Neural Networks, Vol. 3, pp. 453–465, 1990
Printed in the USA. All rights reserved.

ORIGINAL CONTRIBUTION

Speaker-Independent Isolated Digit Recognition: Multilayer Perceptrons vs. Dynamic Time Warping

L. Bottou and F. Fogelman Soulié

LRI, University of Paris XI, Orsay

P. Blanchet and J.S. Liénard

LIMSI, Orsay

(*Received* 4 *April* 1989; *revised and accepted* 31 *October* 1989)

Abstract—*Former experiments have shown the benefit of using specific multi-layer architectures, the so-called time delay neural networks, for phoneme recognition (Waibel, Hanazawa, Hinton, Shikano, & Lang, 1988). Similar experiments on a speaker-independent task were also performed on a small set of minimal pairs (Bottou, 1988). In this paper we focus on a speaker-independent, global word recognition task with time delay networks. We first describe these networks as a way for learning feature extractors by constrained back-propagation. Such a time-delay network is shown to be capable of dealing with a near real-sized problem: French digit recognition. The results are discussed and compared, on the same data sets, with those obtained with a classical time warping system.*

Keywords—Speech recognition, Isolated digits recognition, Speaker independence, Time delay neural network, Dynamic time warping, Hidden cells states clustering.

1. INTRODUCTION

Neural networks are now well known as powerful learning tools in a variety of tasks related to automatic speech recognition problems, where they have achieved encouraging results (Bridle & Moore, 1984) (Prager, Harrison, & Fallside, 1986) (Kohonen, 1988) (Watrous & Shastri, 1987) (Bourlard & Wellekens, 1988).

However, speech recognition is a difficult task. Problems such as speaker independence, continuous speech, noisy environment, vocabulary size, signal variability, coarticulation effect, etc. remain partly unsolved. All the existing methods successfully overcome some of the difficulties but fail solving the others. Our aim here was to compare neural networks, and more precisely multi-layer perceptrons (MLPs), to classical methods on a widely studied and well-mastered task for today's speech recognition systems.

Two main techniques are frequently used: dynamic time warping (DTW) and hidden Markov models (HMM). For a review of those techniques, see Levinson (1985).

An efficient DTW system, has been developed for some years at LIMSI (Gauvain, 1986; Gauvain, Mariani, & Liénard, 1983). Its performances have been shown to be state of the art on various databases (Quenot, Gauvain, Gangolf, & Mariani, 1989). Furthermore, whole word recognition experiments run with HMM and DTW have shown that the results obtained by these two methods are basically equivalent, despite evident differences in their algorithmic or hardware implementations. We thus compared our MLPs to this DTW system on the same speaker-independent digit recognition problem.

The first MLPs developed for speech recognition tasks were fully connected, with usually one hidden

This work was partly supported by Brain project n°ST2J-0418-C, and Esprit project N°2059 "Pygmalion". L.B. also is supported by DRET grant n° 87/808/19 and P.B. by DRET grant n°87/100.

One of us (L.B.) gratefully acknowledges helpful conversations with Geoffrey Hinton and Yann Le Cun, during a stay at the University of Toronto.

Requests for reprints should be sent to Léon Bottou, LRI, Bat 490, Université de Paris Sud, 91405 Orsay Cedex, France.

layer. Such networks have performed correctly on various small-sized problems: (Elman & Zipser, 1987) (Lippman & Gold, 1987; Lubensky, 1988). However, on real-sized data, those architectures are very inefficient. The number of training examples is too small to succeed in specifying the whole set of parameters in the network. As a consequence, generalization is relatively poor, unless a large enough set of examples has been available for training. But in that case, learning would take very long.

One way to reduce the complexity of the network is to use local connections, that is, local fields. Hidden units are thus assigned local interest tasks, that is, feature extraction. Since we cannot ensure very precise time alignment of the signal, and since also the speech signal can be significantly stretched in time without altering its meaning, we would like to design time invariant feature extractors. This led to the idea of building networks with position independent local fields or so-called time-delay neural networks or TDNN (Waibel, Hanazawa, Hinton, Shikano, & Lang, 1987) (Lang & Hinton, 1988).

We propose in this paper a typical experiment of the capabilities of TDNNs with respect to DTW methods. We first describe the speech database we used. The time-delay architecture is then depicted in the third section. In the fourth, we present some experimental results, that were useful for tuning our systems. The fifth section contains the results of a comparison made between DTW and TDNN, using the same database, and the same preprocessing. Some hints for understanding how our TDNNs work internally are provided in the sixth section.

2. DIGIT RECOGNITION

A speech database, in French, has been elaborated at LIMSI. In the experiment reported here, we have only used part of the database, namely the utterances of the 10 digits by 26 speakers, male (40%) and female. Each of the speakers pronounced each digit once.

The signal has been processed in the following way, classically used at LIMSI (Gauvain, 1986; Singer, 1988): the speech signal, from the microphone in a quiet room, has been filtered at 5 KHz through a low-pass filter, then sampled at 10 KHz with a 12 bits A/D converter. High-frequency amplitudes are increased at 6 dB per octave. A DFT is applied on successive 25.6-ms time frames, overlapping by 12.8 ms. Thus 128 energy spectra values are generated in the 0–5-KHz frequency domain. A Bark scaled 16-channels filterbank is then simulated by averaging on triangular frequency windows. The energy spectra are then log-compressed.

This processing thus results in coding the speech signal into sixteen eight bits values per 12.8-ms time frame. The digits, in French, are all monosyllabic words, except "zéro". The ten digits used in this experiment lasted between 15 and 61 time frames (Figure 1).

The resulting database has been used for comparison of three different recognition techniques, namely Euclidian classifier, time-delay multilayer perceptrons and dynamic time warping.

The Euclidian classifier provides a baseline result for evaluating the database difficulty. Although such a classifier is not actually appropriate for speech recognition tasks, this reliable and simple measure of the database complexity may provide useful information to the reader.

3. TIME-DELAY MLP

3.1. Description

In the late 1950s, most of the perceptron architectures combined hand-crafted feature extractors followed by an adaptive layer. These feature extractors are first processed on the input data, producing some feature maps. The resulting information is then used both for teaching the adaptive layer and retrieving the output of the perceptron. Minsky and Papert (1969) showed the weakness of using a unique adaptive layer, and pointed out the resulting theoretical limitations of the perceptron architecture. However, adequately hand-crafted feature extractors usually make the task easier for the adaptive layer, but finding appropriate feature extractors may be too complex to be achieved by hand.

The now well-known gradient back propagation (GBP) rule allows to train multilayer perceptrons (MLPs) and to find near optimal internal representations, that is, MLPs are capable of learning a set of optimal feature extractors.

Let us now describe a network for learning feature extractors. Each feature extractor is built from a set of hidden units. These units are locally connected to a window scanning the input data. As stated above, the speech data have strong time-invariant properties. It thus seems interesting to give to our network some knowledge about these properties. The simplest way to do so is to insure that all the cells belonging to the same feature extractor will perform the same task with respect to their inputs.

We thus enforce, during the training phase, their incoming sets of weights to be identical. Such extensions of the standard back-propagation algorithm were discussed in the PDP book (Rumelhart, Hinton, & Williams, 1986). The general theme of constrained back-propagation has also been extensively studied in Le Cun (1988).

