Neural Networks, Vol. 3, pp. 23–43, 1990
Printed in the USA. All rights reserved.

0893-6080/90 $3.00 + .00

ORIGINAL CONTRIBUTION

A Time-Delay Neural Network Architecture for Isolated Word Recognition

KEVIN J. LANG AND ALEX H. WAIBEL

Carnegie-Mellon University

GEOFFREY E. HINTON

University of Toronto

(*Received* 6 *January* 1989; *revised and accepted* 26 *June* 1989)

Abstract—*A translation-invariant back-propagation network is described that performs better than a sophisticated continuous acoustic parameter hidden Markov model on a noisy, 100-speaker confusable vocabulary isolated word recognition task. The network's replicated architecture permits it to extract precise information from unaligned training patterns selected by a naive segmentation rule.*

Keywords—Isolated word recognition, Network architecture, Constrained links, Time delays, Multiresolution learning, Multispeaker speech recognition, Neural networks.

1. INTRODUCTION

1.1. Motivation for this Study

In the last few years, statistical recognition algorithms using hidden Markov models have replaced dynamic time warping as the dominant technology in speech recognition (Bahl, Jelinek, & Mercer, 1983, Baker, 1975). The great advantage of this approach is that performance can be automatically optimized based on the information in a corpus of training data. Although a hidden Markov model is a simplistic model of speech production compared to the knowledge possessed by human experts in acoustic phonetics, it has proven to be difficult to formalize the knowledge of these experts into an automatic speech recognition algorithm, and so the simplistic but tunable hidden Markov model is more powerful in practice.

Nevertheless, most speech recognition systems based on hidden Markov models are deficient in two respects. One is that information is discarded when vector quantization is used to convert the system's real-valued acoustic input vectors into discrete tokens which can be matched against the output tokens of the hidden Markov model. Another problem is caused by the fact that the model itself contains simplifications such as the Markov assumption and the output independence assumption. While simplifications are always necessary when modeling complex natural phenomena, in this case they invalidate the justification for the commonly used maximum likelihood training procedure, which implicitly assumes that the parameters of a correct model are being estimated.

In his 1987 Carnegie Mellon thesis, Peter Brown showed that the performance of the standard IBM hidden Markov model on a particular subset of a noisy 100-speaker alphabet recognition task could be improved if the acoustic input vectors were modeled directly using continuous probability densities, and if the model were trained so that the mutual information between the acoustic input and the corresponding word sequence was maximized. This training procedure causes explicit discrimination to occur

We thank Peter Brown for much helpful assistance. This work was supported by Office of Naval Research contract N00014-86-K-0167, and by a grant from the Ontario Information Technology Research Center. Geoffrey Hinton is a fellow of the Canadian Institute for Advanced Research.

Requests for reprints should be sent to Kevin J. Lang, NEC Research Institute, 4 Independence Way, Princeton, NJ 08540.

between competing sounds regardless of the correctness of the underlying acoustic model.

Brown's research focused on the "E-set" of letters (whose names all end with the vowel "E"), on which the standard IBM system had been found to generate errors at about 4 times its usual rate. These words are difficult because the distinguishing sounds are short in duration and low in energy. His best system, which modeled continuous acoustic parameters using a special mixture of Gaussians, and which was trained using maximum mutual information estimation, committed less than half as many errors as the standard, vector quantized, maximum likelihood version of the IBM recognition system.

While this demonstration of the value of enhancements to the standard hidden Markov model was under way, a powerful connectionist learning algorithm became available: error back-propagation (Rumelhart, Hinton, & Williams, 1986).[1] This algorithm repeatedly adjusts the weights in a feed-forward network of nonlinear perceptron-like units so as to minimize a measure of the difference between the actual output vector of the network and a desired output vector given a particular input vector. The simple and efficient weight adjusting rule is derived by propagating partial derivatives of the error backwards through the network using the chain rule. It was shown that starting from random initial weights, back-propagation networks can learn to use their hidden (intermediate layer) units to efficiently represent structure that is inherent in their input data, often discovering intuitively pleasing features. Moreover, an experiment with a toy, speech-like problem showed that back-propagation networks can learn to make fine distinctions between input patterns in the presence of noise (Plaut, Nowlan, & Hinton, 1986).

Back-propagation networks learn mappings between real-valued vectors, so it would be easy to build an n-word discrimination system by training a network to map spectrograms to n-tuples representing confidence levels for the various words. When a spectrogram was presented on the input units of such a network, activation would flow up through the connections from layer to layer until each output unit was turned on by an amount that indicated its confidence that the spectrogram was an instance of its own word. During training, the target activation of the output unit corresponding to the correct word would be set to 1.0 and the target activation of the other output units would be set to 0.0. For testing purposes, the most activated output unit would determine the classification of the input pattern.

This straightforward method of using a back-propagation network to perform word recognition possessed both of the crucial features of Brown's improved hidden Markov model: the input to the system consisted of vectors of real-valued acoustic parameters, and the training algorithm explicitly caused discrimination between all pairs of output classes. Therefore, it seemed likely that a back-propagation network would be able to exceed the performance of a standard hidden Markov model on a task in which fine discrimination of highly confusable, short duration sounds is critical.

1.2. Summary of Results

The heart of this paper is a study of network architectures for performing spoken letter recognition. As in Peter Brown's pilot experiments, the four words, "bee," "dee," "ee," and "vee"[2] were used; earlier IBM research had shown that these four words were the most confusable members of the E-set of the alphabet. The data set for our architectural experiments consisted of a 144 ms salient section of each utterance, which contained the consonant-vowel transition as determined by a Viterbi alignment with the standard IBM hidden Markov model (Viterbi, 1967). The waveforms were processed by a standard DFT program and then collapsed into spectrograms containing 16 mel-scaled frequency bands and 12 ms time frames.

A 2-layer network (in which the input units are directly connected to the output units) provided a performance baseline for the design effort. This network was able to correctly classify as many as 87% of the testing tokens when it was trained for right number of iterations on the training tokens. The problem of deciding when to stop training a network was factored out of the architectural experiments by declaring that the highest level of generalization attained by a network is a good measure of the network's worth, regardless of when that generalization level occurred.

While the addition of hidden units to the baseline network improved its performance slightly, this approach to higher performance was hindered by the small size of the training corpus, which limited the number of connections which could be properly trained. A solution was found in sparsely connected network topologies that made use of small receptive fields. A further reduction in the number of weights was attained by tying together the weight patterns of successive receptive fields, resulting in a network that extracted features by repeatedly convolving a set of narrow weight patterns with the contents of a sliding window into the input. However, the network still constructed a spatialized history of the activa-

[1] Versions of back-propagation were independently derived in Parker (1985) and Werbos (1974).

[2] These words will henceforth be denoted by **B**, **D**, **E**, and **V**.

tions of these feature detectors, and so it failed to deal with input registration errors which existed in spite of the fact that the speech patterns had been selected by a Viterbi alignment with a hidden Markov model.

Because the discrimination cues in the task were short in duration, it was possible to build a temporally replicated "time-delay" network that could recognize an input pattern regardless of its alignment. Because they didn't have to account for temporal shifts of the patterns, the weight patterns learned by the new network were more sharply tuned than those of the earlier networks, and the network was able to generalize to 91% of the 144-ms word sections of the test set after being trained for the right amount of time.

While this recognition accuracy was not much higher than that of the baseline 2-layer network, there was reason to believe that the time-delay network was overqualified for the job of classifying pre-extracted salient sections of the utterances. The fact that the network had learned to locate and analyze the single most predictive moment contained in each section of speech (and conversely, to ignore the rest of the pattern) suggested that it might be possible to perform recognition on complete words after training only on salient sections, provided that the sections were long enough to be representative of the acoustical content of the complete utterances.

To test this idea, the length of the training sections was increased from 144 ms to 216 ms. To emphasize the fact that the network did not require carefully aligned training patterns, an *ad hoc* energy-based rule was used to select this new set of salient sections, rather than a Viterbi alignment. In addition, the new segmenter randomly selected a "counter-example" section from the leftover portion of each word, on which the network was trained to output a vector of zeros. Full-word recognition was performed by applying the network to an utterance in every possible position using a sliding window. The utterance's classification was determined by the maximum network output value observed during this procedure.

When trained and tested under these conditions, the time-delay network was able to correctly classify 94% of the full-length training utterances after learning to recognize 92% of those utterances based on 216 ms salient sections extracted by the *ad hoc* segmentation rule. This improvement in accuracy when moving from salient sections to the complete versions of the words indicated that a segmentation-free recognition system for this task was not only possible using a time-delay network, but desirable.

Before using this network and training methodology to build a real recognition system, it was necessary to address the question of when to halt the back-propagation learning procedure. A modified check set procedure was devised which permitted generalization to be estimated without the loss of training data. The method required the network to be trained twice, once with a divided training set so that the location of the network's generalization peak could be estimated, and then again with all of the training data, stopping after the amount of learning which had led to the best estimated generalization.

With this final piece of machinery in place, it was possible to build a recognition system whose performance could be compared to earlier results on the **BDEV** task. A 3-layer time-delay network with six hidden units was trained on the 216 ms vowel-onset and counter-example segments selected by the energy-based vowel finding heuristic, stopping at the high generalization point predicted by a preliminary check set run. At that point, the network could correctly classify 90.9% of the full-length versions of the **BDEV** test cases. This accuracy is much better than the 80% performance that the IBM recognition system achieved on these recordings, and is close to the 94% human performance measured by IBM. It is even slightly better than the 89% **BDEV** performance estimated for Peter Brown's continuous acoustic parameter, maximum mutual information hidden Markov model.[3] This is surprising because Brown's enhanced model possessed the beneficial characteristics of a back-propagation network, plus the advantage of being able to integrate evidence from distant portions of the input in a principled manner. The fact that the time-delay network performed so well without knowledge of the global structure of the utterances shows that it had acquired exceptional powers of local feature discrimination from its unaligned training examples.

2. THE TASK

The data set used in these experiments was created by the speech recognition group at the IBM T. J. Watson Research Center, and was used by Peter Brown as the domain for his thesis research on improved acoustic modeling (Brown, 1987). Using a remote pressure-zone microphone and a 12 bit A/D converter running at 20,000 Hz, digital recordings were made in an office environment of 100 speakers saying the letters of the alphabet twice, one time for training, and one time for testing. The alphabet was spoken in 3 randomized sentences, and the speakers were told to leave spaces between the words. Because of obvious speaking errors, some of the sentences had to be thrown away. When the words **B**, **D**, **E**, and **V** were extracted from the remaining sentences, there were 372 recordings available for train-

[3] It was necessary to estimate the performance of Brown's model because it had been actually been trained on the E-set task of which the **BDEV** task is a subset.

ing, and 396 for testing, ranging in length from 0.3 to 6.4 s with an average of 1.1 s. While this is a multispeaker task rather than a speaker independent one, the confusability of the words and the noisiness of the recordings make the task very difficult.[4] The recordings consist mostly of vocalic and background noise regions that are full of variability which is unrelated to the identity of the words, while the actual discrimination cues are weak and short in duration.

