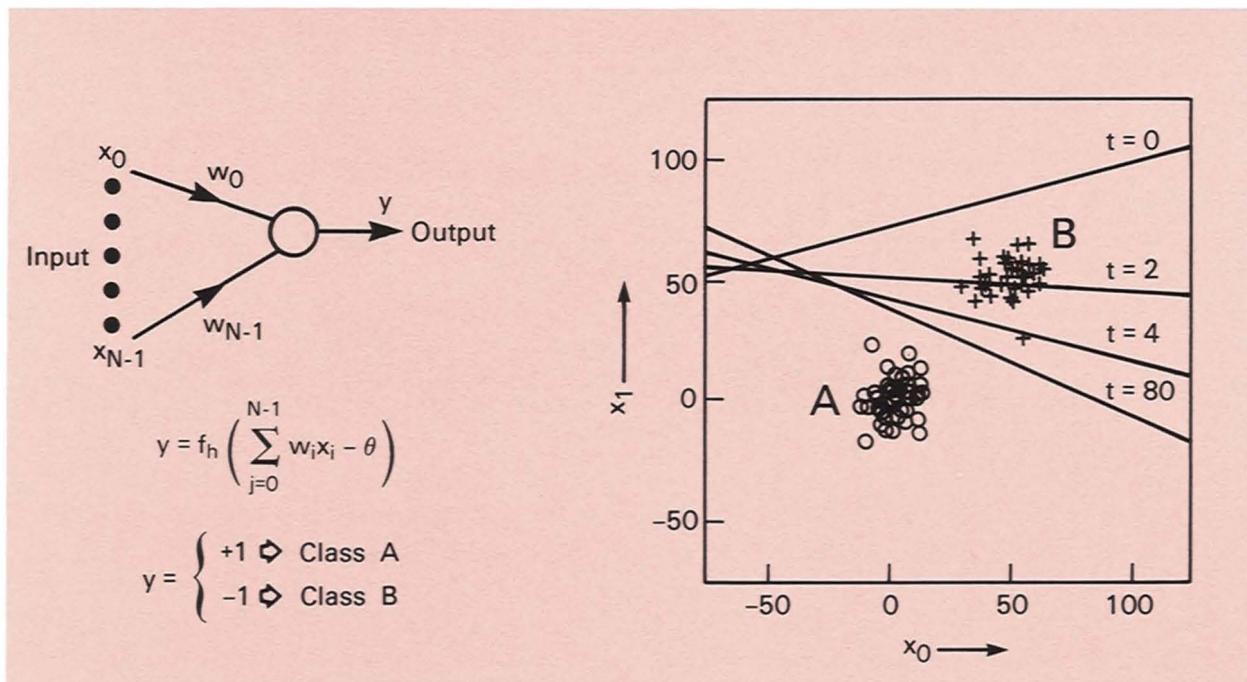


Fig. 4 — A single-layer perceptron and the decision regions it forms for two classes, A and B, using as many as 80 trials.

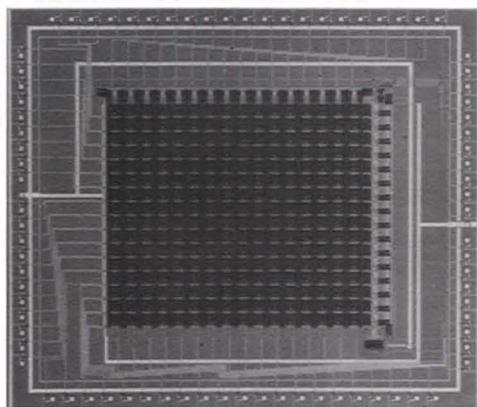
Implementing Neural Nets in VLSI

One of the most important applications of neural network architectures is in the realization of compact real-time hardware for speech, vision, and robotics applications. The chips shown here are initial implementations of two neural network architectures in VLSI. Both chips used a standard MOSIS (MOS implementation system) process that combines digital and analog circuitry on one chip. Both the perceptron chip, shown on the left [8], and

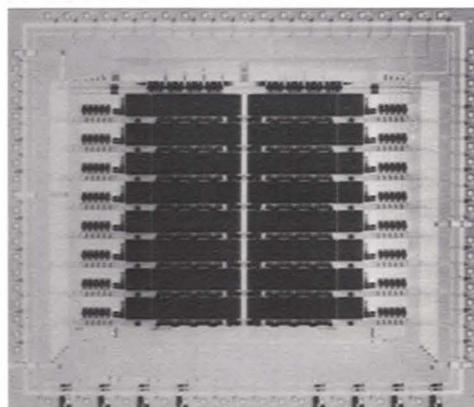
the feature map chip, shown on the right [18], store weights (connection coefficients), in digital form. The feature map chip adapts the weights internally; in contrast, the weights are adapted externally and downloaded to the perceptron chip. Analog circuits on each chip perform the multiplications and additions required by the respective algorithms.

The perceptron chip was tested in a speech recognition applica-

tion and achieved a level of performance comparable to the performance of a digital computer simulation [8]. The multilayer perceptron has 32 inputs, 16 nodes, and 512 weights. The feature map chip has 7 inputs, 16 nodes, and 112 weights. As indicated by the relative simplicity of the designs, these are first-generation chips; VLSI implementations of much greater density, with corresponding increases in capability, are being explored.



Perceptron
Classifier



Feature Map
Vector Quantizer

trial changes over time and improved the classifiers' performance.

Gaussian and k-nearest neighbor classifiers [7], which are often used for speech recognition, were compared with multilayer perceptron classifiers with different numbers of nodes and layers. All classifiers were trained and tested separately for each of the 16 talkers in the TI data base. Classifiers were trained using 10 training examples per digit (70 total) and then tested on a separate 16 tokens per digit (112 total). The Gaussian classifier used a diagonal covariance matrix with pooled variance estimates. This technique produced better performance than a similar classifier with variances estimated separately for each digit.

The Gaussian classifier can be implemented with a single-layer perceptron [4]. In fact, when

it was implemented in VLSI hardware and tested, it was found to provide the same performance on this digit classification problem as a digital computer-based Gaussian classifier [8] (see box, "Implementing Neural Nets in VLSI"). A k-nearest neighbor classifier was also used in this comparison because it performs well with non-unimodal distributions and becomes optimal as training data are increased [7]. The number of nearest neighbors (k) for the classifier was set to "one" because it provided the best performance.