In the case of speech recognition, we use a small number of time-invariant feature extractors on the

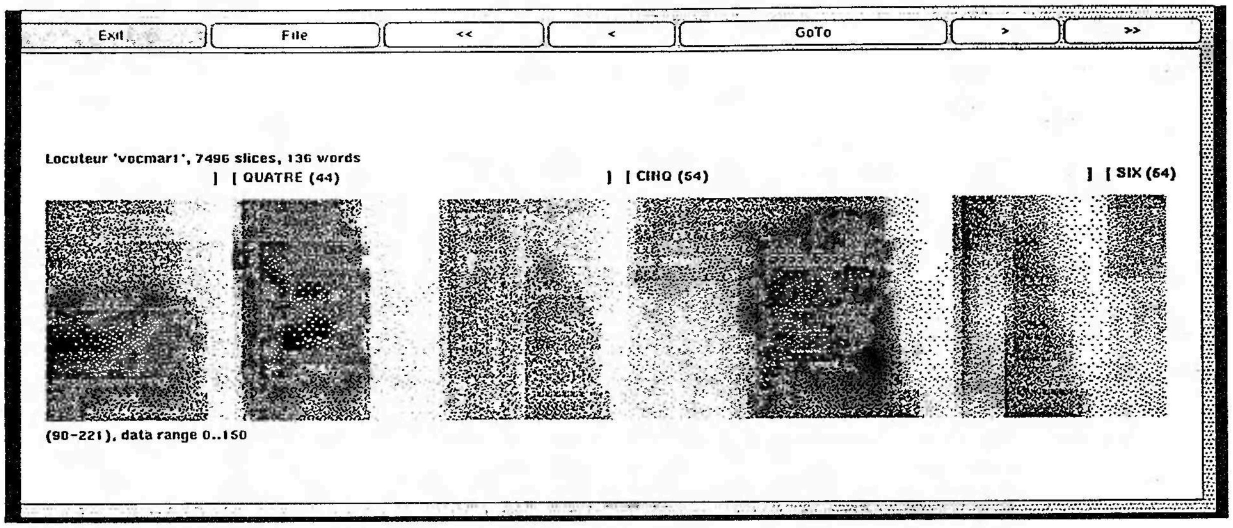

Exit
File
<<
<
GoTo
>
>>
Locuteur 'vocmar1', 7496 slices, 136 words
] [QUATRE (44)
] [CINQ (54)
] [SIX (64)
(90-221), data range 0..150

FIGURE 1. The figure shows some examples of French digits spectrograms, as computed by the LIMSI processor. Each word has been hand segmented (signs [and] on the figure).

input data. The resulting values are then used as input data for another layer of feature extractors and so on. The network is trained by constrained back-propagation.

As already related in Lang and Hinton (1988), there is a nomenclature problem: If the "feature extractors" and the "feature cells" were respectively called "cells" and "cell at a given time," then some connections would have been drawn between "cells at different times" and thus these connections would have been renamed "time-delayed-connections" (Waibel et al. 1987; Lang & Hinton, 1988). Both trends indeed describe the same mathematical object, either as a time-unfolded network, or as a time-delay neural network (TDNN). Both nomenclatures reveal different interesting aspects of the same object.

Experiments have been run (Lang & Hinton, 1988) to compare fully connected, locally connected, and TDNN networks. The experiments have been carried out on the /b/,/d/,/e/,/v/ task. The results show, even on this very simple task, that the time-delay trick is quite appropriate. Other experiments at ATR (Waibel et al., 1987) showed that such networks were capable of achieving better results than a hidden Markov model (HMM) on a Japanese /b/, /d/,/g/ recognition task.

A fundamental question concerning the use of such architectures for more complex speech tasks deals with the scaling problem: how do architectures, learning time, performances, scale with the complexity of the task? In order to explore this problem, we used the digit recognition task: In our database, the speech signal lasts about one second, which is the input to the network. This is to be compared with, for example, the /b/,/d/,/g/ problem (Waibel et al., 1987) where the typical speech data lasted 150 ms only: With our larger framing rate, this means an increase by 4 of the number of input units. We also reduced the number of cells in the hidden layers by progressively reducing the time discretization of the cells. However, our network is about 1300-cells large, where Waibel's was about 400. We now precisely describe our network architecture.

In our digit problem (Figure 2), we have one sixteen-dimensional vector as input every time slice (12.8 ms). A first layer of 8 feature extractors operating on windows of three consecutive vectors transforms these inputs into 1 eight-dimensional vector every two time-slices. A new layer of 8 feature extractors, windowed on seven consecutive vectors, give 1 eight-dimensional vector every ten time-slices. The resulting vectors are then fully connected to 10 decision cells, one for each digit to be recognized.

In a previous work (Waibel et al., 1987), hidden layers states were computed every time slice. We choose to sample these states every two time frames in the first hidden layer, then every ten time frames in the second hidden layer, by reducing the number of cells in these layers. The computational load is thus significantly reduced during both training and retrieval. Finally, in contrast with Waibel's work; the output layer is fully connected to the last hidden layer through adaptive weights: this allows for a last integration step, which is also learnt by the network.

Another argument was used to motivate these choices: The information at the spectrogram level is about 10000 bits/s. The phonetic information for the same data (i.e., the information associated with the corresponding sequence of phonemes) is no more than 50 bits/s. The information carried through our ten thresholded output cells is less than 6 bits/s. Our speech recognition network can thus be viewed as an information filtering device. If the last layer fea-