In an IBM study prior to Brown's work, it was found that human **BDEV** discrimination performance was 94%.[5] The standard IBM hidden Markov model could only recognize 80% of the **BDEV** tokens in this data set correctly, although the average word accuracy of the system on a 20,000 word speaker-dependent isolated word natural language dictation task was 96.5%. It is not possible to give an exact figure for the **BDEV** performance of the best version of Brown's enhanced acoustic model because his main experiments were performed on the full 9-member alphabetic E-set after exploratory experiments with **BDEV** proved successful. However, Brown did calculate an estimate of 89% for the **BDEV** performance of his best model by examining the E-set confusion matrix rows for the words **B**, **D**, **E**, and **V**, and counting only those mistakes for which the wrong answer was also in the 4 word subtask. Thus a **D** identified as a **T** would be counted as correct. This counting rule was intended to offset the disadvantage of being tuned for a larger version of the task.

2.1. Viterbi Alignment

Out of concern for the computational requirements of the then-new back-propagation procedure, our initial experiments were based on a simplified version of the task which Peter Brown had created for an expensive waveform modeling experiment contained in his thesis. Using the standard IBM hidden Markov model, a Viterbi search was performed to determine the most likely path through the stochastic model[6] corresponding to each utterance, the identity of which was known in advance. This path made it possible to assign a label to each frame of an utterance based on the identity of the phone machine which lined up with that frame. These labels were used to extract a 150 ms salient section of each utterance which included 100 ms before the first frame that was labeled "E" (this region should contain the consonant), plus 50 ms of the vowel. It was important to include the initial part of the vowel because the shape of the format tracks in this region help to identify the articulation point of the consonant. On average, .95 s of irrelevant noise and trailing vowel were removed from each utterance, while hopefully, the informative consonant-vowel transition region remained.[7] Column 1 of Figure 1 shows the waveform sections which were extracted from 4 sample words.

While easier to tackle than the full-length recordings, the Viterbi-aligned speech fragments contained enough alignment errors to motivate a shift-invariant neural network that turned out to be capable of good performance on the original, full-length recordings. The demonstration of this fact in section 4 is the main result of this paper.

2.2. Signal Processing

The IBM digital recordings were downsampled from 20,000 samples per second to the Carnegie-Mellon standard of 16,000 samples per second. Then, the CMU makedft program was used to extract the spectral characteristics of these recordings. This program employs a 320 point Hamming window which covers 20 ms and is advanced by 48 samples, or 3 ms, per frame. The last 64 data points from the 320 point window are folded into the first 64, yielding a 256 point real valued input vector which is processed by a 128 point complex DFT which treats the even numbered samples as real values and the odd numbered samples as imaginary values. The last component of the resulting 129 dimensional complex-valued vector is discarded. The remaining components are converted to decibels by the function $20 * \log_{10}(\mathrm{sqrt}(r^2 + i^2))$.

Thus, the program converted our 150 ms waveform samples into spectrograms containing 128 log energies ranging up to 8 kHz, and 49 time frames of 3 ms each.[8] The first frame of each spectrogram was then discarded so that there would be 48 time steps (a highly factorizable number), and the DC bias component of each frame was set to zero. Because each

[4] The signal-to-noise ratio of the data set was estimated to be 16.4 dB by using a hidden Markov model to label the utterances and then dividing the average signal power in the consonant and vowel regions by the average signal power in the background noise regions. This figure is much lower than the 50 dB signal-to-noise ratio of typical lip-mike speech data.

[5] Human performance dropped to 75% on **BDEV** tokens that were resynthesized after being run through the IBM signal processor.

[6] In the IBM system, the words **B**,**D**, and **V** are modeled by a concatenation of the state machines for noise, voiced consonant onset, {B,D,V}, E, E trail-off, and noise. The word **E** is modeled by a concatenation of the state machines for noise, E onset, E, E trail-off, and noise. The state machines contain 3 main states with associated transitions to model the beginning, middle, and end of each phone. The consonant and vowel machines include self-loops to model steady-state portions of the acoustic signal, and all of the machines include null transitions to model short durations.

[7] The phrase "consonant-vowel transition" is being applied uniformly to these words for convenience, even though **E** doesn't really begin with a consonant (except for an occasional glottal stop). In the case of **E**, this phase is being used to refer to the vowel onset of the word.

[8] Windowing effects accounted for the fact that there weren't 50 time frames in these spectrograms.

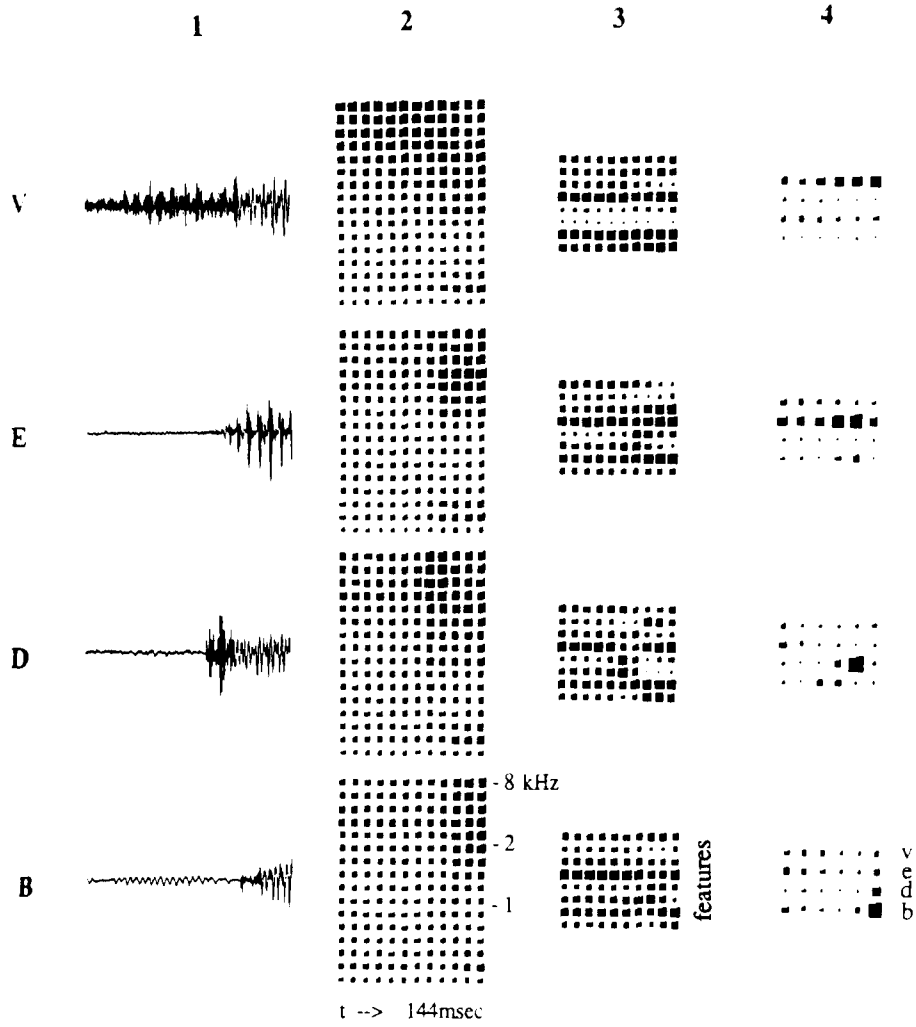

FIGURE 1. The waveforms shown in column 1 are 144 ms slices extracted from around the consonant-vowel transition in the words B, D, E, and V by a Viterbi alignment with the IBM hidden Markov model. Column 2 shows the 12 × 16 input spectrograms derived from these waveforms by the signal processing procedures described in section 2. Column 3 contains the hidden unit activation patterns triggered by these spectrograms in a 3-layer network with replicated hidden units. The 10 copies of 8 hidden units show the presence or absence of 8 features in 10 successive time positions. Column 4 contains output unit activation patterns from a 3-layer network with replicated output units. These patterns show the confidence levels of the network at successive time positions. Observe how the D and B detection events are localized in time.

of the 48 time frames represented 3 ms, the final duration of the spectrograms was 144 ms.

2.3. Spectrogram Post-processing

The input spectrograms which resulted from our signal processing contained 48 × 128 = 6144 points. A network must have at least one connection to each of its input units, and every connection contains a weight that must be trained.[9] Clearly, 372 training examples are insufficient to train a model with some multiple of 6144 parameters (see section 2.4), so it was necessary to decrease the size of the network, and hence the size of the input spectrograms. This was accomplished by combining adjacent columns and rows of the raw spectrograms to generate smaller, less detailed spectrograms to feed into the network. It was also necessary to scale the energy values to a range that back-propagation networks find palatable, namely between 0.0 and 1.0.[10]

There are many plausible sounding methods of compressing a spectrogram and normalizing its component values. The peak generalization of a simple 2-layer back-propagation network was used to empirically compare the various alternatives. A 2-layer network is particularly useful in this sort of role not only because it converges rapidly, but because its solutions are always the best possible for the given

[9] In section 3.5, we will describe a constrained learning procedure that effectively reduces the number of free parameters in a network.

[10] Actually, it might increase the speed of learning to use input values between −1 and +1 with a mean value of 0. The advantage of using an ensemble of input vectors with zero-mean components is that randomly related vectors are roughly orthogonal, which minimizes interference (Hinton & Plaut, 1987).

network with a particular environment; there is no chance of falling into a local minimum that would result in an unfair measure of the quality of the environment.[11]

Following common speech recognition practice, the frequency resolution of the network was fixed at 16 bands. Its time resolution was temporarily set to 6 frames while the other processing options were evaluated. The best level of time resolution was then determined empirically, as described in the next section.

Although the frequency resolution had been fixed at 16 bands, a choice needed to be made as to the method for condensing the 128 points contained in each time step of the raw spectrograms. One possibility was to create a linear frequency scale by collapsing adjacent bands. The alternative was to use a mel scale with variable-width, overlapping bands. The mel scale, which is linear up to about 2 kHz, and logarithmic above that, was motivated by the cochlea, which has good frequency resolution at low frequencies and good temporal resolution at high frequencies. Unlike a cochlear model, fixed window DFTs cannot make that tradeoff, but one might expect the mel-scaled spectrograms to work better than the linearly scaled ones because they provide more resolution in the more informative low frequency regions.

Two possibilities were evaluated for an energy normalization method. The casewise method was to find the lowest and highest energies in a given spectrogram and then scale them to 0.0 and 1.0, respectively. The alternative was to make a global choice for energies to map to 0.0 and 1.0; after examining the overall distribution of energies in our data set, we selected the values of -5 and 105 dB[12] and then clipped any peaks that exceeded those bounds. One would expect the global method to yield better performance if the total energy in a spectrogram was a clue to the identity of the word.

