COMMUNICATIONS RESEARCH CENTER

CENTRE DE RECHERCHES SUR LES COMMUNICATIONS

Neural Network Classifier Architectures for Phoneme Recognition

CRC Technical Note No. CRC-TN-92-001

Communications Canada

Canada

COMMUNICATIONS RESEARCH CENTRE
LE CENTRE DE RECHERCHES SUR LES COMMUNICATIONS

BROADCAST TECHNOLOGIES RESEARCH
LA RECHERCHE EN TECHNOLOGIES DE RADIODIFFUSION

Releasable

Neural Network Classifier Architectures for Phoneme Recognition

CRC Technical Note No. CRC-TN-92-001

by

William Treurniet

Broadcast Technologies Research Branch
Communications Research Centre
Department of Communications

Ottawa, March 26, 1992

Government of Canada
Department of Communications

Gouvernement du Canada
Ministère des Communications

Approved for issue as a CRC Technical Note by : [illegible]

Canada

Communications Canada

Available from:

Communications Research Centre
3701 Carling Ave.
P.O. Box 11490
Station H
Ottawa, Ontario
K2H 8S2
Canada

Disponible de:

Centre de recherches sur les communications
3701, avenue Carling
Case postale 11490, Succursale H
Ottawa (Ontario)
K2H 8S2
Canada

Abstract

Automatic recognition of words in continuous speech is difficult to do with whole-word template models, especially when the vocabulary is reasonably large. Instead, the currently preferred approaches for this task hypothesize the presence of words or sub-words, such as syllables or phonemes, on the basis of likelihood estimates obtained from comparisons of the acoustic data with statistical models derived from training data. This paper is concerned with the application of artificial neural networks, trained with the back-propagation learning algorithm, to modelling phonemes extracted from the DARPA TIMIT multi-speaker, continuous speech data base.

A number of proposed network architectures may be applied to the phoneme classification task. These range from the simple feedforward multilayer network to more complex modular architectures which attempt to assign classifier modules to different regions of the input space. This paper describes experiments exploring the utility of some of these architectures including Hampshire and Waibel's Meta-Pi network, Nowlan and Hinton's system of experts, and Smieja's Minos architecture.

In general, modular architectures that attempt to identify regions of input space could not perform any better than a single network trained to handle the whole input space. The failure to find a performance improvement might have been due to excessive overlaps in class distributions in the training data. These would have hindered unambiguous identification of region boundaries in the input space by the module dedicated to that task. Given highly overlapping distributions, the simple network structure may already have performed as well as could be expected. Also, the self-organizing modular architectures may have had insufficient training data to counteract potentially large imbalances in class frequencies in the regions of input space selected by individual modules.

Two network structures learned to classify to some degree the total phoneme space: a single multilayer network, trained to discriminate among 39 classes, achieved 47.7 percent correct generalization performance, and a novel architecture trained to discriminate among 38 classes, achieved an accuracy of 52.4 percent. The latter employed a number of TDNN modules to map the input space to a different representation of reduced dimensionality which was used by a multilayer network to discriminate among the classes. When the relatively high frequency *silence* class was ignored in generalization tests, accuracies of 48.2 percent and 45.8 percent were obtained from the multilayer network and the modular network, respectively. Thus, the modular network performed somewhat better by making fewer errors identifying the *silence* class.

Résumé

Il est difficile de réaliser la reconnaissance automatique des mots dans le discours continu à partir des modèles à gabarits de mots complets, surtout quand le vocabulaire est relativement étendu. Les approches favorisées pour l'instant posent plutôt comme hypothèse la présence de mots ou de sous-mots, par exemple des syllabes ou des phonèmes, sur la base d'estimations de la probabilité obtenues par comparaison entre les données acoustiques et les modèles statistiques produits à partir des données de formation. Ce document traite de l'application de réseaux neuronaux artificiels, ayant suivi une formation par l'algorithme d'apprentissage par propagation amont, à la modélisation de phonèmes extraits de la base de données du discours continu multilocuteur DARPA TIMIT.