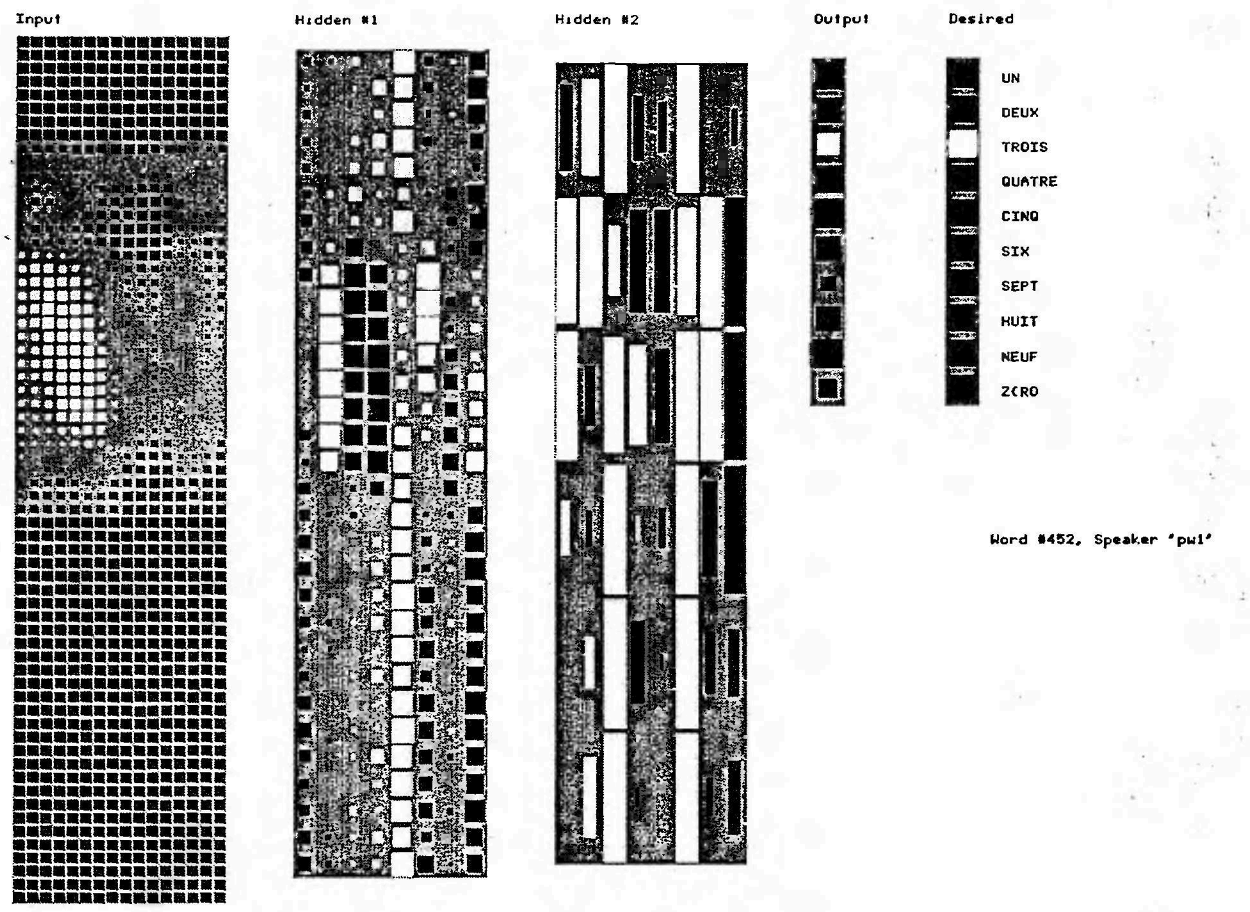


FIGURE 2. The multi-layer network used in our simulations. It has one input layer, with 16 frequency-channels and 65 time frames. The first hidden layer has 8 feature extractors: each one is connected to 3 consecutive time frames, in the entire frequency range in the input level. 31 successive cells implement a feature extractor through the time domain: they overlap by one time-frame. Cell no. 1, for example, is connected to time-frame 1 to 3 and cell no. 2 to time-frame 3 to 5. The second hidden layer has a similar structure: it has 8 feature extractors, connected to 7 consecutive sets of 8 cells in the previous layers, the cells in that layer are overlapping by 5: cell 1 is connected to the 8 feature extractors cells no. 1 to 7 and cell 2 to those numbered 6 to 12. The output level is fully connected to the last hidden layer.

ture extractors were computed every time frame, 625 values per second would have flowed through the last hidden layer. This layer would thus have to perform most of the information filtering. We preferred instead to distribute that task among all the network layers, rather than to rely on a single linear separator.

3.2. Learning strategy

It is necessary, during the training phase, to fix some input layer size. In all our experiments, this input layer is set with 65 time frames, which is slightly longer than the longest word we had to learn. In addition, each utterance is randomly shifted in the 128 first milliseconds of the 832 ms input window, simulating a poor word segmentation.

The network is trained using an all purpose back-propagation algorithm, exactly as described in (Fogelman Soulié, Gallinari, Le Cun, & Thiria, 1987; Le Cun, 1987).

- We use stochastic back-propagation, that is, Widrow Hoff rule, also called "online" back-propagation: The weights are updated after each pattern presentation. Stochastic back-propagation drastically reduces both the learning time in classification tasks and the number of local minima.
- We update the weights using neither momentum nor weight decay. Momentum does not seem really efficient when used with stochastic back-propagation.
- We add on-the-fly Gaussian noise to the inputs. Using this technique slightly improves the network ability to find a more general solution.
- The initial weights are carefully chosen: We ensure that the standard deviation of the weighted sum fit within the linear part of the activation function (a hyperbolic tangent scaled into the [—1.7, 1.7] range).
- The equality constraint is implemented using shared weights. They are updated by multiplying

the learning rate by the sum of the corresponding partial derivatives. In these experiments, the thresholds are not constrained.

- We do not try to further optimize the learning speed. We started with reasonable values for the learning rate (respectively, 0.01, 0.02, and 0.03 in the three weight layers) and standard deviation of the Gaussian noise (0.1), and manually divided them by two as soon as the mean squared error stopped decreasing (at the 5th, 9th, 14th, and 24th sweep, cf. Figure 3). This is a quite robust process for controlling the learning phase, which was largely eased by the use of our graphics simulator (Bottou, 1988).

4. EXPLORATION RESULTS

Before attempting some systematic experiments, we found it useful to first evaluate the different recognition techniques on the same test task. We thus defined a 16 speakers learning set, both male and female. The remaining ten speakers were used as a test set. Speakers were assigned to the two sets using alphabetical order, thus independently of any phonetical clue.

This 10 speakers test set is obviously too small for precisely comparing the three recognition systems. However, strong differences in the recognition scores would have indicated that one or the other was ap-

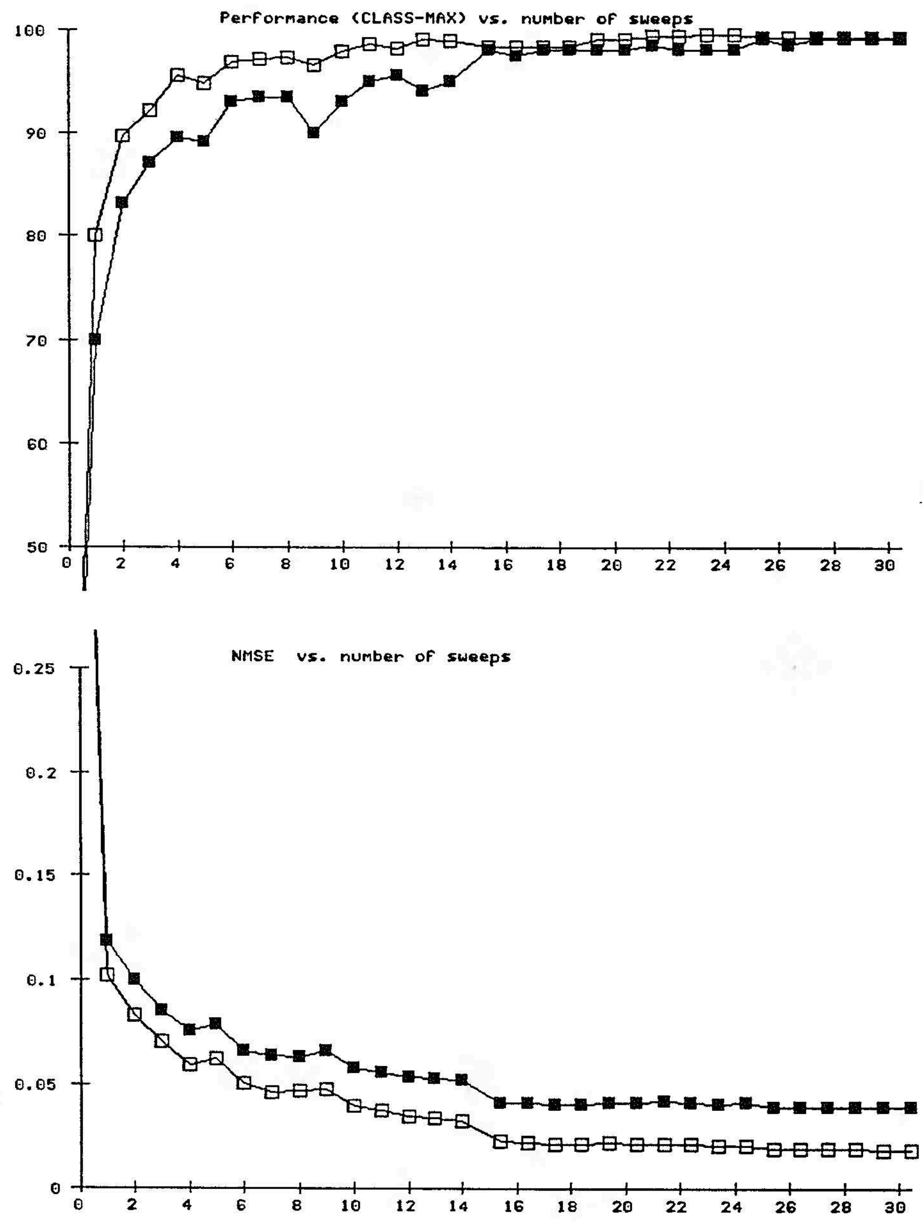


FIGURE 3. The evolution, during training, of the performances both on the training and test sets. The x-axis shows the number of epochs.

proach was definitely unable to compete with the others.