Finally, shaping functions were tested that would drive the input values towards the boundaries of 0.0 and 1.0. Three alternatives were considered: doing nothing to the values, squashing them with a sigmoid function, and squaring them and then multiplying by 1.4.

When training and testing environments were constructed using linear frequency bands, casewise input scaling, and no shaping function, the generalization of the simple 2-layer network peaked at 82%. After trying the other spectrogram post-processing options in various combinations, it was found that 86% peak generalization was possible with an input format that employed mel-scaled frequency bands, global (versus casewise) energy normalization, and input values reshaped by squaring. A linear frequency scale reduced this figure to 83%, as did casewise normalization. Not squaring the components of the compressed spectrograms reduced the network's peak performance by 1%.

2.4. Determining the Optimum Level of Temporal Resolution

The number of components in a network's input pattern affects the number of weights in the network, which in turn affects the network's information capacity and hence its ability to learn and generalize from a given number of training cases. Because there were approximately 400 training cases available for this task, each of which requires an output choice that can be specified with 2 bits, a network would need to learn 800 bits of information to perform the task by table lookup. According to CMU folklore, each weight in a back-propagation network can comfortably store approximately one and a half bits of information, so a network with more than about 500 weights would have a tendency to memorize the training cases and thus fail to generalize to the test set. Given the fact that the network's frequency resolution was fixed at 16 bands, this limit of 500 weights indicated that the network's input format shouldn't contain more than about 8 time steps. However, this generic information capacity argument doesn't take into consideration the specific properties of the domain; it is easy to imagine that compressing a spectrogram to a small number of wide time steps could destroy all traces of some important but fleeting articulatory event.

To test the validity of this estimate, we generated spectrograms with a range of different temporal resolutions (24, 12, 6, and 3 ms frames) and then used them to train appropriately sized 2-layer networks for 1000 epochs of the batch back-propagation procedure.[13] Because the higher resolution networks appeared to have an excessive number of weights given the size of the training set, weight decay was used in hopes of giving them a chance to stay in the game long enough to exploit the additional information that was available to them. For all of the networks, peak parameters of about $\{\varepsilon = .005, \alpha = .95, \delta = .001\}$ were employed, where ε is the factor by which

[11] Strictly speaking, the convergence theorems for 2-layer networks do not apply when a perfect solution doesn't exist (our training set contained conflicting evidence), but in practice, multiple runs from different starting points converge to the same solution on this task.

[12] The absolute size of these log energy values are a meaningless artifact of the IBM digital recording system and the scaling factors employed by the CMU makedft program.

[13] In batch back-propagation, weight updates are based on the sum of the gradient vectors of all of the cases in the training set. The simulator used for these experiments does not normalize the accumulated gradient by dividing by the size of the training corpus.

the gradient is multiplied before modifying the weights, α is the momentum term defined in Rumelhart, Hinton, and Williams (1986), and δ is the factor by which each weight is decayed after each iteration.[14] After every 200 iterations, each network's generalization to the test set was measured by counting the number of cases that the network classified correctly according to the "best guess" rule, which states that the network is voting for the word whose output unit is most active. Peak generalization was defined to be the largest value in the resulting sequence of generalization scores.

It turned out that the networks were indistinguishable using the best-guess metric, which was initially somewhat surprising considering the large number of weights which some of the networks possessed. The explanation for this performance parity is that 2-layer networks are poor table lookup devices because they can only memorize linearly independent patterns. Thus an oversized 2-layer network doesn't suffer as much in generalization as an oversized multilayer network would. In order to get some clue as to the relative advantages of the various input formats, we resorted to a tougher, threshold-based counting rule which only scores a case as correct when the correct output unit has an activation of more than 0.5 and the other three have activations of less than 0.5. The results of these measurements are summarized in Table 1. The spectrogram format with 12 time steps (each representing 12 ms) was the winner by a slim margin. Column 2 of Figure 1 shows four sample words in this format. Similar experiments with more sophisticated network architectures have confirmed that, on this task, the input format utilizing 12-ms frames is in fact the best at providing enough resolution while minimizing the number of connections that have to be trained.

TABLE 1

Peak generalization performance of a 2-layer network as a function of input resolution. Accuracy was computed using a strict, threshold-based counting rule that requires the correct output unit to be more active than 0.5 and the other three to be less active than 0.5.

Temporal resolution	Resulting peak generalization
24 ms	71.0%
12 ms	73.5%
6 ms	71.0%
3 ms	70.7%

[14] Rather than using fixed values for the learning parameters, we started each run with small parameter values and increased them by hand when the learning procedure located and began to follow a ravine in weight space (it is possible to track this process by looking at the cosine of the angle between successive weight steps.) Since the shape of the network's weight space was determined by the training set, multiple learning runs on the task required similar sequences of parameters, and so our initial "hand flying" of the learning parameters evolved into the following fixed parameter schedule for 2-layer networks: initially, $\{\varepsilon = .001, \alpha = .5\}$; after 50 epochs, $\{\varepsilon = .001, \alpha = .9\}$; after 100 epochs, $\{\varepsilon = .002, \alpha = .95\}$; and after 200 epochs, $\{\varepsilon = .005, \alpha = .95\}$.

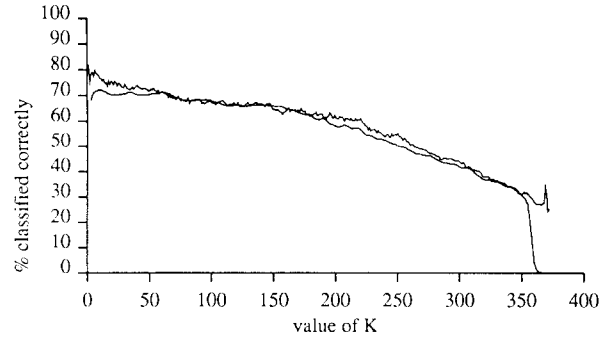

FIGURE 2. This plot shows how *k*-nearest neighbor classification performance on the 144 msec Viterbi-aligned BDEV vowel-onset spectrograms varied with *k*. The lower curve (which was artificially smoothed for clarity) was generated by a cross-validation experiment on the training set, while the higher curve shows generalization from the training set to the testing set. The cross-validation experiment indicated that $k = 9$ would work well, but in fact, the very best generalization rate of 82% occurred with $k = 1$.

2.5. K-nearest Neighbor Results

The simple but powerful k-nearest neighbor algorithm (Duda & Hart, 1973) was used to measure the difficulty of the 144-ms Viterbi-aligned version of the task. When a test input vector is presented to this algorithm, the output vectors associated with the k-nearest input vectors from the training set are used to determine the classification of the test vector. Because the performance of k-nearest neighbor is a function of k, the algorithm was tested using every possible value of k, yielding the jagged curve shown in Figure 2. The highest point on the curve was the 82% generalization spike at $k = 1$.

It can be shown that the error rate of an optimal linear bayesian classifier is no less than half that of a nearest neighbor classifier. Because nearest neighbor yielded an 18% error rate on the 144-ms Viterbi-aligned spectrogram segments, an optimal linear bayesian classifier would suffer from a 9% error rate. Coincidentally, that is the accuracy of the time-delay neural network that will be described in the next section.

3. ARCHITECTURAL EXPERIMENTS

This section contains a sequence of increasingly complicated networks that were evaluated on the Viterbi-aligned version of the **BDEV** recognition task. Each network in the sequence resulted from a modification to the previous network. The first modifications were

motivated by generic issues as information capacity and computational power, but the final and most useful modification was motivated by a priori knowledge about the task.

3.1. A Baseline Network

Column 2 of Figure 1 shows a sample spectrogram of each word in the 12 × 16 format that was selected in section 2. A 2-layer network designed for processing these patterns is shown in Figure 3. The network has four output units which represent the four words of this task. Each of the output units has 192 connections to the input layer, so the network contains 768 weights that must be tuned. After undergoing 1000 iterations of the back-propagation learning procedure (consuming about 5 minutes of CPU time on a Convex C-1), the network answered correctly on 93% of the training set tokens and on 86% of the test set tokens, which is better than k-nearest neighbor had done on the same 144-ms Viterbi-aligned spectrograms.

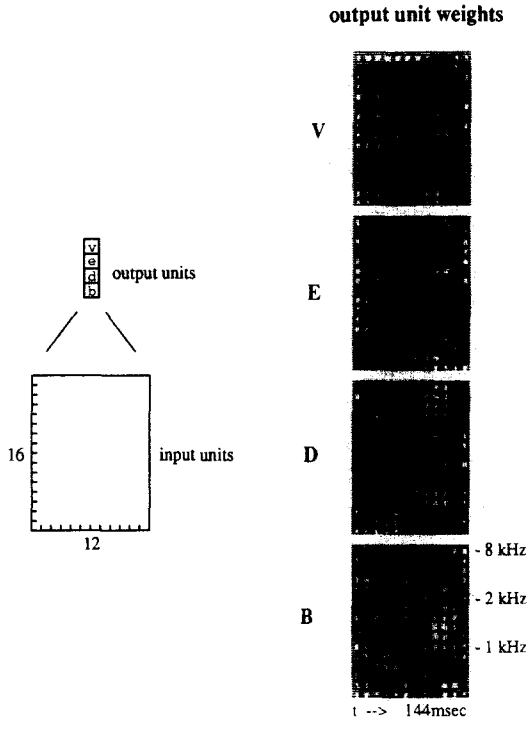

FIGURE 3. A 2-layer network with 4 output units, each of which is connected to the entire 12 × 16 array of input units. Each output unit also has an unshown link to a "true" unit to implement its bias. The 4 rectangular patterns on the right show the weights that the learning procedure chose for connecting each of the 4 output units to the input array. White and black blobs represent excitatory and inhibitory weights, and the size of a blob encodes the magnitude of the weight. These weight patterns show that the network is using the acoustic evidence to perform the task in a sensible way. For example, the row of white blobs on the upper left of the top weight pattern shows that the network considers frication to be evidence in favor of the word V.

More interesting than the network's performance is the fact that the learning procedure managed to extract sensible looking weight patterns from noisy, hard to read spectrograms. Each of the four rectangular patterns on the right side of Figure 3 shows the weights that the network developed for transmitting activation from the rectangular input array to the output unit corresponding to a particular word. Each of the black and white blobs pictured on these rectangles shows the sign and magnitude of a single weight by means of the color and size of the blob. A positive weight (represented by a white blob) is excitatory, and causes the output unit to become activated when energy is present in the component of the input pattern to which the weight is connected. A negative weight (represented by a black blob) is inhibitatory, and causes the output unit to become deactivated when energy is present in the corresponding component of the input pattern. A zero weight (represented by a blank spot in the pattern) causes the output unit to ignore the contents of the corresponding component of the input pattern. Time is represented by the horizontal axis of the weight displays, and frequency by the vertical axis.