All multilayer perceptron classifiers had 22 inputs, seven outputs, and used logistic sigmoidal nonlinearities along with the back propagation training algorithm (see box, "Supervised Training"). The class selected during testing was the one corresponding to the out-

put node with the largest output value. During training, the desired output was greater than 0.9 for the output node corresponding to the correct class and less than 0.1 for the other output nodes. A single-layer perceptron was compared with five two-layer perceptrons that possessed 16 to 256 hidden nodes, and to four three-layer perceptrons that possessed 32 to 256 nodes in the first hidden layer and 16 in the second hidden layer.

Results averaged over all talkers are shown in Figs. 7 and 8. Figure 7 shows the percentage error for all classifiers. The k-nearest neighbor classifier provided the lowest error rate (6.0%) and the single-layer perceptron the highest (14.4%). The Gaussian classifier was intermediate (8.7%) and slightly worse than the best two-layer (7.6%) and three-layer (7.7%) perceptrons.

The number of training examples that the perceptrons required for convergence is of special significance. These results are illustrated in Fig. 8. Perceptron networks were trained by repeatedly presenting them with the 70 training examples from each talker. The convergence time for the multilayer perceptrons was defined as the time required for the error rate to reach zero on the training data. For the single-layer perceptrons, this occurred infrequently, so the convergence time was defined as the time required for the error rate to fall and remain below 5%. The average number of examples presented for the single-layer perceptron (29,000) excludes data from two talkers when this criterion wasn't met even after 100,000 presentations.

There is a marked difference in convergence time during training between the single-layer

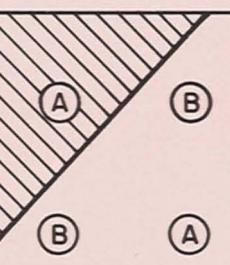
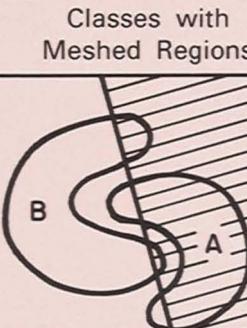
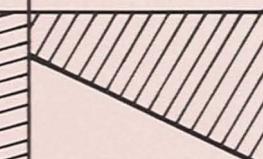
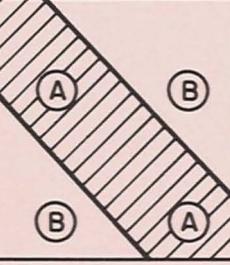
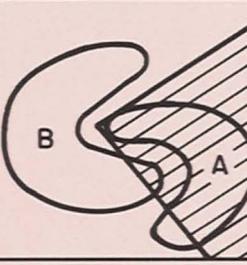
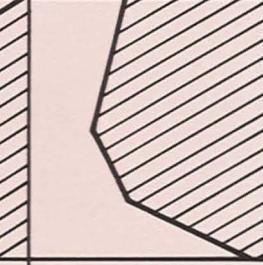
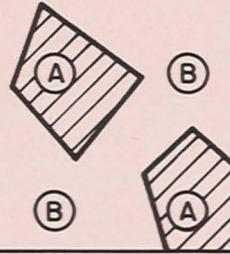
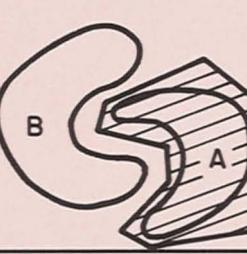
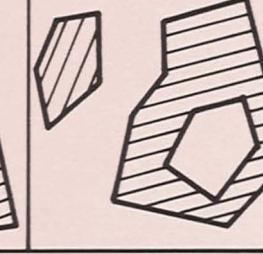
| Structure | Types of Decision Regions | Exclusive-or Problem | Classes with Meshed Regions | Region Shapes |
|--------------|---------------------------|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| One Layer | Half-Plane |  |  |  |
| Two Layers | Typically Convex |  |  |  |
| Three Layers | Arbitrary |  |  |  |

Fig. 5 — Decision regions can be formed by single-layer and multilayer perceptrons with one and two layers of hidden nodes and two inputs. Shading denotes decision regions for Class A. Smooth closed contours bound input distributions for Classes A and B. Nodes in all nets use hard limiting step nonlinearities.