Un certain nombre d'architectures réseau proposées peuvent être appliquées à la classification des phonèmes. Cette plage d'architectures s'étend du simple réseau multicouche à alimentation aval aux architectures modulaires plus complexes, qui tentent d'assigner des modules classificateurs aux diverses régions de l'espace d'entrée. Le document décrit des expériences explorant l'utilité de certaines de ces architectures, y compris le réseau Méta-Pi de Hampshire et Waibel, le système d'experts de Nowlan et Hinton, et l'architecture Minos de Smieja.

En général, la performance des architectures modulaires qui tentent d'identifier des régions de l'espace d'entrée n'était pas meilleure que celle d'un réseau unique formé à la gestion de l'ensemble de l'espace d'entrée. L'échec des tentatives d'amélioration de la performance peut avoir été dû à un chevauchement excessif dans les distributions de classes des données de formation, ce qui aurait nui à l'identification non ambiguë des limites des régions dans l'espace d'entrée par le module affecté à cette tâche. Il est possible qu'en raison du fort chevauchement des distributions, on ait déjà atteint la performance maximale de la structure de réseau simple. De plus, les architectures modulaires à auto-organisation peuvent avoir été utilisées avec des données de formation qui ne permettent pas de compenser les déséquilibres potentiellement importants des fréquences des classes dans les régions de l'espace d'entrée choisies par les modules individuels.

Deux structures de réseau ont appris à classifier dans une certaine mesure l'espace de phonèmes total: un réseau multicouche unique, formé à la discrimination parmi 39 classes, a obtenu une exactitude de généralisation de 47,7 pour cent, tandis qu'une architecture nouvelle formée à la discrimination parmi 38 classes, a obtenu une exactitude de 52,4 pour cent. La deuxième architecture a utilisé un certain nombre de modules TDDN pour établir des correspondances entre l'espace d'entrée et une représentation différente à dimensionnalité réduite, utilisée par un réseau multicouche pour la discrimination entre les classes. Lorsque la classe *silence* de fréquence relativement élevée était ignorée dans les tests de généralisation, on obtenait des exactitudes de 48,2 pour cent et de 45,8 pour cent respectivement pour le réseau multicouche et pour le réseau modulaire. Par conséquent, la performance du réseau modulaire a été quelque peu améliorée grâce à la réduction du nombre d'erreurs lors de l'identification de la classe *silence*.

Contents

List of Figures

List of Tables

1 Introduction

"In time, speech will become the primary means of making ourselves understood to computers, and vice versa. Progress will accelerate dramatically as computing devices grow smaller, making voice the only means of communicating with them."[1]

The above quote points towards the emergence of a new mode of interaction between computer systems and their users. Descriptions of people conversing with computer systems are easily found in fiction, and the contexts are often quite realistic scenarios[2]. However, some difficult technical problems need to be solved before people can converse as naturally with computers as they do with each other. These include machine recognition of speech units in continuous speech spoken by any speaker, and machine understanding of the meaning of the spoken words.

Automatic recognition of continuous speech from any speaker is a hard problem to solve by computers due to the variability in acoustic representations of utterances of the same word. However, the literature shows that a steady improvement in performance of laboratory systems has occurred since about 1975 [20]. Part of this improvement was due to better modelling of speech sounds [21], but further remarkable improvement has recently been shown using knowledge sources such as models of the task and users' goals to constrain the search space. For example, with no knowledge sources, one study showed 74 percent correct word recognition, but with the addition of knowledge sources such as the goals of the task, the recognition accuracy improved to 95 percent [36]. These reports show that speaker-independent recognition of continuous speech has been achieved and demonstrated to be useful for limited domain vocabularies.

Most commercially available speech recognizers typically are not "speaker-independent" in that they require training by the intended user. Further, the spoken words to be recognized must be clearly separated by gaps of silence to facilitate word end-point detection. These characteristics of speaker-dependence and discrete-word inputs severely limit the applicability of speech recognition technologies. Recent announcements of new products, such as the "Dragon-Dictate" speaker-adaptive recognizer from Dragon Systems, relaxes the speaker-dependence constraint somewhat by allowing the user to incrementally train the recognizer during use. However, this advance still does not provide the capability needed for conversational interactivity.