4.1. Performance of a Euclidian classifier

We first used a simple Euclidian classifier on 65 time frames long patterns.

We built 640 training patterns out of the 160 utterances by shifting them in four random positions within the first 128 ms. In the same way, with only two random shifts, we built 200 patterns out of the 100 test utterances.

Class means were computed by averaging all the training set patterns. We attempted then to classify the test set patterns by measuring the Euclidian distance to the class mean. This rather rude technique performs poorly: 71.5% only of the test patterns were correctly classified.

4.2. Network results

In a preliminary experiment, we trained the network without randomly shifting the training data in the input window. We easily got a perfect recognition of the training set, and a peak 95% performance on the test set.

Unfortunately, although the hidden layers of our network are designed for shift invariance, *the last weight layer is not.* We then tested the previous weight configuration on the same randomly shifted patterns that were used for the Euclidian classifier. When the test utterances were randomly shifted in the 0–51-ms range, the performance sank to 89%. With even more shifted data, in the 0–102-ms range, the network achieved a poor 70% performance.

In order to get a real shift invariance capacity, we then decided to train the network with the randomly shifted data described above (shifted by 0–128 ms). Moreover, such a random shift simulates a poor word segmentation, and is in fact much closer to real speech tasks than perfectly aligned test and training data.

These small experiments clearly show that the decision layer is the weak point of time-delay networks. In these experiments, the layer is a simple linear separator. Experiments reported by both Lang (Lang & Hinton, 1988) and Waibel (Waibel et al., 1987) rely on a slightly more complicated integration layer, which is more adequate for phoneme processing.

We then ran this learning task on the shifted data a couple of times during the first week of August 1988. We always stopped the simulation after 30 sweeps of the 640 training patterns (i.e., 19,200 single pattern presentations). The network never achieved less than 98% correct answers on the training set and 94% on the test set. One sweep of the 640 training patterns plus test cycles on both the training and test sets took about three minutes on a Sun 4 workstation running our general purpose back-propagation simulator SN. (Bottou & Le Cun, 1988).

The best run produced a network able to correctly classify 99.21% of the training patterns and 99% of the test patterns (i.e., 1 *only unrecognized utterance out of the* 100 *test utterances*). Unfortunately, our speech database is clearly too small for really validating such performances. However, we reproduced a couple of times this result with different initial weights. It is interesting to notice that after 6 sweeps, the network already achieved 97.9% on the training set and 93% on the test set, and after 15 sweeps, 98.3% and 98% (see Figure 3a and 3b).

These results also show how a small number of epochs is sufficient for efficiently teaching a time-delay network, without problem specific or machine dependent tricks.

4.3. DTW results

We first compared these results with an existing LIMSI's time warping system, usually performing best on cepstral coefficients computed as follows:

Each time frame of the spectrogram log-compressed energies is first normalized. A cosine transformation is then applied for extracting 8 cepstral coefficients. The time warping process is performed using these 8 parameters and the mean energy. The sixteen learning utterances for each word are precisely segmented, then averaged along their time warping path, using an algorithm described in (Singer, 1988). The resulting references require approximately as much memory space as the MLP weights.

The 100 test utterances have been presented and 99 were correctly classified. The error source was identified as a badly segmented training reference. We thus conclude that the DTW system achieved a 99% performance, similar to the network.

5. MORE EXPERIMENTS

The above experiments actually show that a network is able to perform correctly on a speech recognition

TABLE 1

Set	DTW	MLP (Shifted Data)	MLP (Nonshifted Data)
A	1	11	14
B	3	9.25	9
C	6	8.25	4
D	2	7	4
Total (out of 640 utterances)	12 (1.9%)	35.5 (5.5%)	31 (4.8%)

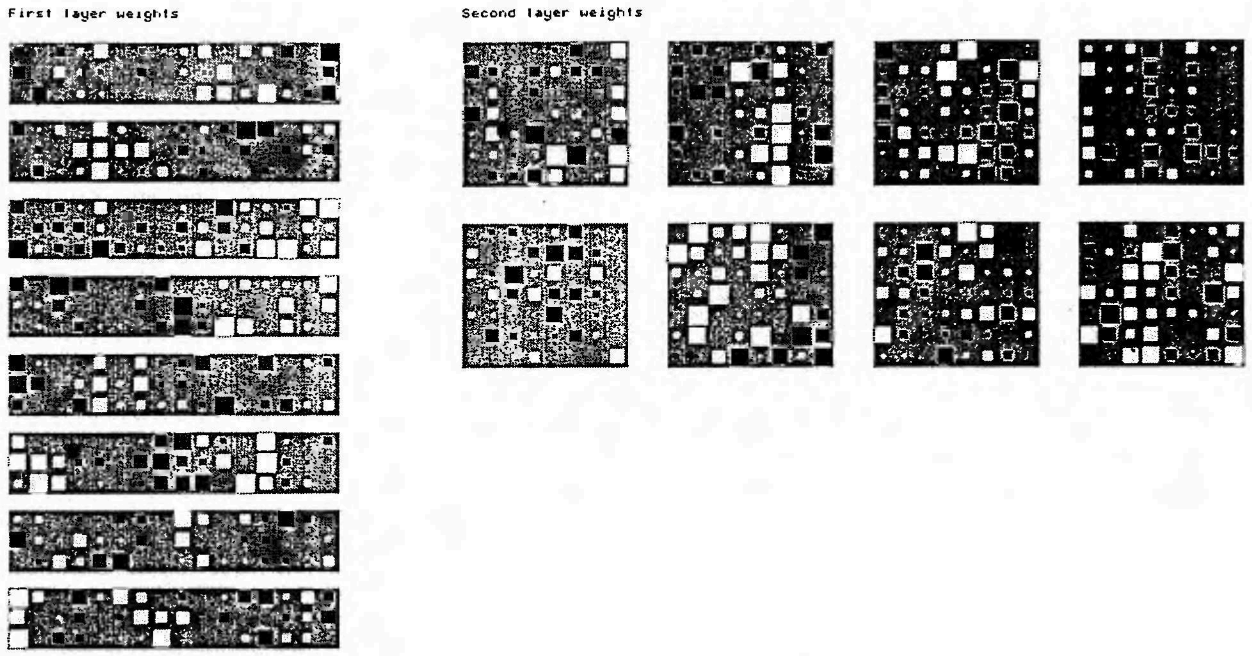


FIGURE 4. Weights of the first and second hidden layers, that is, the feature detectors or masks computed by the network. The last mask in the first hidden layer, for example, is implementing the detection of a climbing second formant in the frequency domain. Weights in the second hidden layer implement combinations of detectors and are thus almost impossible to interpret.

task. However, the test set is so small that no comparison can be significantly achieved with these results. Moreover, the DTW system uses a quite different preprocessing, and is thus hard to truly compare.

We thus decided to lower the performances by using less speakers for training. Moreover, a modified DTW system, working on spectral data, was just released at LIMSI. We agreed on four 10 speakers training sets, and four 16 speakers corresponding test sets with males and females in the same proportions.

We also improved the consistency of the shifts applied to the utterances for generating the MLP patterns. During training, the utterances are randomly shifted on-the-fly within the first 128 ms. Tests were performed on both randomly shifted and fixed utterances.

Table 1 shows the number of errors made on each test set, by

- the DTW system,
- the MLP, on randomly shifted test utterances.