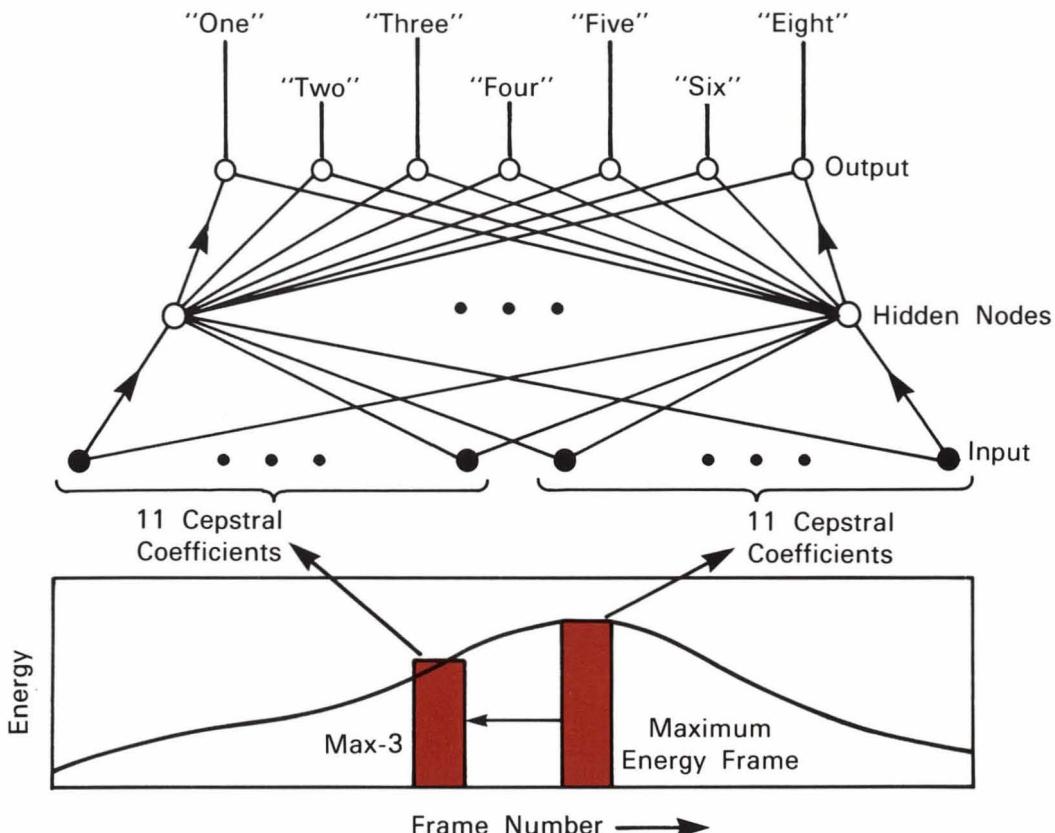


Fig. 6 — Cepstral data from the maximal-energy frame, and from the frame that was three frames before the maximal-energy frame in each word, were presented to multilayer perceptrons. Their recognition performance was compared with the performance of conventional classifiers for spoken digits.

and multilayer perceptrons. The two largest three-layer perceptrons (128 and 256 nodes in the first hidden layer) converge in the shortest time (557 and 722 examples, respectively), and the single-layer perceptron takes the longest time ($> 29,000$ examples).

These results demonstrate the potential of multilayer perceptron classifiers. The best multilayer perceptron classifiers performed better than a reference Gaussian classifier and nearly as well as a k-nearest neighbor classifier, yet they converged during training with fewer than 1,000 training examples. These results also demonstrate the necessity of matching the structure of a multilayer perceptron classifier to the problem. For example, the two-layer perceptron with the most hidden nodes took much longer to train than other two- and three-layer perceptron classifiers but the performance was no better. The poor performance of the Gaussian classifier (8.7%) relative to the k-nearest

neighbor classifier (6.0%) suggests that complex decision regions are required for this problem. These complex decision regions couldn't be provided by the single-layer perceptron. As a result, the single-layer perceptron classifier provided the worst error rate (14.4%), required thousands of presentations for convergence, and never converged for two of the talkers.

Multilayer perceptrons produce complex decision regions via intersections of half-plane regions formed in each layer (Fig. 5). The three-layer perceptrons converged fastest in the digit-classification experiment, presumably because there were more half-plane decision boundaries close to required decision region boundaries when training began. Since these boundaries were closer to the desired decision regions, they only had to be shifted slightly to produce the required regions. Other boundaries, more distant from required decision re-

gions, moved only slightly because of a derivative term in the back propagation training algorithm. These results suggest that three-layer perceptrons produce the best performance even when only simple convex decision regions are required. Although two-layer perceptrons can form such regions, convergence times with these nets may increase greatly if too many hidden nodes are provided.

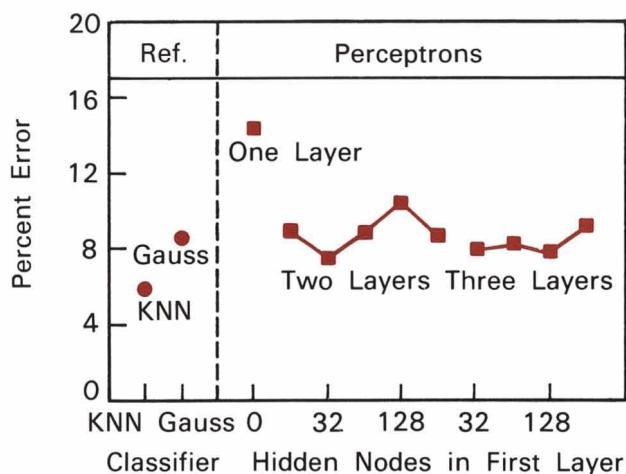
VOWEL CLASSIFICATION

Unsupervised training with unlabeled training data can often substantially reduce the amount of supervised training required [9]. Abundant unlabeled training data are frequently available for many speech and image classification problems, but little labeled data are available. The feature map classifier in Fig. 9 uses combined supervised/unsupervised training to circumvent this dearth of labeled training data [10]. This feature map classifier is similar to histogram classifiers used in discrete observation HMM speech recognizers [11]. The first layer of the classifier forms a feature map by using a self-organizing clustering algorithm [2]. The second layer is trained using back propagation or maximum likelihood training [10].

Figure 10 compares two conventional (*k*-nearest neighbor and Gaussian) and two neural net (two-layer perceptron and feature map) classifiers on vowel formant data [12]. These data were obtained by spectrographic analysis of vowels in words formed by "h," followed by a vowel, followed by a "d." The words were spoken by 67 persons, including men, women, and children. First and second formant data of 10 vowels were split into two sets, resulting in 338 training examples and 333 testing examples. These formants are the two lowest resonant frequencies of a talker's vocal tract.

Lines in Fig. 10 represent decision region boundaries formed by the two-layer perceptron. These boundaries are very near those typically drawn by trained phonologists. All classifiers had similar error rates, but there were dramatic differences in the number of supervised training examples required. The feature map classifier with only 100 nodes required less than 50 labeled examples for convergence, while

the perceptron trained with back propagation required more than 50,000 labeled examples. The first stage of the feature map classifier and the multilayer perceptron were trained by presenting them repeatedly with randomly selected examples from the available 338 training examples. Although this set of 338 examples provided labels for training, the labels were



*Fig. 7 — Percentage error in the spoken digit classification experiment for a Gaussian classifier, a *k*-nearest neighbor classifier, a single-layer perceptron, a two-layer perceptron with 16 to 256 hidden nodes, and a three-layer perceptron with 32 to 256 nodes in the first hidden layer and 16 nodes in the second hidden layer. *KNN* = *k*-nearest neighbor.*

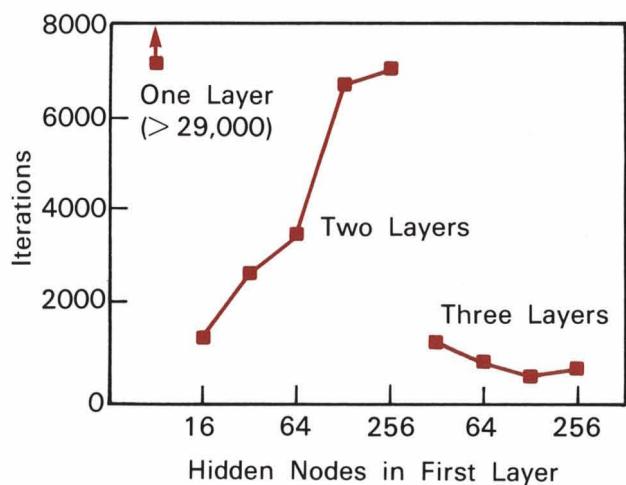


Fig. 8 — Number of iterations before convergence in the spoken digit classification experiment, for a single-layer perceptron and for multilayer perceptrons as a function of the number of nodes in the first layer.