Because all four of the patterns change character near the 9th time frame, it appears that the vowel onset typically occurs in that position. The white blobs near the top of the 8th, 9th, and 10th frames of the weight pattern of the **D** unit show that it is stimulated by the high frequency energy burst which occurs at the vowel onset of that word. The white blobs in the highest frequency band during the first half of the weight pattern of the **V** unit show that it is stimulated by frication. The black blobs in the two lowest frequency bands at the beginning of the weight pattern of the **E** unit show that it is inhibited by pre-voicing.

While each of these features only votes for or against a single word, the sloped pattern occurring in the middle frequency bands at the vowel onset in the weight patterns of the **B** and **E** units shows that the network is using the presence or absence of a rising F_2 to perform pairwise discrimination of these two words. The weights in the corresponding region of the **D** unit's pattern all have small magnitudes, indicating that this feature has little predictive power for **D** on this task.

3.2. Temporarily Allocating More Training Data

The ceiling of 500 weights hypothesized in section 2.4 seemed like a serious impediment to the construction of interesting networks, especially since the generalization penalty for exceeding the ceiling increases as the sophistication and computational power

TABLE 2

A comparison of the learning trajectories of a 2-layer network and a 3-layer network with 4 hidden units. This table shows the error rates of the networks on the training and testing sets at selected times during the training process. Two error metrics were used: the mean squared error per case and the number of erroneously classified utterances (out of 668 training cases and 100 testing cases). The 3-layer network learned more of the training cases, but took longer to do so. It achieved a lower mean squared error on the test set, but didn't actually classify any more of the test cases correctly.

2-layers no hidden units				3-layers four hidden units			
training		testing		training		testing	
epochs	errors	mse	errors	epochs	errors	mse	errors
200	90	.120	20	2000	97	.113	18
400	53	.117	16	4000	49	.110	16
800	45	.120	14	6000	16	.128	14

of a network's architecture grows. In order to raise this ceiling, all but 25 randomly selected examples of each word were transferred from the testing set to the training set during the following experiments. While this redistribution provided more freedom to investigate complicated network architectures, it also invalidated comparisons with experiments conducted on the task in its original form. Moreover, it turned out to be unnecessary in the end because our final network had a complex structure but only a small number of weights, and hence was trainable with a limited amount of data. Therefore, in section 3.11, the training and testing sets will be reverted to their original forms.

3.3. Adding a Hidden Layer

Although the 86% peak generalization achieved by the 2-layer network of Figure 3 was better than the 82% performance of *k*-nearest neighbor on the same Viterbi-aligned segments,[15] it still fell short of the 94% human **BDEV** accuracy measured by the IBM speech group.

By adding layers to a back-propagation network, one can increase the complexity of the decision functions that it can compute. Hopefully the expanded family of computable functions will then permit a more natural fit to the training data. To measure the effect of adding a layer to the network separately from the effect of changing the number of weights in the network, a 3-layer containing only 4 hidden units was built and trained.

This network was not very different from the 2-layer network discussed in section 3.1. The two networks had nearly the same number of connections (792 vs. 772), and in their second layers, both networks were forced to represent all of the relevant information using only four activation values. The similarity between the two networks is reflected by their similar learning trajectories, which are summarized in Table 2. The distinguishing characteristics of the 3-layer version are a longer training time, the ability to learn more of the training cases, and slightly better test scores according to the mean squared error metric which has more of an analog character than the best-guess metric. The additional layer helps the 3-layer network squeeze the activations of its output units closer to their target values, regardless of whether the rank order of the various activations is correct.

The second multilayer network that was tried had a better chance for improved performance. This network contained twice as many (8) hidden units, which doubled both the network's information capacity and the bandwidth of its hidden layer. The learning trajectory of this network is shown in the left half of Table 3. The network easily consumed the training set, mastering 99% of the cases in 2400 epochs. At that point, the network was able to correctly classify 89% of the test cases. An examination of the network's weights, which are pictured in Figure 4, shows that the network used four of its hidden units as templates for the four words (much like those developed by the 2-layer network). The remaining hidden units represented the disjunctions {**BD**}, {**BE**}, and {**EV**} and an alternate form of **D**.

3.4. Receptive Fields

The 3-layer networks described in the previous section are of the unsophisticated "bag-of-hidden-units"

[15] It is surprising that a simple 2-layer network can outperform *k*-nearest neighbor on this task for any value of *k*. The *k-nn* algorithm has all 372 training patterns available for reference purposes, while the 2-layer network has only four weight patterns (each as large as an input pattern) with which to represent all of the information in the training set. In (Lippmann & Gold, 1987), *k-nn* outperformed 2-, 3-, and 4-layer networks with various numbers of hidden units on a digit recognition task. Its poor performance here may be due to the fact that the discriminative information in a word from the E-set makes up only a small portion of the input pattern. The back-propagation network can learn small weights which allow it to ignore input components to which *k-nn* must give equal weight when computing euclidean distances.

TABLE 3

A comparison of the learning trajectories of two 3-layer networks, each of which contained about 1500 weights. In the first network, all of the hidden units were connected to the entire input. In the second, each hidden unit was connected to a window of 3 time steps out of 12. Both networks were able to learn 99% of the training cases, and peaked at 89% generalization. The first network began to overtrain after 2400 epochs. The second network took twice as long to learn the task, but produced lower mean squared error values on the test set.

8 fully connected hidden units				30 narrow receptive field hidden units			
training		testing		training		testing	
epochs	errors	mse	errors	epochs	errors	mse	errors
800	65	.110	18	1000	88	.120	21
1200	43	.115	14	2000	43	.108	16
1600	23	.118	13	3000	27	.110	13
2000	16	.116	14	4000	17	.110	13
2400	9	.112	11	5000	9	.109	12
2800	6	.115	14	6000	6	.110	11

variety. Every hidden unit is connected to all of the input units and to all of the output units. The weight patterns of Figure 4 show that each hidden unit tends to form an overall spectrogram template for one or more of the words.

According to the standard intuitive explanation of the behavior of multilayer feed-forward networks, hidden units are supposed to extract meaningful features from the input patterns. These features are then passed on as evidence for the output units to consider as they decide on the network's answer. The intuitive notion of a spectrogram feature generally involves a localized subpattern in the spectrogram. One can force a network to develop localized feature detectors by restricting its connectivity, giving each hidden unit a receptive field that only covers a small region of the input.

Because the total number of its weights is a relatively small multiple of the number of its hidden units, a small-receptive-field network enjoys a large ratio between the information bandwidth of the hidden layer and the total information capacity of the network. Thus, a network with small receptive fields can possess a rich inventory of hidden layer codes to represent subtleties of the input, without being burdened by an excessive number of free parameters that would allow the network to learn its training set by rote.[16]

[16] This assumes that the capacity limitation that forces good generalization is the number of weights. Some networks achieve good generalization by restricting the width of a "bottleneck" hidden layer instead of the weights (Hinton, 1987a).

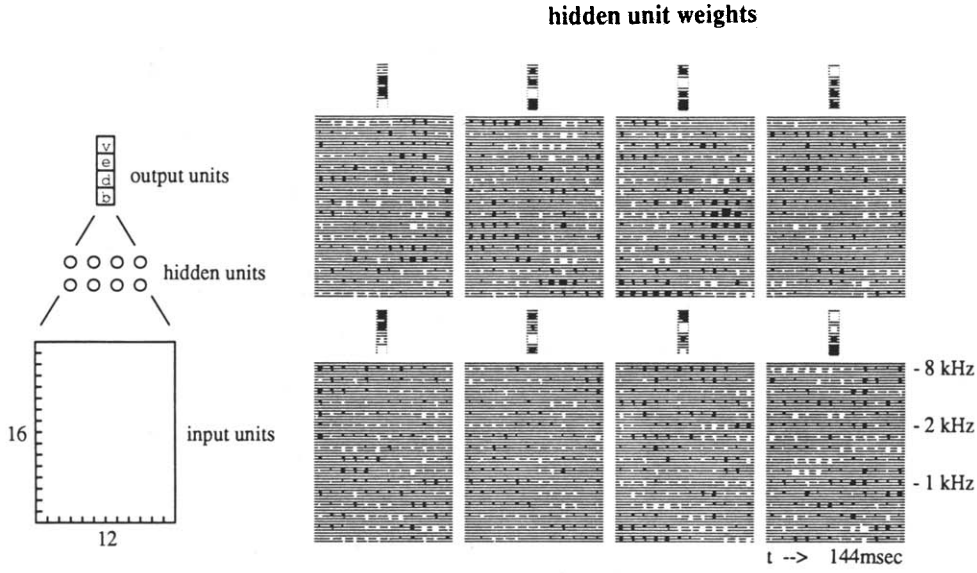

FIGURE 4. A 3-layer network with 8 hidden units. The 8 patterns on the right show the weights learned by each of the 8 hidden units. Each pattern includes connections to the 12 × 16 input array and to the 4 output units.

To determine the benefits of this network architecture, a small-receptive-field network was built with approximately the same number of weights as the fully connected 8 hidden unit network. As shown in Figure 5(a), each hidden unit in the new network was connected to a slice of the input spectrogram that contained only 3 time steps (but all 16 frequency bands). Since there are 10 ways to position a 3-step window on a 12-step input pattern, the input was covered by 10 different time windows. To permit the detection of multiple features in each slice of the input, the network had 3 separate hidden units connected to each of the 10 receptive fields, for a total of 30 hidden units. The 4 output units were connected to all 30 hidden units, so the network contained $30 \times 3 \times 16 + 4 \times 30 = 1560$ weights.

The learning trajectories contained in Table 3 show that this network performed slightly better than the fully connected network according to the mean squared error metric. Although the small-receptive-field architecture did not provide a big improvement over the fully connected architecture on this task, we have found it to be clearly superior on tasks that require a network to discriminate between consonants in multiple vowel contexts, in which case it is useful for the network to be able to represent information about different parts of the spectrogram using separate hidden units.

3.5. Position Independent Feature Detectors

In the small-receptive-field architecture described in the last section, the hidden units are all free to develop weight patterns for detecting the features that are most relevant to the particular portions of the words that lie in the units' receptive fields. This freedom to develop specialized hidden units for analyzing the various parts of the words would be desirable if all of the exemplars of the words had exactly the same alignment relative to the input array. Unfortunately, when a misaligned word is presented as input, the fact that these specialized detectors are hard-wired to the input array means that the wrong detectors will be applied to the wrong parts of the word. One way to eliminate this problem is to force the network to apply the same set of feature detectors to every slice of the input.