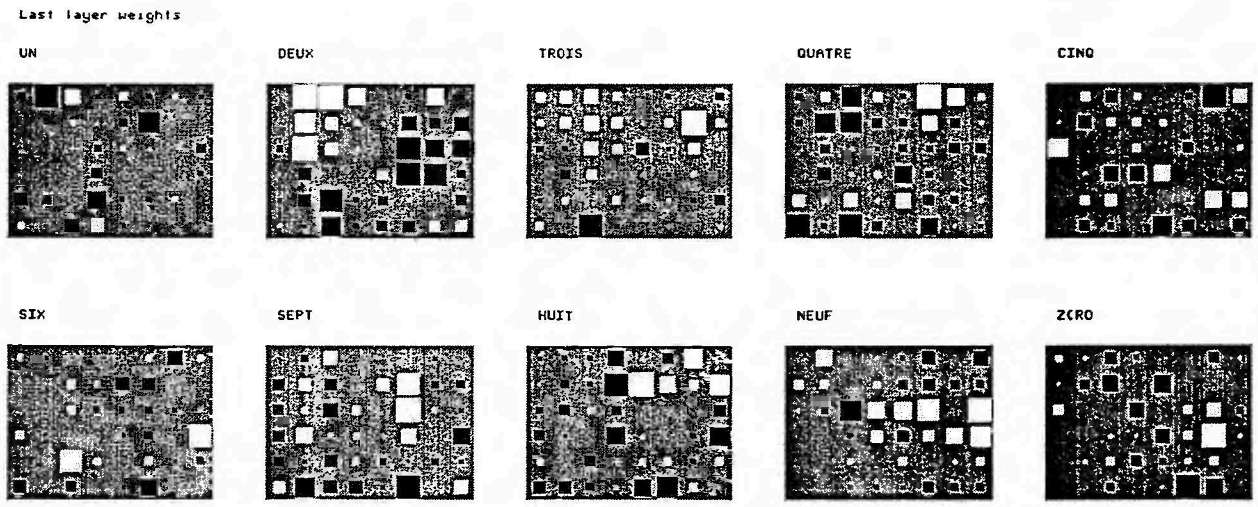


FIGURE 5. Weights of the second hidden to output layer. This represents the way by which the network weights the codes computed in the last hidden layer for each different digit.

Each utterance is tested four times, in four different positions. The "numbers of errors" thus are in fact averages over these four tests,

- by the MLP, on utterances starting 64 ms after the beginning of the input window. The network has still been trained on randomly shifted data

In these experiments, the network clearly did not achieve the same level of performance as the DTW based system. Testing against shifted or unshifted utterances does not sensibly modify the performance level.

Reducing the number of training utterances has drastically degraded the network performances, while the DTW results remain good. These results, however are slightly compensated by the following remarks:

- The MLP learns better on poorly segmented utterances, while the DTW requires a good knowledge of the starting and end-points of each reference.
- The errors made by the MLP and the DTW are rather different: Set "A", for example, is the best reference set for the DTW, and the worst training set for the MLP. A hundred training utterances is possibly not enough for training the MLP to resist time distorsions. This invariance is built in the DTW process! If this interpretation is correct, the MLP performances could be improved by using artificially distorted training patterns, as in Bottou (1988).

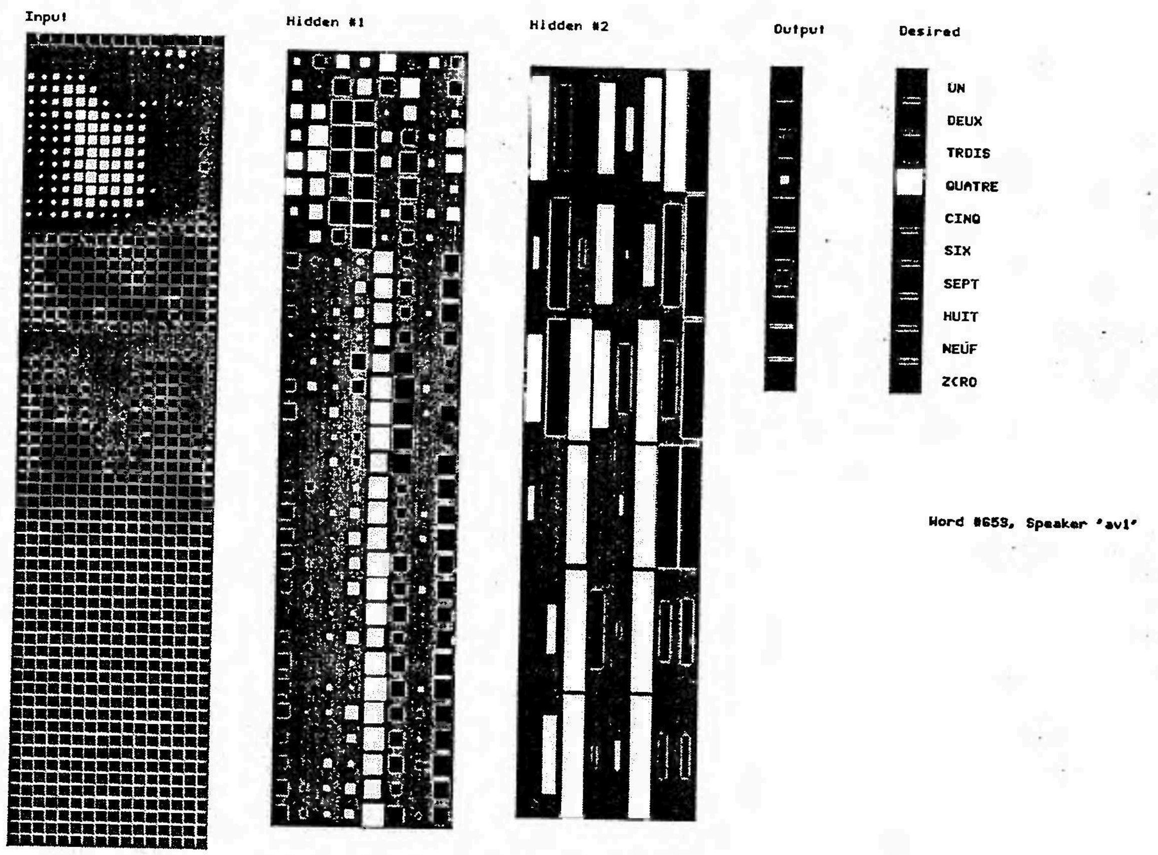
Nonetheless, the DTW requires less utterances than the MLP for achieving a given level of performances, proving therefore, that classical systems actually are not obsolete.

6. INTERNAL NETWORK SPEECH REPRESENTATION

Understanding the nature of the internal network representations may be the best way to improve its performances. It seems reasonable to study these