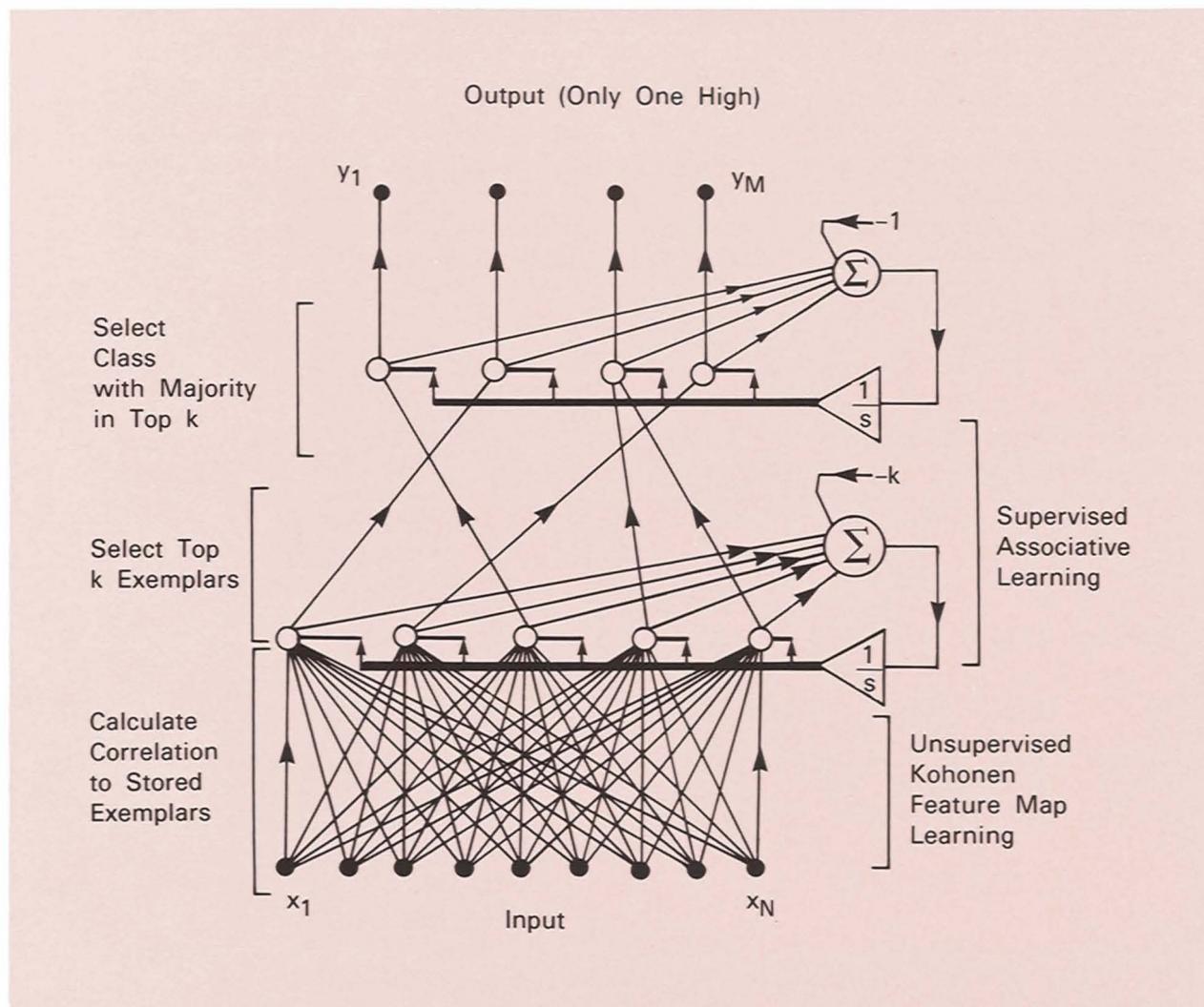


Fig. 9 — This feature map classifier combines supervised and unsupervised training, which reduces the required amount of supervised training.

stripped before training, to test the performance of the feature map classifier trained without supervision. These results demonstrate how a neural net can provide the rapid, single-trial learning characteristic of children who are learning new words. The RCE classifier [13] listed in Fig. 2 also can, with supervised training data, provide this type of rapid learning.

TEMPORAL ALIGNMENT

The digit and vowel classification tasks considered above used static input patterns. Speech, however, consists of temporal pattern

sequences that vary across utterances, even for the same talker, because of variations in talking rate and pronunciation. A temporal alignment algorithm is required to cope with this variability. Such an algorithm requires a stored word model for each vocabulary word. Nodes in a word model represent expected sequences of input patterns for that word. Time alignment typically associates each input pattern at a given time with one node in each word model. Nodes in a word model compute the distance between the unknown input pattern sequence and the expected pattern sequence for a word.