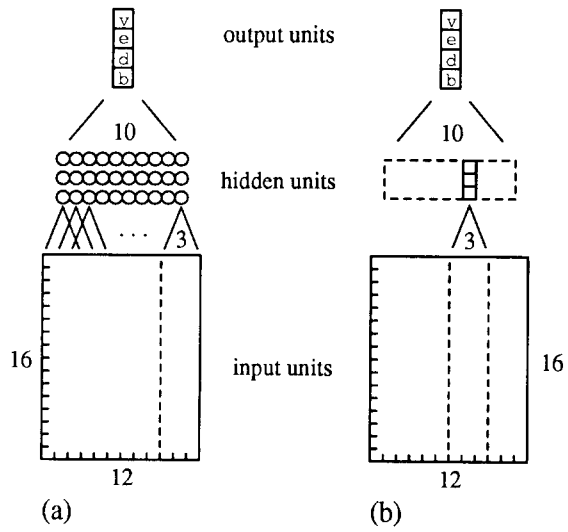

FIGURE 5. Two views of a 3-layer network containing 30 hidden units, each of which is connected to a window of 3 time steps.

A small modification to the back-propagation learning procedure is required to make a small-receptive-field network act in this manner. Consider the network of Figure 5(a), which contains 3 rows of 10 hidden units connected to 10 successive 3×16 windows into the input. The 10 weights connecting the hidden units of a given row to the 10 successive input units representing a given receptive field component[17] are thrown into an equivalence class. After the weights are updated at the end of each iteration of the learning procedure, every weight in an equivalence class is set to the average of the weights in that class (Rumelhart, Hinton, & Williams, 1986). When the network is trained using this rule, all of the hidden units in a given row will have learned the same weight pattern, so the row can be thought of as a single hidden unit replicated 10 times to examine 10 successive input slices for the presence of one feature.

This new interpretation of the small-receptive-field network is shown schematically in Figure 5(b). The network effectively contains only 3 different hidden units. Because each hidden unit is connected to the input units via a 3 by 16 weight pattern, there are $3 \times 3 \times 16 = 144$ weights between the first and second layers. Although the 10 copies of a given hidden unit possess identical weights, they can assume 10 different activation levels to represent the presence or absence of the unit's feature in the 10 slices of the input. Since the activation level of each copy of a hidden unit conveys unique information about the input pattern, every copy gets a separate connection to the output layer. Thus there are $4 \times 10 \times 3 = 120$ weights between the second and third layers.

Experiments with networks containing 4, 6, and 8 replicated hidden units showed that a network with 8 hidden units worked the best on this task. Column 3 of Figure 1 exhibits the activation levels of the $8 \times 10 = 80$ hidden unit copies of this network on four sample words. These hidden unit activation patterns can be thought of as pseudo-spectrograms that have arbitrary features rather than frequency band energies displayed on the vertical axis.

[17] For example, the upper left-hand corner of the field.

3.6. Position Independent Output Units

An analysis of the errors made by the previous networks of this section showed that the most common source of error was incorrect alignment of the utterance on the input array. Because the position of the vowel onset in each utterance was chosen by a mostly accurate Viterbi alignment procedure, there weren't nearly enough different starting points in the training data to allow a network to learn to generalize across time.[18]

To solve this problem, we devised a network that is inherently time-symmetric because it integrates output activations over time. The network contains multiple copies of each output unit. The copies of an output unit apply the same weight pattern to successive narrow slices of the input pattern, attempting to locate a subpattern which is characteristic of the word denoted by that unit. During learning, the equivalence class rule described in section 3.5 constrains the weights of all of the copies of each output unit to be the same.

A 2-layer version of this network is shown in Figure 6. Whereas the output units of the simple 2-layer network of Figure 3 had been connected to the entire input layer, the output units in this network are connected to narrow receptive fields that only cover 5 time steps. Since there are 8 ways to position a 5-step window on a 12-step pattern, the network contains 8 copies of each output unit. When an input is presented to the network, each of the $4 \times 8 = 32$ output unit copies is activated by an amount that indicates the copy's confidence that its word is present, based on the evidence that is visible in its receptive field. The overall value of each of a network's outputs is defined to be the sum of the squares[19] of the activations of all of that output unit's temporal

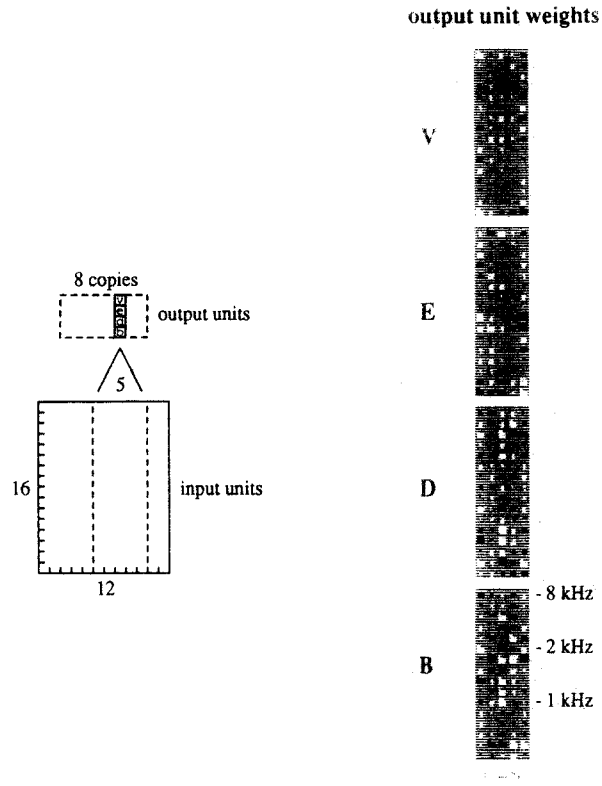

FIGURE 6. A 2-layer network whose output units are replicated across time. The weight patterns shown on the right are applied by 8 copies of the 4 output units to successive 5-step windows into the input. These weight patterns are cleaner than those of the conventional 2-layer network pictured in Figure 3 because they don't have to account for time shifts.

copies, so the computation performed by the network is a mapping from spectrograms to real-valued 4-tuples, as always.

While it is clear that a network with this structure and an appropriate set of weights could perform shift-invariant pattern recognition, it is less obvious how the network could acquire such a set of weights given the lack of temporal supervision caused by the summation of the activations of the multiple output unit copies. The network only receives a single error signal for each training token, which must somehow guide the development of all of the network's replicated weights, even though some of the weight copies are processing portions of the input pattern that are useful for identifying the word, and others are processing portions of the input that are completely irrelevant. By combining evidence from the entire corpus of training data, the network does ultimately learn which of the subpatterns possess the most predictive power, at which point the detectors for those subpatterns can be made very precise because they don't have to account for shifted versions of the patterns.

For example, the weights shown on the right side

[18] Hinton (1987a) demonstrated that a network *could* learn to perform position-independent recognition of bit patterns from scratch when the training set provided nearly complete coverage of the cross product of patterns and positions. This would be infeasible when training a network to perform a real-world task.

[19] The motivation for squaring the activations was to allow the activation of the output unit replica that found the best match to predominate in the overall answer. To find out whether this effect was really beneficial, we trained a toy network consisting of 5 input units and 2 replicated output units to distinguish between the patterns 101 and 110 regardless of the patterns' alignment on the 5 input units. Using the squared activation rule, the network took 311 iterations to learn the task, employing weights whose average size was 0.99. When we tried the same task using the sum of the output units' unsquared activations as the network's outputs, the network needed 558 iterations and weights of average size 1.23 to solve the problem, so the squared activation rule appeared to be superior. The performance of the two rules has not been compared on a speech task, but the unsquared activation rule worked well in Waibel, Hanazawa, Hinton, Shikano, and Lang (1987).

of Figure 6 allowed the replicated 2-layer network to correctly classify 94% of the training cases and 91% of the test cases of the reallocated version of the data set. A comparison of the weight patterns learned by this network and the fixed position 2-layer network of Figure 3 is illuminating. In the new network, the rising F_2 of the **B** pattern and the high-frequency burst of the **D** pattern are cleanly localized in time, while in the old network, these events were smeared over two or three time steps. Becuase the replicated network has time symmetry built into its architecture, it no longer has to compensate for variable word alignment by blurring its weight patterns, thus allowing the network to analyze the critical portions of the spectrograms in more detail.

Having demonstrated the value of the time-symmetric replicated output unit architecture with this 2-layer network, we next applied the idea to a more powerful 3-layer network. The first layer of the 3-layer replicated network consisted of 192 input units encoding a spectrogram. The hidden layer contained 10 copies of 8 hidden units that were each connected to 3 frames of the input. The third layer had 6 copies of the 4 output units, each looking at 5 frames of the pseudo-spectrogram generated by the hidden layer.

The weight space of a highly constrained multi-layer network is more difficult to explore than that of a simpler network, requiring smaller and more carefully chosen learning parameters. More than 20,000 iterations with peak parameters of $\{\varepsilon = .001$ $\alpha = .95\}$ were needed to tune the network into a model that accounted for 93% of the 668 training cases and 93% of the 100 test cases of the modified task. The activation patterns of this network's output units on four sample utterances are pictured in column 4 of Figure 1. The detectors for **E** and **V** show little time locality, apparently utilizing global characteristics of the tokens. However, the network recognizes the stops **B** and **D** by examining the vowel onset, so the output activation patterns for them clearly show the alignment of the utterance. Notice that the network fired later on the **B** than it did on the **D**, as one would expect from looking at the corresponding waveforms.

It is significant that the network learned to locate and analyze the consonant-vowel transition region for these words, despite the fact that the training environment did not include any explicit information about the usefulness of this region of the word, much less any information about where to find the vowel onset in a given utterance. The network's success at learning to find and exploit the most informative region of each input pattern suggests that the Viterbi alignment initially used to clip a 144 ms salient section from each utterance was unnecessary; the network might have done just as well on complete, unsegmented words. This idea is explored in section 4.

3.7. An Implementational Detail

So far, we have glossed over the details of training a network with replicated output units. The error of a back-propagation network on a given case is a function of the differences between the network's actual output values o_j and the corresponding target values d_j.

$$E = \frac{1}{2} \sum_j (o_j - d_j)^2.$$

In an ordinary back-propagation network, the output values o_j are just the activation levels of the network's output units y_j. Plugging this fact into the definition of E and then differentiating by y_j gives us the partial derivative of the error with respect to the activations of the output units. These values provide the starting point for the backward pass of the learning algorithm.

$$\frac{\partial E}{\partial y_j} = y_j - d_j.$$

In the replicated network, each output value of the network is the sum of the squares of the activations of several temporal replicas of an output unit.

$$o_j = \sum_\tau y_{j\tau}^2.$$

Plugging this into the definition of E and then differentiating yields the partial derivative of the error with respect to activation of the replica of unit j at time τ.

$$\frac{\partial E}{\partial y_{j\tau}} = 2y_{j\tau} \left(\left(\sum_\tau y_{j\tau}^2 \right) - d_j \right).$$

3.8. Time-delay Neural Networks

The architecture of our best **BDEV** network was originally formulated in terms of replicated units trained under constraints which ensured that the copies of a given unit applied the same weight pattern to successive portions of the input (Lang, 1987). Because the constrained training procedure for this network is similar to the standard technique for recurrent back-propagation training (Rumelhart, Hinton, & Williams, 1986), it is natural to re-interpret the network in iterative terms (Hinton, 1987b). According to this veiwpoint, the 3-layer network described in section 3.6 has only 16 input units, 8 hidden units, and 4 output units. Each input unit is connected to each hidden unit by 3 different links having time delays of 0, 1, and 2. Each hidden unit is connected to each output unit by 5 different links having time delays of 0, 1, 2, 3, and 4. The input spectrogram is scanned one frame at a time, and activation is iteratively clocked upwards through the network.