The limitations of current speech recognizers arise, to a large extent, from the template matching methodology employed. During a training phase, recognizers use discrete word utterances to form templates that represent the words in a vocabulary. Following training, an utterance to be recognized is compared to each stored template, and is identified as the word represented by the most similar template. Because this approach usually requires prior identification of word end points in the unknown speech, it is not easily extended to

[1] excerpt from a MIT Media Laboratory brochure.

[2] see, for example, Clarke's "2001" [4] or Heinlein's "Number of the Beast" [14].

continuous speech recognition where word boundary detection is often inaccurate.[3] Further, speaker-independent recognition suffers since a template formed from multiple speakers' data usually results in poorer discrimination among vocabulary items than does one formed from a single speaker's data[4].

Since template matching methods are not particularly suited to speaker-independent, large vocabulary, continuous speech recognition, other methods have been devised and evaluated. In general, these attempt to deal with the variability among speakers and contexts by deriving probability distributions describing the likelihood of a speech unit given the evidence in the input data stream.

The most developed method is the hidden Markov model (HMM) which is a finite state network where transitions among states are probabilistic [26, 6]. Further, given a state transition in the network, emission of a particular symbol is also probabilistic. All probability distributions for a given model are estimated from training data. Following training, the output of the model is the likelihood estimate that the input sequence is represented by the model. A HMM requires sufficient training data to reliably estimate the probability distributions, and may be used to model speech units of any size (e.g., phonemes, words, or sentences). In fact, HMMs may be arranged hierarchically so that the likelihood of a word may be estimated given the likelihood of phonemes in the sequence of input data.

A related method is based on neural network models trained to behave as classifiers [2]. For a given input, such networks generate an output decision based on network parameters that implicitly model the probability distributions inherent in training data [12]. Although the application of neural network models to speech recognition is relatively recent, a considerable number of reports are already available that describe the performance of various types of neural network models as phoneme or digit classifiers. Currently, one of the more promising network models is the time-delay neural network [34, 33, 29] which will be discussed further, as well as the feature map classifier [17, 31].

Very high levels of speaker-dependent recognition performance on phonemes are reported using neural network classifiers, and Treurniet, Hunt, Lefebvre, and Jacobson were among the first to report investigations on speaker-independent recognition [32]. In their experiments, network performance was tested with stop-consonant data spoken by people different from those who provided the training data. Recognition accuracy of 76 percent was reported, but this was subsequently increased to about 90 percent with improved training techniques. Similarly, Leung and Zue reported 67 percent accuracy in discriminating among 16 vowels excised from continuous speech spoken by 155 different people [22].

Neural network classifiers appear to be superior to HMMs for recognition of small speech units such as phonemes [18, 33], but methods for modelling larger units such as words are not yet well developed. Currently, word recognition is accomplished more effectively by passing

[3]But see [3] for a dynamic programming method that uses discrete-word templates to identify words in continuous speech.

[4]The severity of this problem may be reduced somewhat by forming multiple reference templates for a given word based on a cluster analysis of many utterances of the same word [35].

network outputs to word-level Hidden Markov Models where they are considered estimates of phoneme probabilities [10, 9]. Such a hybrid approach gives better word recognition results than one that uses neural networks alone.

A number of neural network architectures using the back-propagation learning algorithm [28] have been proposed for the phoneme classification task. This paper reviews the derivation of the learning algorithm and its adaptation to the different architectures, and describes experiments evaluating their performance in classifying phonemes extracted from the TIMIT multi-speaker, continuous speech data base. New extensions to some of the models are also described and evaluated wherever possible.

2 The Back-Propagation Model

In neural network models, biological neurons are represented as nodes in a network, and synapses are simulated by connection weights that modify information transmitted between neurons. Each simulated neuron has an activation function that determines how inputs are combined to produce an output. Learning algorithms enable the network's "synapses" to adapt in accordance with regularities in training data.