00	UN	1	1	4	8	8	8	50	SIX	6	6	6	6	6	8
100	UN	3	1	4	8	8	8	150	SIX	6	5	6	8	8	8
200	UN	1	1	4	8	8	8	250	SIX	6	6	6	6	6	8
300	UN	1	1	4	8	8	8	350	SIX	6	6	6	6	6	6
400	UN	3	1	4	8	8	8	450	SIX	6	6	6	6	6	8
500	UN	1	1	4	8	8	8	550	SIX	6	5	6	8	8	8
600	UN	1	1	4	8	8	8	650	SIX	6	6	6	6	6	6
700	UN	3	1	1	4	8	8	750	SIX	6	6	6	6	3	4
800	UN	1	1	4	8	8	8	850	SIX	6	6	6	6	6	8
900	UN	1	1	4	8	8	8	950	SIX	6	6	5	6	6	8
10	DEUX	7	2	2	8	8	8	60	SEPT	6	7	7	8	8	8
110	DEUX	7	2	2	8	8	8	160	SEPT	6	7	2	8	8	8
210	DEUX	7	2	2	8	8	8	260	SEPT	6	7	2	8	6	8
310	DEUX	7	2	2	8	8	8	360	SEPT	6	7	2	8	8	8
410	DEUX	7	5	5	2	8	8	460	SEPT	6	7	2	8	8	8
510	DEUX	7	2	2	8	8	8	560	SEPT	6	2	4	8	8	8
610	DEUX	7	7	2	2	8	8	660	SEPT	6	7	2	8	8	8
710	DEUX	7	2	2	2	8	8	760	SEPT	6	6	7	8	3	4
810	DEUX	7	5	5	8	8	8	860	SEPT	6	6	2	7	8	8
910	DEUX	7	2	2	8	8	8	960	SEPT	6	7	2	8	6	8
20	TROIS	7	3	4	4	8	8	70	HUIT	5	6	4	8	8	8
120	TROIS	8	3	3	4	8	8	170	HUIT	5	6	4	8	8	8
220	TROIS	3	3	4	8	8	8	270	HUIT	5	6	4	8	8	8
320	TROIS	7	3	3	4	8	8	370	HUIT	5	5	6	8	6	8
420	TROIS	7	3	4	8	8	8	470	HUIT	7	5	6	8	8	8
520	TROIS	7	3	3	8	8	8	570	HUIT	5	5	4	8	8	8
620	TROIS	3	3	3	4	8	8	670	HUIT	5	5	5	8	6	8
720	TROIS	8	3	3	4	8	8	770	HUIT	5	5	8	3	8	8
820	TROIS	3	3	4	8	8	8	870	HUIT	5	6	4	8	6	8
920	TROIS	3	3	4	8	8	8	970	HUIT	5	5	4	8	8	8
30	QUATRE	2	1	4	4	8	8	80	NEUF	7	2	1	8	8	8
130	QUATRE	7	4	4	4	8	8	180	NEUF	2	1	6	8	8	8
230	QUATRE	7	1	4	4	8	8	280	NEUF	7	2	2	8	8	8
330	QUATRE	7	1	4	4	8	8	380	NEUF	2	1	6	8	8	8
430	QUATRE	7	1	4	8	8	8	480	NEUF	5	1	4	8	8	8
530	QUATRE	2	4	4	8	8	8	580	NEUF	5	2	1	8	8	8
630	QUATRE	7	1	4	4	4	4	680	NEUF	5	1	2	6	6	8
730	QUATRE	7	4	4	4	4	8	780	NEUF	5	1	6	8	3	8
830	QUATRE	7	1	4	4	8	8	880	NEUF	5	2	2	6	6	8
930	QUATRE	7	1	4	4	8	8	980	NEUF	2	1	4	8	8	8
40	CINQ	6	3	2	8	6	8	90	ZERO	5	2	3	8	8	8
140	CINQ	6	7	1	7	6	8	190	ZERO	5	2	3	3	8	8
240	CINQ	6	8	1	2	6	8	290	ZERO	6	5	3	8	8	8
340	CINQ	6	7	1	1	8	8	390	ZERO	7	2	3	3	3	8
440	CINQ	6	7	1	2	6	8	490	ZERO	7	5	2	3	8	8
540	CINQ	6	3	2	8	8	8	590	ZERO	7	5	2	3	8	8
640	CINQ	6	6	1	2	7	8	690	ZERO	7	5	2	3	3	8
740	CINQ	6	3	1	8	7	8	790	ZERO	5	2	3	3	8	8
840	CINQ	6	1	2	8	6	8	890	ZERO	7	5	2	3	3	8
940	CINQ	6	3	1	8	6	8	990	ZERO	7	2	2	3	8	8

FIGURE 8. The result of the clustering algorithm for 8 clusters. Each word is coded by a sequence of 6 numbers, which give the number of the cluster the corresponding activity code was in. For example, 8 is clearly the prototype of the code for silence. Different utterances of each digit are coded by different sequences, but the coding is still rather stable, for example, "deux" is coded ..728... or ..758..., "quatre" as ..714... or ..214... or ...74... or ...24....



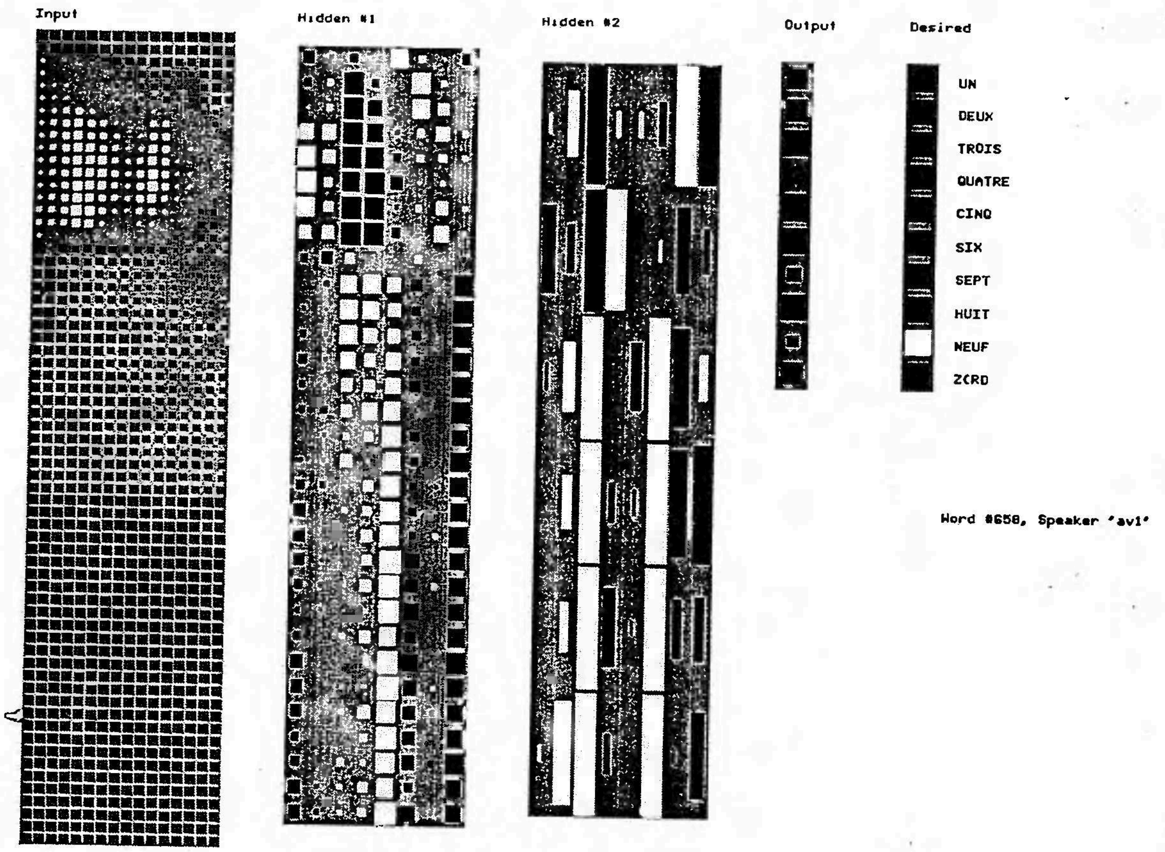


FIGURE 6. The error made by the network in test phase. It classifies the 9 "neuf" shown on Figure 6a as a 4 "quatre," shown on Figure 6b, pronounced by the same speaker "av1." The figures represent the activities of the cells in the different layers, from input (left) to output (right). The desired output is shown in the last column.