The time-delay nomenclature associated with this iterative viewpoint was employed in describing the experiments at the Advanced Telecommunications Research Institute in Japan which confirmed the power of the replicated network of section 3.6 by showing that it performed better than all previously tried techniques on a set of Japanese consonants extracted from continuous speech (Waibel et al., 1987).

3.9. Related work

The idea of replicating network hardware to achieve position independence is an old one (Fukushima, 1980). Replication is especially common in connectionist vision algorithms where local operators are simultaneously applied to all parts of an image (Marr & Poggio, 1976). The inspiration for the external time integration step of our time-delay neural network (TDNN) was Michael Jordan's work on back-propagating errors through other post-processing functions (Jordan, 1986).

Waibel (1989) describes a modular training technique that made it possible to scale the TDNN technology up to a network which performs speaker dependent recognition of all Japanese consonants with an accuracy of 96.7%. The technique consists of training smaller networks to discriminate between subsets of the consonants, such as **bdg** and **ptk**, and then freezing and combining these networks along with "glue" connections that are further trained to provide interclass discrimination.

Networks similar to the TDNN have been independently designed by other researchers. The time-concentration network of Tank and Hopfield (1987) was motivated by properties of the auditory system of bats, and was conceived in terms of signal processing components such as delay lines and tuned filters. This network is interesting because variable-length time delays are learned to model words with different temporal properties, and because it is one of the few connectionist speech recognition systems actually to be implemented with parallel hardware instead of being simulated by a serial computer.

An interesting performance comparison between a TDNN and a similarly structured version of Kohonen's LVQ2 classifier on the ATR **bdg** task is reported in Mcdermott and Katagiri (1989). The same 15 × 16 input spectrograms were used for both networks. In the LVQ2 network, a 7-step window (which is the amount of the input visible to a single output unit copy in the TDNN) was passed over the input, and the nearest of 150 LVQ2 codebook entries was determined for each input window position. These codebook entries were then summed to provide the overall answer for a word. The replicated LVQ2 network achieved nearly identical performance to the TDNN with less training cost, although recognition was more expensive.

An comprehensive survey of the field of connectionist speech recognition can be found in Lippmann (1989).

3.10. Multiresolution Training

In order to facilitate a multiresolution training procedure, the time-delay network of section 3.6 was modified slightly so that the widths of its receptive fields would be divisible by 2. While the network had previously utilized hidden unit receptive fields that were 3 time steps wide and output unit receptive fields that were 5 time steps wide, its connection pattern was adjusted to make all of its receptive fields 4 time steps wide (see Figure 7(b)). Because this modification would have increased the total number of weights in the network, the number of hidden units was decreased from 8 to 6. After these changes, the network contained 490 unique weights. The half-resolution version of the network shown in Figure 7(a) was also constructed. This network covered the input patterns using six 24-ms frames rather than the twelve 12-ms frames of the full-resolution network. In the half-resolution version of the network, the receptive fields were all 2 frames wide.

Multiresolution training is conducted in two stages. In the first stage, the half-resolution network is trained from small random weights on half-resolution versions of the training patterns until its training set accuracy reaches a specified level. Then, the network's weights are used to initialize the full-resolu-

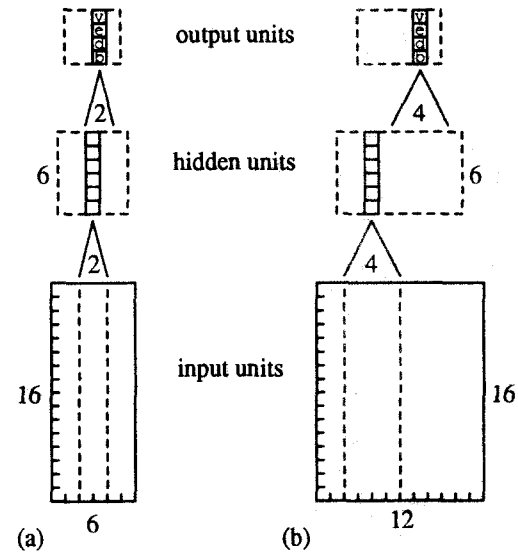

FIGURE 7. (a) A 6-step half-resolution TDNN. (b) A 12-step TDNN initialized by network (a). Although the 6-step network contains fewer time steps, each of these steps represents more time. The duration of the input patterns is 144 ms for both networks.

tion network, which is further trained on full-resolution versions of the training patterns. Figure 8 illustrates this two-stage training procedure, which saves time because the half-resolution network can be simulated with only one-fourth as many connections as the full-resolution network.

3.11. Discussion

The architectural experiments described earlier in this section were performed on a modified version of the **BDEV** task in which the data had been reapportioned between the training and testing sets. An additional experiment was required to measure the time-delay network's performance on the Viterbi-aligned version of the task with the original training and testing sets.

Starting from random weights distributed uniformly on the interval $(-0.01, +0.01)$, the low resolution TDNN of Figure 7(a) was trained of the 372 training patterns until its accuracy reached 85%. This required 3000 epochs using the parameter schedule of Table 4(a). The network's weights were then transferred to the high resolution network, and learning continued. The previously employed target activations of 0.2 and 0.8, which are reputed to improve generalization, were abandoned in favor of the naive target activations of 0.0 and 1.0, which actually work better for this task. Peak generalization occurred after the high-resolution network had been trained for 10,000 epochs, at which point the network got 95.4%

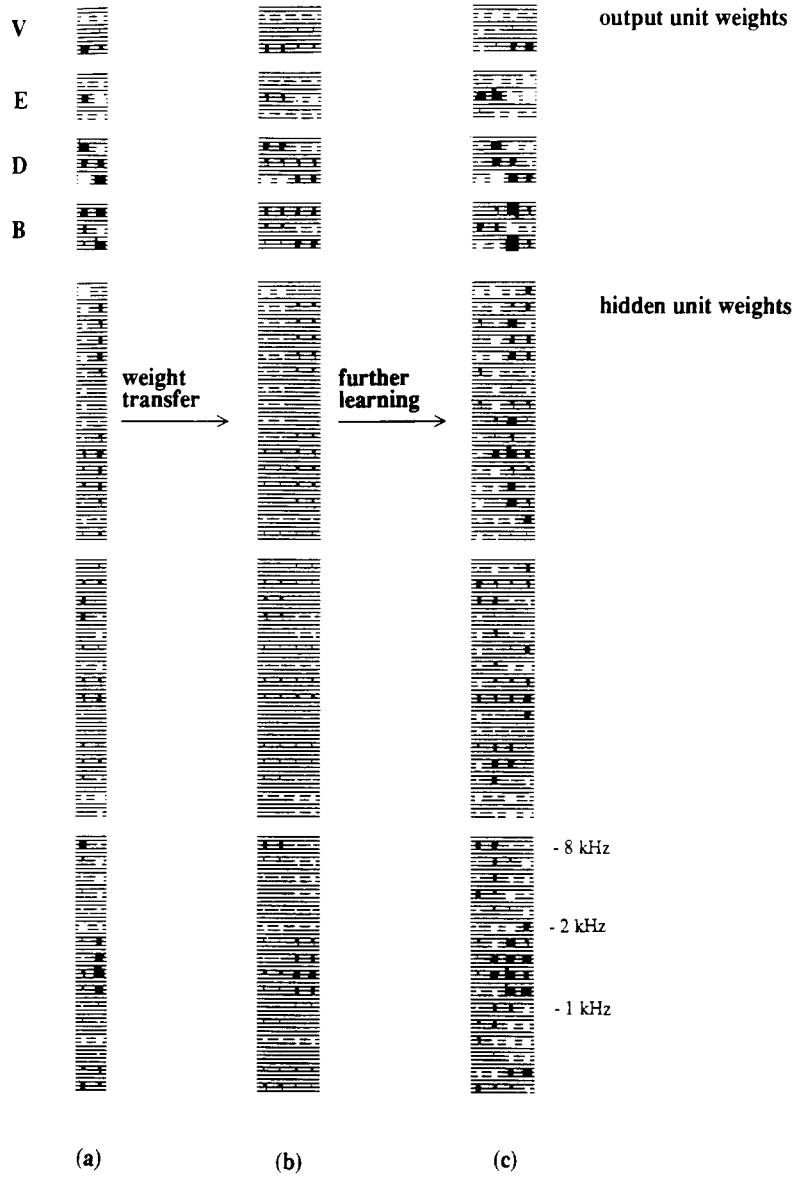

FIGURE 8. (a) Weights taken from a half-resolution TDNN to generate (b) initial weights for a full-resolution TDNN, which learned (c) these full-resolution final weights. For clarity, only half of the networks' six hidden units are shown here.

TABLE 4
The parameter schedule used for multi-resolution training. The low initial momentum allows the networks to find the bottom of a ravine. Because the second stage network starts out with weights learned by the first stage network, it is already located in a ravine, and can accelerate more rapidly.

(a)	First-stage network				(b)	Second-stage network			
epoch	0	200	1000	2000...	0	50	100	200	1000...
epsilon	.0001	.0001	.0005	.0010	.0001	.0001	.0005	0010	
momentum	.05	.9	.95	.95	.5	.9	.95	.95	

of the training cases and 91.4% of the testing cases correct. During an additional 10,000 epochs of training, the network's performance increased to 98.1% on the training set, but generalization fell to 88.1%.

The baseline 2-layer network of section 3.1 reached an 86.9% generalization peak when trained under the same conditions, so the peak performance of the time-delay network was 4.5% better. Based on this comparison, it seems like the additional complexity of the multilayer time-delay network did not buy very much. However, the time-delay network's ability to learn to sharply focus on the best discrimination cues in an utterance, as evidenced by the **B** and **D** output activation patterns in column 4 of Figure 1, are an indication that the TDNN was underutilized on the simplified, Viterbi-aligned version of the task which was the domain for all of the experiments described up to this point. When the original, unsegmented version of the task was tackled using the methods described in the next section, the peak generalization of the time-delay network actually increased to 92%, while the generalization of the simple 2-layer network plummeted to 61%.