Figure 11 shows an example of the time

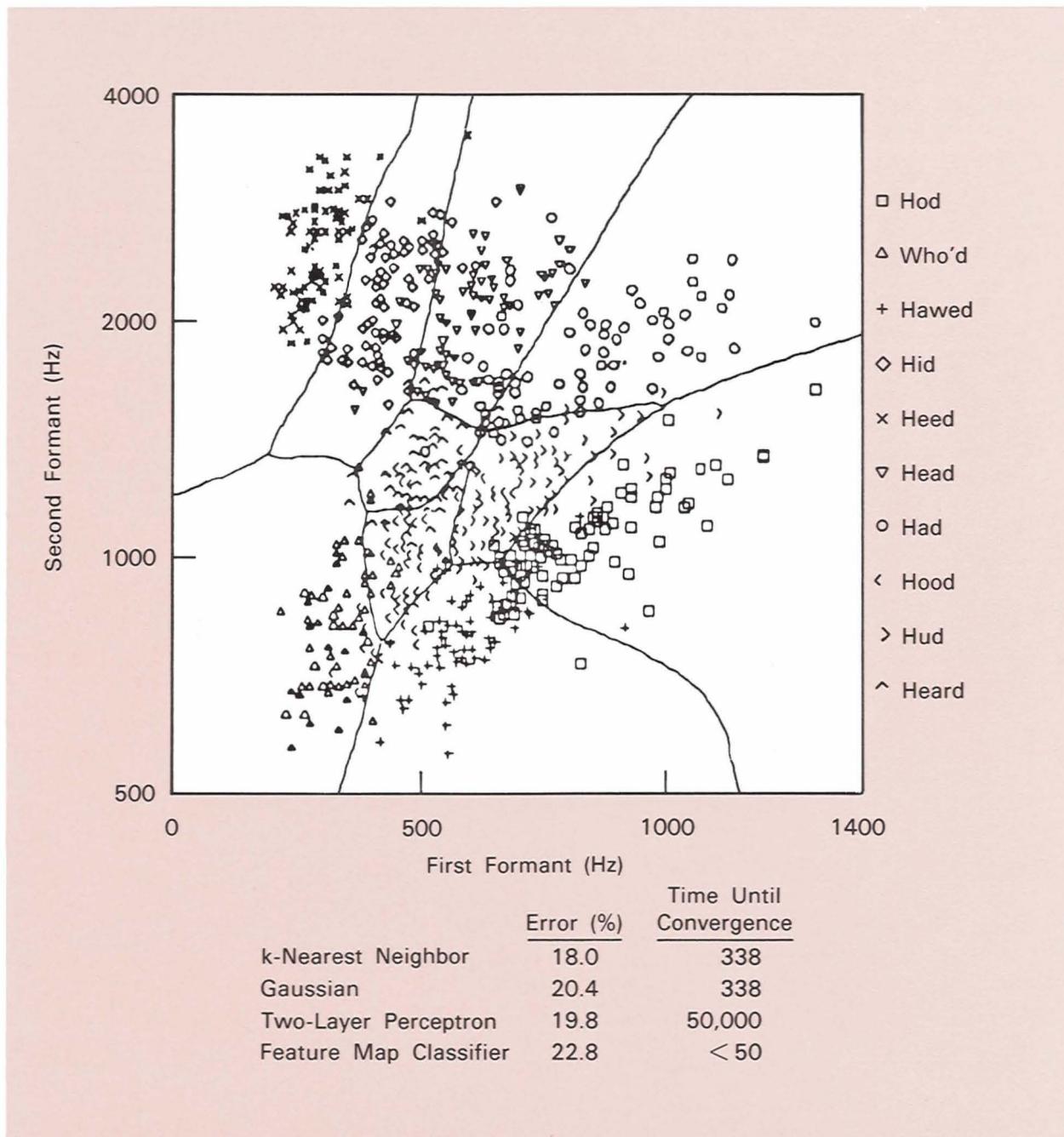


Fig. 10 — The four classifiers had similar error rates for the vowel classification task, but the required training varied dramatically. The feature map classifier needed less than 50 training examples; the two-layer perceptron required 50,000.

Viterbi Nets

A Viterbi net is a neural network architecture that uses analog parallel processing to implement the temporal alignment and matching score computation performed in conventional HMM recognizers. A block diagram of a simple Viterbi net for one word is shown in Fig. A. Nodes represented by large open triangles correspond to nodes in a left-to-right HMM word model. Connections to these nodes are adjusted to provide maximal output when there is a match between the input spectral pattern sequence and the sequence expected for the word that the net is designed to recognize. A new input pattern is applied at the bottom of the net every 10 ms; the output represents the matching score between the input sequence and the sequence expected for the word used to adjust connection weights. The output is greatest when the input sequence matches the expected sequence.

An example of the outputs of four Viterbi nets — designed to match the words "go," "no," "hello," and "thirty" for one talker in the Lincoln Stress-Speech Data Base — is presented in Fig. B. A test token for "go" was input to each of these nets. The curves plotted are the output of the next-to-last classifier node of each of

the nets, not the last node. The last node, or anchor node, matches background noise.

Time in the plot is relative to the beginning of the file containing the word "go." The file contains a background noise interval, followed by "go," followed by an additional background noise interval. The word "go" begins at roughly 160 ms and ends at approximately 580 ms.

The output from the Viterbi net designed to recognize the word "go" is, as expected, the highest. The "no" and "hello" nets provide the next highest outputs, because "hello" and "no" are acoustically similar to "go." The output of the Viterbi net designed for the word "thirty" is much lower than that of any of the others, demonstrating the effectiveness of the nets' discrimination capability.

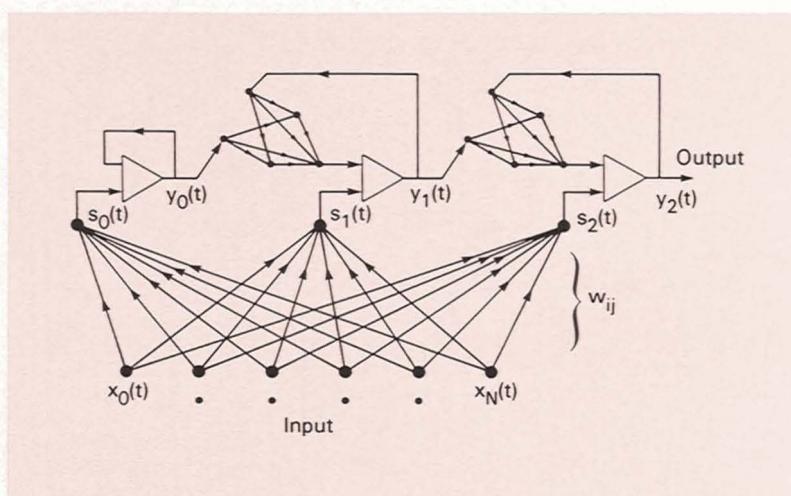
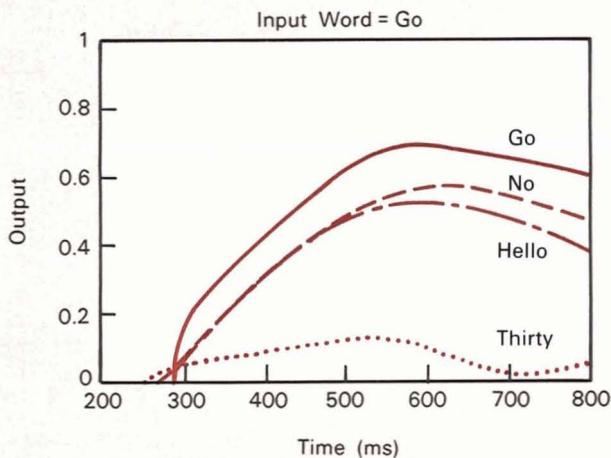


Fig. A — One Viterbi net is required for each word. The net's accuracy is greater than 99%, equivalent to the most robust HMM system.