The back-propagation learning algorithm [28] is a procedure for automatically adjusting weights on connections in a network model in order to associate patterns from the environment with a desired output vector. The "feedforward" form of the model, (as opposed to "recurrent") assumes a layered network topology where no restriction is placed on the number of layers or the number of nodes per layer. Connections are allowed between different network layers but not among nodes within a layer. One layer is designated the input layer where patterns from the environment are encoded, and another is identified as the output layer where the results of processing the input patterns are obtained. The intervening layers contain "hidden" nodes in that their inputs and outputs are not directly set or read, respectively, by the environment. A schematic of such a "multilayer" network is presented in Figure 1. Relationships between input and desired output patterns are "learned" by the network by iterative adjustments of the connection weights. In each iteration, the sizes of the weight adjustments are directly related to the errors at the output nodes. The error for a given pattern, p, is often considered to be the Euclidian distance between a target vector, $\mathbf{t}$, and an output vector, $\mathbf{o}$.

$$E_p = \frac{1}{2}\sum_j (t_{pj} - o_{pj})^2 \tag{1}$$

The objective is to minimize E_p, and this is to be done by adjusting the connection weights, w_{ji}, so that

$$\Delta w_{ji} \propto -\frac{\partial E_p}{\partial w_{ji}}$$

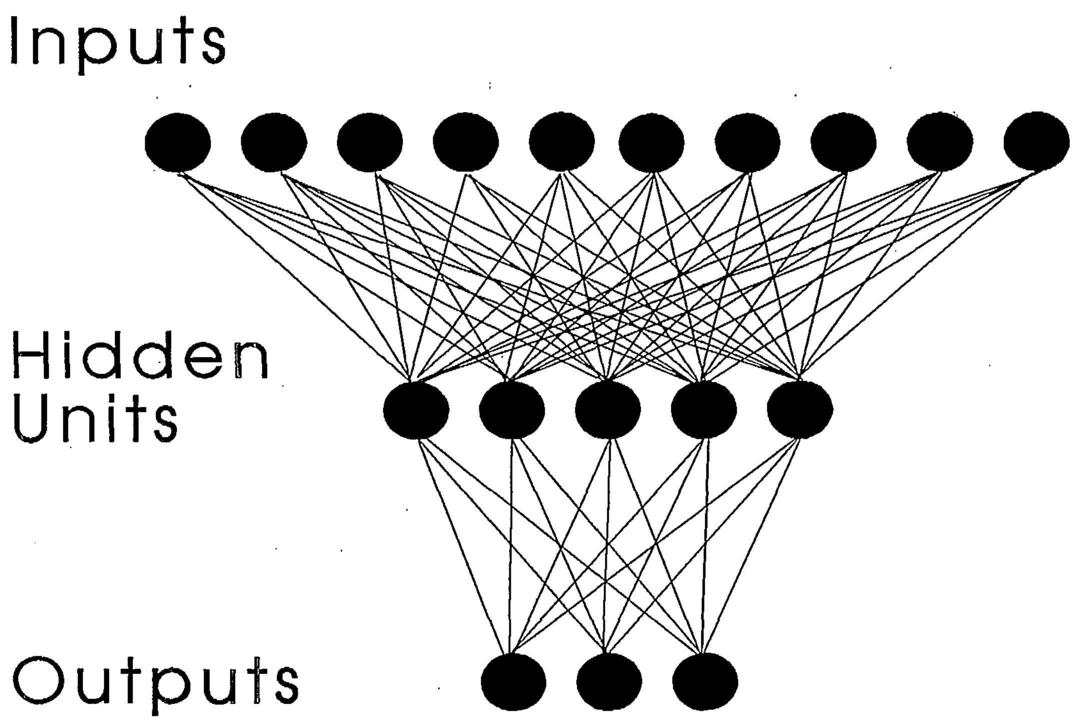


Figure 1: A multilayer network structure

Let node j's input, x_j, and output, o_j, be defined as

$$x_j = \sum_i w_{ji} o_i$$

and

$$o_j = \frac{1}{1 + e^{-x_j}} \tag{2}$$

The outputs of the nodes are computed in a forward pass through the network from the input layer to the output layer. Nodes in the current layer are indexed by j and nodes in the previous layer are indexed by i. Using the chain rule,

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial x_j} \frac{\partial x_j}{\partial w_{ji}}$$

Since $\frac{\partial x_j}{\partial w_{ji}} = o_i$, and if δ_j is defined as $-\frac{\partial E}{\partial x_j}$, then

$$-\frac{\partial E}{\partial w_{ji}} = \delta_j o_i$$

Thus, to do a gradient descent in E, the weights should be changed according to

$$\triangle w_{ji} = \eta \delta_j o_i$$

where η is a learning rate parameter. The value of δ_j for each node of the network remains to be determined.