4. BEYOND SEGMENTATION

In order to simplify our initial foray into connectionist speech recognition, we had tried to avoid the time alignment problem by using short (144 ms) sections of each utterance selected by the IBM hidden Markov model. As explained in section 3.6, this segmentation was generally accurate, but there were several cases where the position of the vowel onset differed from the norm, defeating networks that had learned to expect the most prevalent alignment. To solve this problem, a network was built that summed the squares of the activations of multiple output unit copies which could each see a different portion of the input pattern. During training, the overall network gradually learned to locate and focus on the most relevant portions of the utterance, ignoring the rest.

Thus armed with a network that could learn to find and classify the relevant portion of a long utterance, it was feasible to attack the same full-length utterances that Peter Brown had used in his hidden Markov model experiments. These recordings for the **BDEV** set ranged in length from 0.3 to 6.4 seconds, and averaged 1.1 seconds. In each recording, the word itself was fairly short, and was preceded and followed by "silence," which was actually rather noisy, containing knocking sounds and background conversation.

In principle, we could have trained and tested a gigantic version of our replicated network on the full-blown recordings, which would have been desirable since systems generally work best when they are trained on a version of the task that exactly corresponds to the one encountered in performance. However, in the interest of speed and convenience, we instead approximated that approach by training the network on a new set of wider consonant-vowel transition regions selected by an *ad hoc* energy-based segmenter, augmented with "counter-example" regions randomly chosen from the leftover portion of each utterance (which consisted of background noise and the trailing part of the E vowel). Testing was performed by scanning the network across the complete, unsegmented version of an utterance, looking for the maximum output activation level which resulted. When trained and tested in this manner, the network achieved better peak generalization than it had on when trained and tested on the 144 ms Viterbi-selected segments, probably because the system no longer depended on a potentially errorful segmentation during recognition.

4.1. Training on Heuristically Selected Segments

Because of the alignment-invariance of the replicated TDNN network architecture, precise segmentation of the training data is not necessary; it is sufficient to have a section of each utterance that somewhere contains enough information to discriminate between the alternatives. On the **BDEV** task, this information is concentrated in the consonant-vowel transition region. Assuming that most of the energy in these words is contained in the vowel, the consonant-vowel transition can be located by an *ad hoc* program that finds the beginning of the largest concentration of energy in an utterance.

Figure 9 shows how such a program works on an

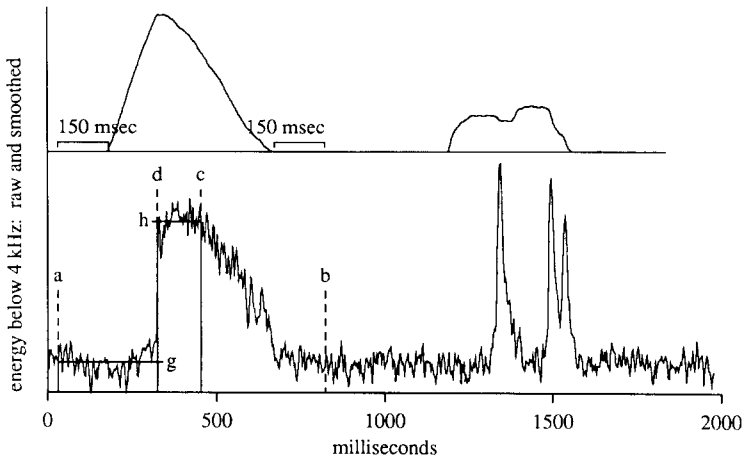

FIGURE 9. This simple heuristic, which locates the beginning of the largest energy hump in an utterance, was used to extract vowel-onset segments in order to expedite training. The best testing performance, however, was achieved by presenting complete, unsegmented spectrograms to the network.

instance of the word **B**. Starting from a raw spectrogram containing 128 frequency bands ranging up to 8 kHz, the total energy below 4 kHz is computed for every 3 ms time step, and then the energies are normalized by subtracting the smallest value from all of the others, yielding the lower curve in the diagram. (The two spikes near the end are background noise.) This curve is then smoothed with a 150 ms window and thresholded with the median smoothed value to obtain the top curve, which in this case contains two contiguous energy blobs. According to our assumption, the larger blob represents the vowel. The boundaries of this blob are expanded by 150 ms in each direction to obtain the points *a* and *b*.

Returning to the original, unsmoothed curve, point *c* is then fixed at the energy midpoint of the interval *ab*, so that the area under the curve on the interval *ac* equals the area under the curve on the interval *cb*. Finally, point *d* is scanned across the interval *ac* while values are computed for $g(d)$ and $h(d)$, which are the average energies on the intervals *ad* and *dc*, respectively. The output of the program is the value of *d* for which $h(d) - g(d)$ is maximized.

From each training utterance, the section from time $d - offset$ to time $d - offset + length$ was extracted to form a training segment, where *d* was the heuristically determined vowel onset position in that utterance, *offset* was 120 ms, and *length* was 216 ms. The 50% increase in length from the previously used value of 144 ms was motivated by the reduced precision of the new *ad hoc* vowel finding rule. The longer training segments also increased the amount of irrelevant material that the network would have to learn to ignore.

Despite their increased length, the new training patterns couldn't provide a comprehensive picture of the acoustical content of the training corpus because they were all positioned around the consonant-vowel transitions of the words. Therefore, after the heuristic segmentation program had extracted a slice containing the vowel onset from a given utterance, it randomly selected an additional 216 ms section from the leftover portion of that utterance. These "counter-example" segments were placed in the training set with target values of zero for all of the network's output units. The augmented collection of training segments constructed by this technique contained essentially the same information as the raw, unsegmented corpus, but with reduced redundancy; while the tiny consonant-vowel transition region of every word contained valuable information, the long stretches of background noise and vowel in the utterances were comparatively uniform, and could be adequately characterized by random samples.

4.2. Testing on Complete Utterances

Testing was accomplished by a scanning procedure in which an unsegmented utterance was divided into consecutive, overlapping 216 ms input patterns, and the network was repeatedly applied to convert this sequence of patterns into a sequence of output vectors for the utterance.[20] The 216 ms input window was shifted by 12 ms (or 1 input frame) between successive computations. The largest single vector component observed in the output vector sequence for a given utterance determined the classification of that utterance.

4.3. A Question of Supervision

This section began with the claim that the replicated TDNN architecture could handle the unsegmented

[20] When a time-delay network was used, each value in an output vector was the sum of the squares of the activation levels of several output unit copies.

version of the **BDEV** task because it did not require supervision in the time domain, that is, it did not need to be told the location of the discrimination cues in a given utterance. In the name of efficiency, redundancy in the training corpus was then reduced by first extracting a 216 ms slice around each hypothesized vowel onset, and then randomly selecting an additional 216 ms slice from each word on which the network would be trained to output a vector of zeroes. This training method sounds suspiciously supervised, calling into question the need for a network as powerful and expensive to train as a TDNN.

To find out whether this training method would eliminate the need for a network that can learn to find the most meaningful event in a longer input pattern, a conventional network and a time-delay network were both trained on the new set of training segments and then tested on the full-length training utterances.

The conventional, fully connected network had an 18 × 16 input array, 8 hidden units, 4 output units, and 2358 weights. The network was trained twice: once on the set of 216 ms vowel-onset segments alone, and once on those segments plus the counter-example segments randomly chosen from the leftover portions of the utterances. After each 2000-epoch training session using a parameter schedule which peaked at $\{\varepsilon = .0005, \alpha = .95\}$ after 400 epochs, the network's performance on the training set was measured in two different ways. First, the vowel-onset segments of the training set were classified using the best-guess rule. Second, the network was scanned across the full-length versions of the training utterances and the maximum output activation was noted.

A 216-ms version of the time-delay network of Figure 7 was then evaluated. As before, the output values produced by this network were the sums of the squares of the activations of several output unit copies, each of which could only see only 84 ms of the input. However, there were now 12 copies of each output unit, rather than just 6. Using the multiresolution training paradigm described in section 3.11, the network was trained once on vowel-onset segments alone, and again on vowel-onset segments plus counter-example segments. After both runs, the network's recognition accuracy was measured on the vowel-onset segments and on the full-length versions of the training utterances.

Table 5 contains the results of this experiment. When trained on vowel-onset segments alone, the conventional network learned nearly all of the training patterns, but was unable to correctly classify more than a third of the corresponding full-length utterances. By contrast, the time-delay network's training set performance only fell slightly when going from the vowel-onset segments to the complete utterances. When trained on vowel-onset segments together with counter-example segments, the error rate of the conventional network on full-length utterances was nearly halved, but was still an order of magnitude higher than on the training segments. Under the same conditions, the time-delay network actually performed *better* on the full-length utterances than on the segments with which it had been trained.

To provide some intuition into these performance numbers, Figure 10 was made. The left-hand plots, which correspond to training on the vowel-onset segments alone, have a dramatically different character for the two networks. The time-delay network, which had already learned to isolate the most informative region contained in each 216 ms segment, behaved in a controlled manner when confronted with the full-length versions of the utterances, while the outputs of the conventional 3-layer network fired erratically throughout the utterances as random noise stimulated its comparatively undiscriminating feature detectors.

The right-hand plots show that the use of counter-example segments in the training process cleaned up the firing patterns of both networks, eliminating spurious firings in the vocalic and background noise portions of the words. Still, the conventional network fired erratically when the vowel-onset regions of the word was shown to the network in novel positions, while the time-delay network was unfazed because its replicated architecture allowed it to recognize known patterns imbedded in previously unseen material.

While the training method described in this sec-

TABLE 5

This table shows how networks with two different architectures fared when forced to classify unsegmented spectrograms after being trained on short segments of those spectrograms. Both networks were trained to recognize more than 90% of the training segments which were extracted from around the consonant-vowel transition region. The conventional network was unable to transfer its knowledge of the training segments to the full-length task, while the time-delay network's performance actually improved when given the full-length training utterances to classify.