Fig. B — Four Viterbi nets, each designed for the word labeled on the output chart, were presented with the input word "go." Those designed for words similar to "go" responded with strongly positive outputs; the net designed for the word "thirty" responded with a weak output.



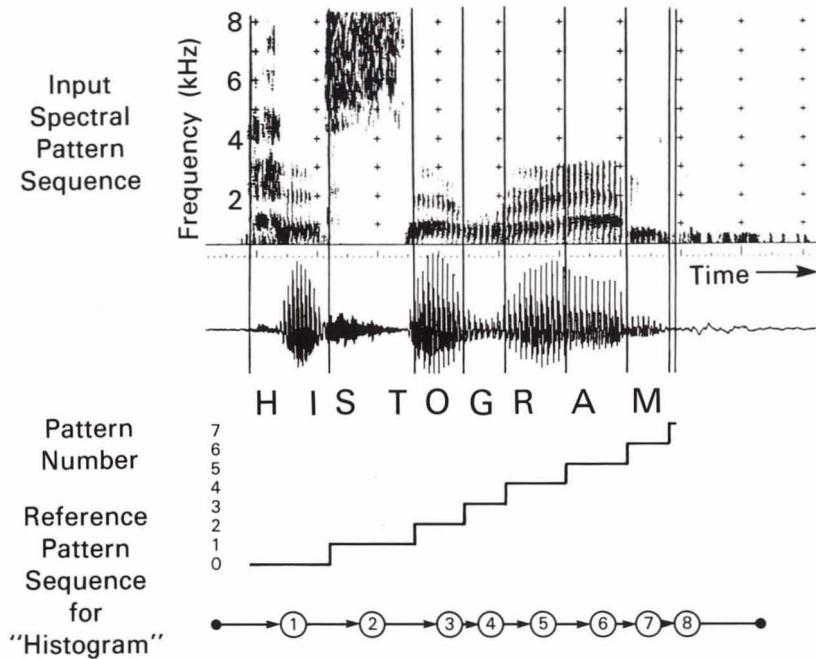


Fig. 11 — Time alignment is the process of matching an input pattern sequence to the stored expected pattern sequence of a vocabulary word. A word model is used to time-align patterns derived from frames in an unknown input with stored expected patterns.

alignment process for the word "histogram." The upper part of this figure shows a speech pressure waveform and a speech spectrogram. The spectrogram demonstrates the amount of spectral energy at different frequency regions plotted over time. Each region in the speech waveform has been associated or aligned with one node in the stored word model of the word "histogram," shown at the bottom of Fig. 11. Temporal alignment algorithms perform the alignment and compute an overall word-matching score for each word contained in the vocabulary. The overall score is computed by summing the local matching scores computed by individual nodes in stored word models. This score, in turn, determines the word that the recognizer identifies.

Classical approaches to time alignment include the use of a Viterbi decoder in conjunction with HMM recognizers, dynamic programming for time warping recognizers, and the matched or chirp filters that are used in radar applications. A number of neural net approaches have been suggested for use in time alignment, but few have been tested with large

speech data bases.

A new net that has been tested on such a data base is described in the box, "Viterbi Nets." A Viterbi net uses the Viterbi algorithm to time align input patterns and stored-word patterns for recognizers with continuous-value input parameters [11]. The process of time alignment is illustrated in Fig. 12. The solid line in this figure indicates how input patterns are aligned to stored reference patterns; the large dots represent the possible points of alignment. The Viterbi net differs from other neural net approaches because it implements a slightly modified version of a proven algorithm that provides good recognition performance. In addition, the Viterbi net's weights can be computed by using the forward-backward training algorithm [11]. It has a massively parallel architecture that could be used to implement many current HMM word recognizers in hardware.

The recursion performed in the Viterbi net is similar to the recursion required by the Viterbi algorithm when the node outputs before the first input $[y_i(0), 0 \leq i \leq M - 1]$ are allowed to decay to zero. The recursion is different though,

because node outputs can never become negative and because initial conditions do not force the first input frame to be aligned with the first classifier node. But a study performed with the Lincoln Stress-Speech Data Base of 35 words with difficult, acoustically similar, subsets showed that these differences caused no degradation in performance [14,15]. Weights in Viterbi nets with 15 classifier nodes were adjusted according to means, variances and transition probabilities obtained from the forward-backward algorithm with five training examples per word for each of the 35 words in the data base. The first and last nodes in these nets served as anchors and were designed to match background noise input. Talker-dependent experiments were performed for each of the nine talkers in the data base, using 13 normally spoken tokens per word for a total of 4,095 test tokens over all talkers.

An HMM recognizer [15] was compared with a modified recognizer that implemented the above recursion. Inputs to both recognizers consisted of 12 mel cepstra and 13 differential

mel cepstra that were updated every 10 ms. The HMM word models were trained using multi-style training and grand variance estimates [15]. Performance was almost identical with both algorithms. The error rate with a normal Viterbi decoder was 0.54% (22 per 4,095 tokens) and the error rate with the Viterbi net algorithm was 0.56% (23 per 4,095 tokens).