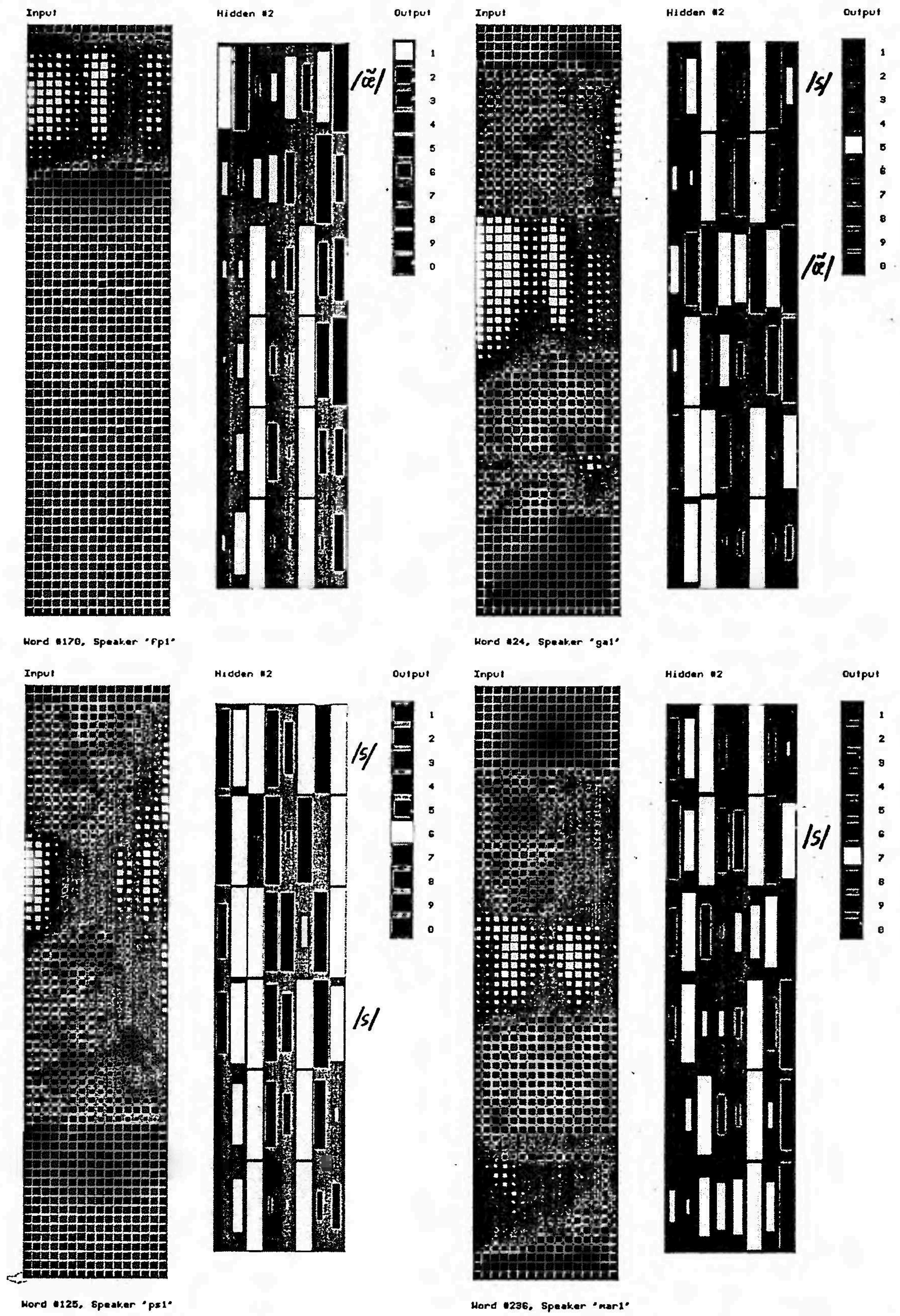


FIGURE 7. The figure shows the results obtained for the four digits: 1 "un," 5 "cinq," 6 "six" and 7 "sept" pronounced by different speakers. The phonemes /s/ and /œ̃/ (in "un" and "cinq") have been extracted by the network and coded in the last hidden layer activity: For example, in "six," phoneme /s/ is coded 01100101 (from left to right and in digital form) and detected in the first (first occurrence of /s/ in "six") and fourth (second occurrence of /s/ in "six") set of cells. That same code is used by the network to detect phoneme /s/ in "cinq" and "sept" in the first set of cells. The phoneme /œ̃/ is coded 10011010 both in "cinq" and "un." (Note, only the input, the second hidden layer and the output are shown here.)

representations on our best MLP. We thus focus in this section on the weights configuration generated by the network described in the fourth section.

The unique error made by this network actually is not acceptable. In French, the digits 5 (cinq) and 7 (sept) are pronounced in a similar way, and French digits recognizers classically might mix them. In our networks, we traced by hand the cells activities for the unique error. A 9 (neuf) had been recognized as a 4 (quatre), which is rather disappointing at first sight (Figure 6).

We attempted to visually identify significant information in the first layer weights (Figures 4 and 5). For example, it seems easy to recognize formant's movements detectors or other phonetic features detectors. But we are unable to really distinguish between the randomly produced ones and those that the network effectively uses for classifying the patterns.

However, it is often claimed that neural networks are good at extracting meaningful internal states. In order to test for this ability here, we have studied the activities of the last hidden layer cells: there are 6×8 such cells, whose activities can be viewed as 6 vectors in an eight-dimension space (Figure 7). Each of these vectors represents the activity of the 8 masks in 6 different portions of the signal: Figure 7 shows that those vectors provide some sort of encoding of the digit into phonetic features similar to phonemes. This can be further tested as follows: The set of the activity vectors for all the available patterns (training and test sets) is analyzed through a k-means clustering technique. The result is a description of each digit as a sequence of subword units. In Figure 8, for example, we show the results of a clustering of the 600 first patterns (100 words) in 8 clusters: The network has clearly extracted a rather stable decomposition of each digit in subword units, of a slightly larger grain than phonemes. Clustering in 12 and 16 clusters does not yield better results, the derived coding is simply more redundant: The network does not seem to require more than 8 features to make its decision.

Moreover, this clustering technique also shows that the decomposition of the 9 that the network has extracted is the least stable. This remark could give some insight to start understanding the origin of the 9 versus 4 error: In addition, the stop-consonant in that 4 is quite short and weak, which means that the network could not use the stop-consonant information for disambiguation. It seems that our network structure exceedingly discretizes time inside the last hidden layer: Cells with finer grain should be used to allow for the detection of very short events.

Of course, we do not claim that the results obtained so far actually achieve the extraction of subword units which are completely understandable and meaningful. However, we think that following up this line of investigation could provide significant results in that area.

7. CONCLUSION

We have presented here a time-delay neural network trained on a French digit recognition task. This work clearly demonstrates that adequate neural nets have many significant abilities for speaker-independent speech recognition tasks. However our networks did not achieve as good results as a well-tuned classical system. We think that this is rather normal for the neural net approach, which has not been yet fully explored nor optimized.

Moreover, such connectionist adaptive systems may have interesting properties, that would bring much to today's speech recognition systems. The MLP has shown its ability to learn on badly segmented references, which suggests that a phonetical speech recognition system could be trained with affordable coarsely segmented databases. We also have given some hints to utilize MLPs for subword units extraction. More investigation is needed to get significant results. However, we think that the results obtained so far are an incentive to go further in the direction presented here.

Finally, we would like to comment on the learning and retrieving computational loads. Training a MLP takes longer than storing references in a time-warping system. However, training our system was much faster than the four days on a supercomputer reported in the original Waibel's paper. Recently, this research team reported a speed-up by a factor of 1,000 of their learning speed (Haffner, Waibel, & Shikano, 1988), which is more consistent with our own results. The retrieving load is about 20,000 multiply-adds per second, which is rather small in comparison with the preprocessing Fourier transform and is much faster than the computation required by the time-warping process.

REFERENCES

Bottou, L. (1988). Reconnaissance de la parole par réseaux multicouches. In *Proceedings of Neuro-Nîmes* 88, **EC2** (Ed), 371–382.