Counter-examples used?	No		Yes	
Type of input pattern:	Segments	Full-length	Segments	Full-length
3-layer conventional net	99%	34%	97%	54%
3-layer time-delay net	94%	91%	92%	94%

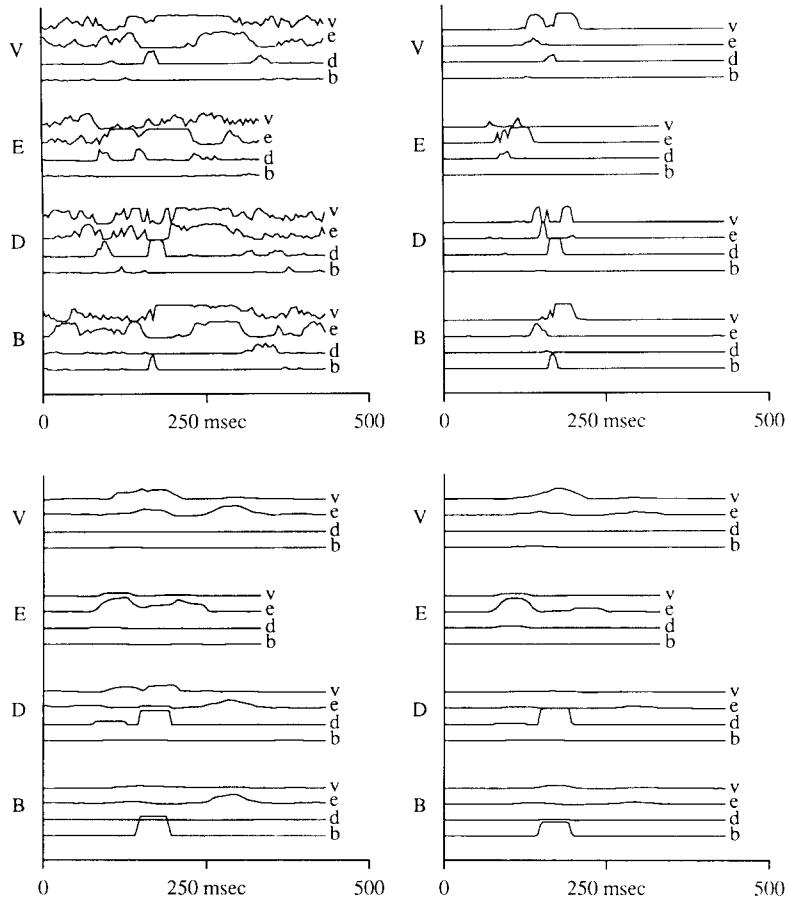

FIGURE 10. Output activation traces obtained by scanning a network across full-length training utterances. The top two plots are for a 3-layer fully-connected network, while the bottom two plots are for a 3-layer time-delay network. The plots on the left resulted from training on 216 ms segments extracted from around the consonant-vowel transition of each word, while the plots on the right resulted from training on those segments plus counter-example segments randomly chosen from the leftover portions of the words.

tion might *appear* to be closely supervised in time (i.e., the network should fire here, but not there), the training segments are longer than the events which are significant for this task, and are not aligned with any degree of precision. A conventional network is unable to learn enough from these segments to successfully classify unsegmented utterances, while the time-delay network is fully capable of learning from patterns that contain small pieces of crucial information in unknown positions.

5. PERFORMANCE

During the architectural study of section 3, networks were rated according to peak generalization; at regular intervals while each network was being trained, its performance on the test set was measured. These scores would typically rise to a maximum value and then fell again as the network learned facts about the training set which were not true of the test set. Although peak generalization is a useful measure for comparing the utility of different network architectures, it fails to capture the flavor of a real application, where training must be completed without any reference to the ultimate testing set.

When one is lucky enough to have a clean and consistent training set, good generalization may be achieved by simply training the network until it makes no errors on that set. When the training set contains outliers that result in diminished generalization, as does our **BDEV** training set, a more insightful halting methodology is required.

The standard technique for deciding when to stop an excessively powerful learning procedure is to set aside part of the training set as a "check set" to be used for tuning. This leaves a reduced collection of cases for actually training the network's weights. When the training data is organized in this way, the peak generalization method can be used to decide when to stop learning. The network is trained on the reduced training set until it achieves peak performance with respect to the check set. Then the network's performance on the test set can be measured exactly once, yielding a true generalization score. Although this technique works, the reduction in size of the

training set can be a disadvantage when the training set is small to begin with.

In a separate set of experiments, we proposed and tested several decision rules that did not result in a loss of training data. The simplest technique was to first perform a check set run to estimate the shape of the network's generalization curve, and then retrain the network using all of the training data until its mean squared error reached the level at which the best estimated generalization had occurred during the check set run.

In order to obtain an official **BDEV** performance rating for the recognition system described in section 4, a check set was created by setting aside 100 of the 216-ms vowel-onset segments (25 per word) and 100 of the counter-example segments. Using the multi-resolution training procedure described in section 3.11,[21] a 216-ms version of the network of Figure 7 was trained on the remaining training segments for 10,000 epochs, which was long enough to see that the network's mean squared error on the check set was rising from the minimum value that it had reached at 6,000 epochs.

The network was then retrained on the full training set of 216-ms heuristically selected vowel-onset segments plus counter-example segments, including the segments which had been temporarily removed to form the check set. This retraining started from the same set of half-resolution weights which had been used during the check set experiment, and employed identical learning parameters. At the end of 6,000 epochs, the network showed the same mean squared error on the training set that it had after 6,000 epochs during the check set run, so training was halted, and the network's scanning mode generalization to the full-length utterances of the real testing set (which otherwise was not touched during this experiment) was measured to be 90.9%

This true generalization score is not only much better than the standard IBM hidden Markov model's 80% **BDEV** accuracy, it also compares favorably with the estimated 89% **BDEV** performance of the IBM system when it had been enhanced with the continuous parameter, MMIE acoustic model that is the main result of Peter Brown's thesis. It should be emphasized that our network's 90.9% test set performance was attained on exactly the same noisy, variable-length recordings that the hidden Markov models were faced with. Thus, we have shown that without the benefit of a presegmentation step, a properly designed neural network is capable of recognition performance on a highly confusable small-vocabulary multispeaker recognition task that is competitive with the best achieved by an enhanced hidden Markov model which was also specially designed for the task.

It is interesting to contrast the methods by which these two systems achieved their good performance on this task. The hidden Markov system had the advantage of being able to model the global temporal structure of the utterances. By recognizing the vocalic and background portions of the utterances, the HMM was able to accurately position the consonant models that actually provided the discrimination between the words. The time-delay network, while unaware of everything about an utterance that was not directly under its nose at a given moment, used its superior discrimination power to ignore everything but the maximally informative consonant-vowel transition in each utterance.

6. CONCLUSION

The primary result of this paper is the time-delay neural network architecture. This architecture, which factors out the position of features in its input patterns by summing the activations of replicated output units connected to small receptive fields, has benefits that extend far beyond the property that input registration errors are tolerated. The temporally unsupervised TDNN training procedure amounts to a small-scale iterative labeling/training loop which permits the network to acquire extremely sharp feature detectors.

The fact that a time-delay network can learn precise weight patterns from imprecisely prepared training examples makes the system an attractive foundation for the construction of a practical recognition system. Also, the number of weights that must be stored and convolved with the input stream during recognition is small, and the network's narrow receptive fields require only short input buffers, thereby minimizing both the memory requirements and latency of such a system.

The decision rule which was used to halt the back-propagation learning procedure in section 5 also has practical benefits. At the expense of training the network twice, once on a version of the training set from which a check set had been withheld, and again on the full training set up to the point had which yielded the best check set generalization, this method permitted an informed decision to be made about when to stop training without reducing the size of the training set.

[21] During this run, the momentum parameter was set to .98 after 1000 epochs rather than .95 in the interest of faster learning. Weight decay was performed, with $h = .002$. During the experiments described in earlier sections, weight decay was performed by multiplying every weight by $(1 - h)$ after each weight step. For this experiment, the simulator was modified to perform weight decay by adding a decay vector $-h\mathbf{w}$, where $\mathbf{w}$ is the weight vector, to the gradient before computing the actual weight step using momentum. This implementation has the desirable property that weight decay does not interact with momentum.

Finally, it is hoped that the tour through a portion of network design space in section 3 provided some insight into the issues that are nearly always relevant to the construction of a successful network for a given application. Consideration must be given to a network's information capacity relative to the amount of training data, to the bandwidth of a network's information channels relative to the sorts of internal codes that will be needed, and to the computational power of a network compared to the complexity of the input-output mapping that it is being asked to perform. It is also important to consider whether a network can learn the essential properties of a task from the training data that is actually available. When a network's architecture permits a desirable mode of operation, but that mode cannot be learned from the training set, the network must be redesigned so that it will behave correctly despite the inadequacies of the training data. Although back-propagation is often touted as a black-box learning procedure, the best results are obtained when it is used to tune the best possible network for a given task.

REFERENCES

Bahl, L. R., Jelinek, F., & Mercer, R. (1983). A maximum likelihood approach to continuous speech recognition. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, **PAMI-5**, 179–190.

Baker, J. K. (1975). Stochastic modeling for automatic speech understanding. In R. Reddy (Ed.), *Speech recognition* (p. 521). New York: Academic Press.

Brown, P. F. (1987). *The acoustic-modeling problem in automatic speech recognition*. Unpublished doctoral dissertation, Carnegie-Mellon University, Pittsburgh, PA.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. New York: John Wiley & Sons.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, **36**, 193–202.

Hinton, G. E. (1987a). Learning translation invariant recognition in a massively parallel network. In G. Goos & J. Hartmanis, (Ed.), *PARLE: Parallel architectures and languages Europe*. Berlin: Springer-Verlag.

Hinton, G. E. (1987b). *Connectionist learning procedures* (Tech. Rep. CMU-CS-87-115). Pittsburgh, PA: Carnegie-Mellon University.

Hinton, G. E., & Plaut, D. C. (1987). Using fast weights to deblur old memories. *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*.

Jordan, M. (1986). Attractor dynamics and parallelism in a connectionist sequential machine. *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum.

Lang, K. J. (1987). *Connectionist speech recognition*. Ph.D. thesis proposal, Carnegie-Mellon University.

Lippmann, R. P., & Gold, B. (1987). Neural net classifiers useful for speech recognition. *1st International Conference on Neural Networks* (417–426). San Diego, CA: IEEE.

Marr, D., & Poggio, T. (1976). Cooperative computation of stereo disparity. *Science*, **194**, 283–287.

Mcdermott, E., & Katagiri, S. (1989). Shift-invariant, multi-category phoneme recognition using Kohonen's LVQ2. *IEEE International Conference on ASSP* (pp. 81–84). Glasgow, Scotland.

Parker, D. B. (1985). *Learning-logic* (Tech. Rep. TR-47). Cambridge, MA: Sloan School of Management, Massachusetts Institute of Technology.

Plaut, D. C., Nowlan, S. J., & Hinton, G. E. (1986). *Experiments on learning by back-propagation* (Tech. Rep. CMU-CS-86-126). Pittsburgh, PA: Carnegie-Mellon University.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, **323**, 533–536.

Tank, D. W., & Hopfield, J. J. (1987). Neural computation by concentrating information in time. *Proceedings of the National Academy of Sciences, USA*, **84**, 1896–1900.

Viterbi, A. J. (1967). Error bounds for convoluted codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, **IT-13**, 260–269.

Waibel, A. (1989). Modular construction of time-delay neural networks for speech recognition. *Neural Computation*, **1**(1), 39.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., & Lang, K. (1987). *Phoneme recognition using time-delay neural networks* (Tech. Rep. TR-I-0006). Japan: Advanced Telecommunications Research Institute.

Werbos, P. J. (1974). *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. Unpublished doctoral dissertation, Harvard University, Massachusetts.