A block diagram of the Viterbi net is shown in the box, "Viterbi Nets," Fig. A. Classifier nodes in this net, represented by large open triangles, correspond to nodes in a left-to-right HMM word model. Each classifier node contains a threshold logic node followed by a fixed delay. Threshold logic nodes set the output to zero if the sum of the inputs is less than zero. Otherwise, they output the positive sum. Nodes positioned above classifier nodes are threshold logic nodes, and nodes below the classifier nodes simply output the sum of all inputs. A temporal sequence of T input vectors is presented at the bottom of the net and the desired output is $y_{M-1}(T)$, which is the output of the last classifier node sampled at the last time step T . Here M is

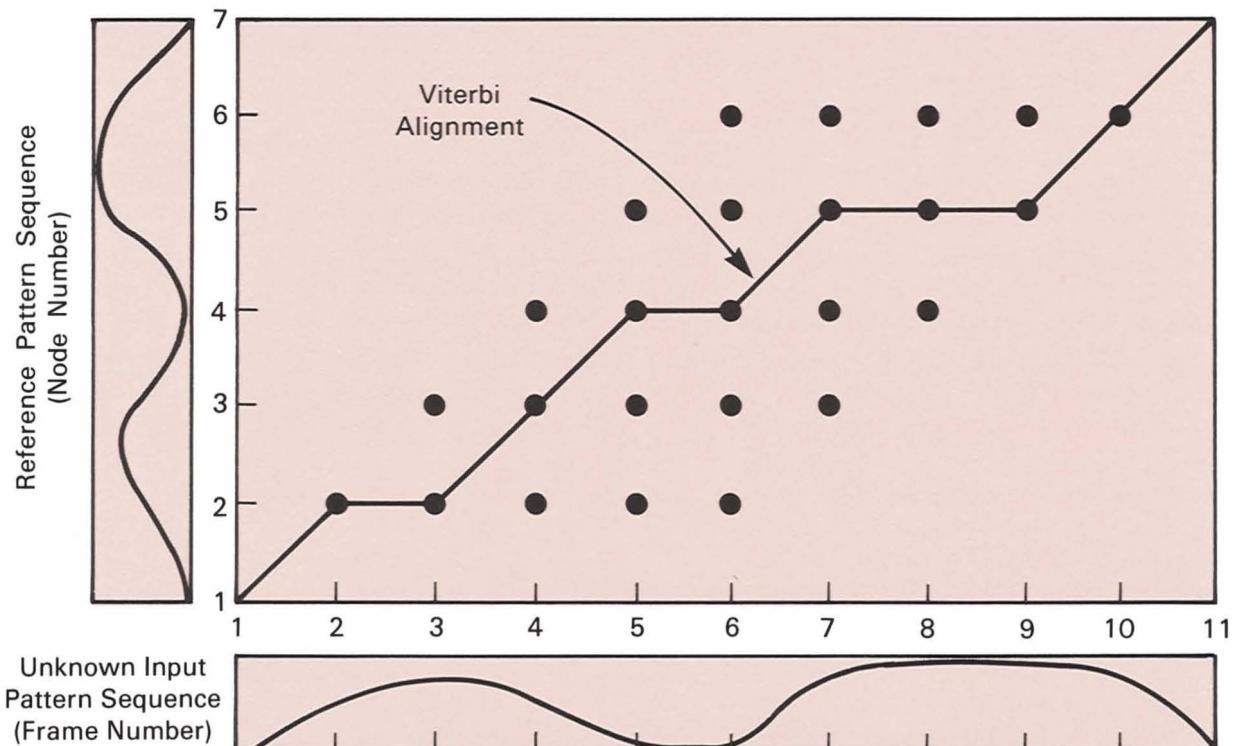


Fig. 12 — Viterbi alignment between the reference pattern sequence for a word model and the unknown input pattern sequence. Frames from the input pattern sequence are time aligned to a stored pattern sequence for a word model and, in the process, the distance from the input sequence to the word model is computed.

the number of classifier nodes in the Viterbi net. The output is monotonically related to the probability P , which is normally calculated with a Viterbi decoder. P is the probability of producing the applied sequence by traversing the HMM word model corresponding to the net over the best path that maximizes this probability. Although the Viterbi net produces a score that is related to this probability, it does not store the best path.

Delays used in the classifier nodes must equal the time interval between changes in the input vectors. Inputs at time step t , denoted $x_j(t)$ ($0 \leq j \leq N-1$, $1 \leq t \leq T$), are supplemented by a fixed input $x_N(t)$ equal to 1, which provides offsets to the summing nodes. The values denoted $s_i(t)$ ($0 \leq i \leq M-1$) represent a matching score between the current input vector and the exemplar input expected in classifier node i . When inputs are assumed to be jointly Gaussian and independent, the exemplar classifier node i is defined by mean values m_{ij} and variances σ_j^2 . Weights w_{ij} can then be assigned to form a Gaussian classifier and compute the matching score required by the Viterbi algorithm:

$$s_i(t) = \sum_{j=0}^{N-1} \frac{2m_{ij}x_j(t)}{\sigma_j^2} - \sum_{j=0}^{N-1} \frac{m_{ij}^2}{\sigma_j^2}$$

where $0 \leq i \leq M-1$. The score is strongly positive if the current input matches the exemplar that is specified by weights on links to classifier node i ; the score is slightly positive or negative otherwise.

The Viterbi net uses the following recursion to update node outputs:

$$y_0(t+1) = f[s_0(t+1) + y_0(t) + \ln(a_{00})]$$

where $0 \leq t \leq T-1$, and

$$y_i(t+1) = f \left\{ s_i(t+1) + \max_{i-1 \leq k \leq i} [y_k(t) + \ln(a_{ki})] \right\}$$

where $0 \leq t \leq T-1$, $1 \leq i \leq M-1$. In these equations, the a_{ij} terms represent transition probabilities from node i to node j in the HMM word model corresponding to the Viterbi net. The expression $f(a)$ is a threshold logic function [$f(a) = 0$, $a \leq 0$; $f(a) = a$, $a > 0$]. The maximum picking operation required in this recursion is

performed by the five-node subnets above the two right classifier nodes (see box, "Viterbi Nets"; Fig. A). These nets output the maximum of two inputs [4]. To improve clarity, the addition of the $\ln(a_{ij})$ terms to the classifier node outputs is not shown in the Fig. This extra term requires adding fixed inputs to all fed-back classifier node outputs.

The recursion increases classifier node outputs successively from left to right. It builds up node outputs when the correct word is presented and the input patterns match exemplar patterns stored in successive classifier nodes. Outputs build up only slightly or remain near zero for acoustically dissimilar words.

The above description models only a simple Viterbi net and the simplest recursion. The recursion and the Viterbi net can be generalized to allow more arbitrary transition matrices than those used by the left-to-right model. In addition, the input data rate to the Viterbi net can be increased, which provides fine temporal resolution but no change in net structure. This feature is a potential advantage of the Viterbi architecture over other approaches.