Bottou, L, & Le Cun, Y. (1988). SN: Un simulateur pour réseaux connexionnistes. In *Neuro-Nîmes* 88, **EC2**, 197–218.

Bourlard, H., & Wellekens, C. J. (1988). Links between Markov models and multilayer perceptrons. In D. S. Touretzky (Ed.), *Advances in Neural Information Processing Systemes*. Denver: Morgan Kaufman (Publisher).

Bridle, J. S., & Moore, R. K. (1984). Boltzmann machines for speech pattern processing. *Proceedings of the Institute of Acoustics, Autumn Meeting* 1984.

Elman, J. L., & Zipser, D. (1987). *Learning the hidden structure of speech* (*Technical Report*). San Diego: University of California.

Fogelman Soulié, F., Gallinari, P., Le Cun, Y., & Thiria, S. (1987).

Evaluation of network architectures on test learning tasks. *IEEE First International Conference on Neural Networks* (IEEE Catalog No. 87TH0191-7. Vol. II, pp. 653–600). San Diego, June 1987.

Gauvain, J. L. (1986). A syllable-based isolated word recognition experiment. In *Proceedings IEEE Conference on Acoustic, Speech and Signal Processing* 1986.

Gauvain, J. L., Mariani, J., & Liénard, J. S. (1983). On the use of time compression for word-based speech recognition. *Proceedings IEEE Conference on Acoustic, Speech and Signal Processing* 83.

Haffner, P., Waibel, A., & Shikano K. (1988). Fast back-propagation methods for neural networks. In *Speech.* Proceedings from the Fall meeting of the Acoustical Society of Japan.

Kohonen, T. (1988). The "neural" phonetic typewriter. *IEEE Computer*, pp. 11–22, March 1988.

Lang, K., & Hinton, G. E. (1988). *The development of TDNN architecture for speech recognition* (Technical Report CMU-CS-88-152). Carnegie Mellon University.

Le Cun, Y. (1987). *Modèles connexionnistes de l'apprentissage.* Thèse, Paris. Université de Paris VI.

Le Cun, Y. (1988). A theoretical framework for back-propagation. In D. S. Touretzky (Ed.), *Connectionist models: A summer school.* Denver: Morgan-Kaufmann.

Levinson, S. E. (1985). Structural methods in automatic speech recognition. *Proceedings of the IEEE*, Vol. 73 No. 11.

Liénard, J. S. (1977). *Les processus de la communication parlée.* Masson.

Lippmann, R. P., & Gold, B. (1987). Neural-net classifiers useful for speech recognition. *IEEE First International Conference on neural networks* (IEEE Catalog No. 87TH0191-7, IV, pp. 417–426). San Diego.

Lubensky, D. (1988). Learning spectral-temporal dependencies using connectionist networks. *Proceedings IEEE Conference on Acoustic, Speech and Signal Processing* 1988 (S-Vol. 1, pp. 418–421), New York.

Minsky, M., & Papert, S. (1969). *Perceptrons.* Cambridge, MA: MIT Press. (Expanded edition in 1988).

Prager, R. W., Harrison, T. D., & Fallside, F. (1986). Boltzmann machines for speech recognition. *Computer, speech and language*, **1**, 3–27.

Quenot, G., Gauvain, J. L., Gangolf, J. J., & Mariani, J. (1989). A dynamic programming processor for speech recognition. *IEEE Jl of Solid State Circuit*, **24** (2).

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland (Eds.), *Parallel distributed processing* (Vol. 1, pp. 318–362). Cambridge, MA: MIT Press.

Singer, H. (1988). Utilisation de dissylabes pour la reconnaissance de la parole. *Rapport LIMSI*, 1988–4.

Waibel, A., Hanazawa, T., Hinton, G. E., Shikano, K., & Lang, K. (1988). Phoneme recognition: Neural networks vs. hidden Markov models. *Proceedings IEEE Conference on Acoustic, Speech and Signal Processing* 1988 (S-Vol. 1, pp. 107–110), New York.

Waibel, A., Hanazawa, T., Hinton, G. E., Shikano, K., & Lang, K. (1987). *Phoneme recognition using time-delay neural networks.* Preprint, ATR Interpreting Telephony Research Laboratories.

Watrous, R. L., & Shastri, L. (1987). Learning phonetic features using connectionist networks. *IEEE First International Conference on Neural Networks* (IEEE Catalog No. 87TH0191-7, IV, pp. 381–388). San Diego.

APPENDIX: ERROR LISTS AND CONFUSION MATRICES

We present here the list of errors and the confusion matrices for both the MLP and the DTW system, for the ten training speakers problem. The errors are listed for each set as follows:

(Digit.Speaker)→(Wrong class)

Each line of the confusion matrices shows here each digit has been recognized.

MLP ERROR LIST

SET A	SET B	SET C	SET D
(5.heb1)→(7)	(5.heb1)→(7)	(9.av1)→(4)	(5.cb1)→(7)
(2.lac1)→(0)	(5.lac1)→(0)	(3.tas1)→(4)	(2.fb1)→(7)
(3.lac1)→(0)	(9.pha1)→(2)	(5.tas1)→(1)	(5.fp1)→(9)
(7.lac1)→(5)	(7.pr1)→(2)	(6.tas1)→(8)	(5.jjg1)→(7)
(8.lac1)→(6)	(9.pw1)→(2)	(7.tas1)→(4)	
(5.mar1)→(7)	(3.tas1)→(4)		
(9.pr1)→(2)	(5.tas1)→(1)		
(9.pw1)→(2)	(6.tas1)→(8)		
(3.tas1)→(4)	(7.tas1)→(9)		
(5.tas1)→(1)			
(6.tas1)→(8)			
(7.tas1)→(4)			

APPENDIX: ERROR LISTS AND CONFUSION MATRICES (Continued)

MLP CONFUSION MATRIX (1.56% = 1 error)

	recognized as									
	1	2	3	4	5	6	7	8	9	0
d 1	100.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
i 2	0.00	96.88	0.00	0.00	0.00	0.00	1.56	0.00	0.00	1.56
g 3	0.00	0.00	93.75	4.69	0.00	0.00	0.00	0.00	0.00	1.56
i 4	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
t 5	4.69	0.00	0.00	0.00	84.38	0.00	7.81	0.00	1.56	1.56
6	0.00	0.00	0.00	0.00	0.00	95.31	0.00	4.69	0.00	0.00
7	0.00	1.56	0.00	3.12	1.56	0.00	92.19	0.00	1.56	0.00
8	0.00	0.00	0.00	0.00	0.00	1.56	0.00	98.44	0.00	0.00
9	0.00	6.25	0.00	1.56	0.00	0.00	0.00	0.00	92.19	0.00
0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00

DTW ERROR LIST

SET A	SET B	SET C	SET D
(6.tas1)→(8)	(1.bg1)→(7)	(1.bg1)→(7)	(1.bg1)→(7)
	(5.tas1)→(1)	(5.bg1)→(7)	(9.ga1)→(2)
	(6.tas1)→(8)	(9.ga1)→(2)	
		(5.tas1)→(1)	
		(6.tas1)→(8)	
		(7.tas1)→(4)	

DTW CONFUSION MATRIX

	recognized as									
	1	2	3	4	5	6	7	8	9	0
d 1	95.31	0.00	0.00	0.00	0.00	0.00	4.69	0.00	0.00	0.00
i 2	0.00	100.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
g 3	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
i 4	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
t 5	3,13	0.00	0.00	0.00	95.31	0.00	1.56	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00	0.00	95.31	0.00	4.69	0.00	0.00
7	0.00	0.00	0.00	1.56	0.00	0.00	98.44	0.00	0.00	0.00
8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00	0.00	0.00
9	0.00	3.13	0.00	0.00	0.00	0.00	0.00	0.00	96.87	0.00
0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.0