SUMMARY

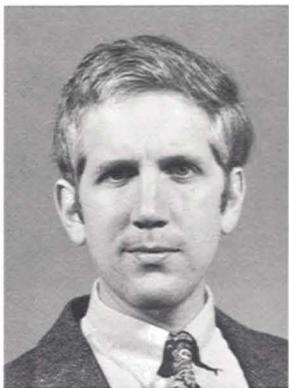
Neural nets offer massive parallelism for real-time operation and adaptation, which has the potential of helping to solve difficult speech recognition tasks. Speech recognition, however, will require different nets for different tasks. Different neural net classifiers were reviewed and it was shown that the three-layer perceptrons can form arbitrarily shaped decision regions. These neural net classifiers performed better than Gaussian classifiers for a digit classification problem. Three-layer perceptrons also performed well for a vowel classification task. A new net, called a feature map classifier, provided rapid single-trial learning in the course of completing this task.

Another new net, the Viterbi net, implemented a temporal decoding algorithm found in current word recognizers by using a parallel neural net architecture and analog processing. This net uses threshold logic nodes and fixed delays. Talker-dependent isolated-word tests using a difficult 35-word vocabulary demon-

strated good performance (greater than 99% correct) with this net, similar to that of current HMM word recognizers. Current research efforts are focusing on developing sequential adaptive training algorithms that do not rely on the forward-backward training algorithm, and on exploring new neural net temporal decoding algorithms.

For Further Information

A more extensive introduction to neural net classifiers and a more complete set of references is contained in "An Introduction to Computing with Neural Nets [4]." Various conference papers [10,14,16] describe the experiments reviewed in this paper in greater detail. Papers describing recent work in the field of neural nets are available in the *IEEE First International Conference on Neural Networks* (San Diego, June 1987), in the proceedings of the *IEEE Conference on Neural Information Processing Systems — Natural and Synthetic* (Denver, November 1987), and in the journal, *Neural Networks*, which is published by the newly formed International Neural Network Society. The Winter 1988 issue of *Daedalus* contains many articles that discuss the relationships between neural networks and artificial intelligence and another recent article provides a survey of neural net models [17].



RICHARD P. LIPPMANN received the BS degree in electrical engineering from the Polytechnic Institute of Brooklyn in 1970 and the SM and PhD degrees from MIT in 1973 and 1978, respectively. His SM thesis dealt with the psychoacoustics of intensity perception and his PhD thesis with signal processing for the hearing impaired. From 1978 to 1981 he was the Director of the Communication Engineering Laboratory at the

Boys Town Institute for Communication Disorders in Children in Omaha. Rich worked on speech recognition, speech training aids for the deaf, and signal processing for hearing aids. In 1981 he joined Lincoln Laboratory, where he has worked on speech recognition, speech input/output systems, and routing and system control of circuit-switched networks. Rich's current interests include speech recognition; neural net algorithms; statistics; and human physiology, memory, and learning.

REFERENCES

1. R. Rosenblatt, *Principles of Neurodynamics*, Spartan Books, New York, NY (1959).
2. T. Kohonen, K. Makisara, and T. Saramaki, "Phonotopic Maps: Insightful Representation of Phonologic Features for Speech Recognition," *Proc. 7th Int. Conf. on Pattern Recognition*, IEEE, New York, NY (1984).
3. D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing* (D.E. Rumelhart and J.L. McClelland, eds.), MIT Press, Cambridge, MA (1986).
4. R.P. Lippmann, "An Introduction to Computing with Neural Nets," *IEEE ASSP Mag.* **4**, 4 (1987).
5. G.R. Doddington and T.B. Shalk, "Speech Recognition: Turning Theory into Practice," *IEEE Spectrum* **18**, 26 (1981).
6. D.B. Paul, "A Speaker-Stress Resistant HMM Isolated Word Recognizer," *Int. Conf. on Acoustics Speech and Signal Processing* (1987).
7. R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, John Wiley, New York, NY (1973).
8. J. Raffel, J. Mann, R. Berger, A. Soares, and S. Gilbert, "A Generic Architecture for Wafer Scale Neuromorphic Systems," *Proc. First Annual IEEE Int. Conf. on Neural Nets*, IEEE, New York, NY (1987).
9. D.B. Cooper and J.H. Freeman, "On the Asymptotic Improvement in the Outcome of Supervised Learning Provided by Additional Nonsupervised Learning," *IEEE Trans. Comput.* **C-19**, 1055 (1970).
10. W.Y. Huang and R.P. Lippmann, "Neural Net and Traditional Classifiers," *Proc. IEEE Conf. on Neural Information Processing Systems — Natural and Synthetic*, IEEE, New York, NY (1987).
11. L.R. Rabiner and B.H. Juang, "An Introduction to Hidden Markov Models," *IEEE ASSP Mag.* **3**, 4 (1986).
12. G.E. Peterson and H.L. Barney, "Control Methods Used in a Study of Vowels," *J. Acoust. Soc. Am.* **24**, 175 (1952).
13. D.L. Reilly, C. Scofield, C. Elbaum, and L.N. Cooper, "Learning System Architectures Composed of Multiple Learning Modules," *First Int. IEEE Conf. on Neural Networks*, IEEE, New York, NY (1987).
14. R.P. Lippmann, W.Y. Huang, and B. Gold, "Neural Net Classifiers Useful For Speech Recognition," *Proc. First Annual IEEE Int. Conf. on Neural Nets*, IEEE, New York, NY (1987).
15. R.P. Lippmann, E.A. Martin and D.B. Paul, "Multi-Style Training For Robust Isolated-Word Recognition," *Int. Conf. on Acoustics Speech and Signal Processing* (1987).
16. W. Huang and R.P. Lippmann, "Comparisons Between Neural Net and Conventional Classifiers," *Proc. IEEE First Int. Conf. on Neural Networks*, IEEE, New York, NY (1987).
17. R.P. Lippmann, "A Survey of Neural Network Models," *3rd Int. Conf. on Supercomputing, International Supercomputing Institute* (1988).
18. J. Mann, R.P. Lippmann, R. Berger, and J. Raffel, "A Self-Organizing Neural Net Chip," *Neural Networks for Computing Conf.* (1988).