Using the chain rule again,

$$-\frac{\partial E}{\partial x_j} = -\frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial x_j}$$

From Equation 2,

$$\frac{\partial o_j}{\partial x_j} = o_j(1 - o_j)$$

and, from Equation 1,

$$\frac{\partial E}{\partial o_j} = -(t_j - o_j)$$

So, for the nodes in the output layer,

$$\delta_j = (t_j - o_j) o_j (1 - o_j)$$

Deriving δ's for nodes in hidden layers requires the δ's already computed for the adjacent layer on the output side. That is, where k is the index for nodes in that adjacent layer,

$$\frac{\partial E}{\partial o_j} = \sum_k \frac{\partial E}{\partial x_k} \frac{\partial x_k}{\partial o_j}$$

Since

$$\frac{\partial x_k}{\partial o_j} = w_{kj}$$

and since $\frac{\partial E}{\partial x_k} = -\delta_k$ by definition, then

$$\frac{\partial E}{\partial o_j} = -\sum_k \delta_k w_{kj}$$

Therefore, δ for the jth hidden layer node is

$$\delta_j = o_j(1 - o_j)\sum_k \delta_k w_{kj}$$

When other definitions of the objective function (Equation 1) and the activation function (Equation 2) are employed, the definitions of $\frac{\partial E}{\partial o_j}$ and $\frac{\partial o_j}{\partial x_j}$ must be modified appropriately.

The learning algorithm was implemented on a Mercury MC3200AT array processor installed on an IBM-PC compatible host computer running the MS-DOS operating system. Some heuristics were also included to accelerate learning. These included Fahlman's suggestions to add a constant of 0.1 to the derivative of the logistic to minimize the effect of the "flat spot" at either end of the function, and to increase very small output errors using a hyperbolic arctangent transform [8]. The learning rate adjustment heuristics known as the delta-bar-delta method [15] were implemented as well to speed gradient descent. However, when the delta-bar-delta heuristics were not used, the learning rate for connections entering a node was reduced in proportion to the fan-in (i.e., the number of incoming connections) at that node [25].

3 Speech Data Preparation

Before discussing the experiments, this section presents information regarding the speech data base and the methods employed to prepare the data. Included is a description of the TIMIT speech data base, the software developed to extract sets of balanced training and test data from the data base, and the pre-processing method adopted to prepare the data for presentation to the neural network models.

3.1 The TIMIT Data Base

The Texas Instruments (TI) and MIT laboratories collaborated with DARPA assistance to produce a corpus of continuous speech data spoken by multiple speakers. The result is called the DARPA TIMIT Acoustic Phonetic Continuous Speech Data Base. A prototype version made available by the National Institute of Standards [11], consists of speech from 420 male and female talkers from eight dialect regions, each speaking 10 different sentences.

The data for each sentence consists of three files; a 'txt' file contains the sentence in ASCII format, a 'adc' file contains a talker's digitized speech sampled at 16 KHz, and a 'phn' file contains a list of phonemes in the sentence accompanied by the starting and ending address of each phoneme in the 'adc' file. Thus, a phoneme may be extracted from the 'adc' file by first consulting the 'phn' file for its location. Table 1 presents a list of the phonemes in the data base, including common words containing the phonemes. Note the subtle differences in pronunciation among some of the phonemes.

The data base is valuable in that it enables speech laboratories to begin research on continuous speech recognition with a minimum of effort, and also enables comparisons of research results among laboratories when the same training and test data are used.

3.2 The Data Extraction Software

The strategy for training classifier networks with supervised learning algorithms requires a set of tokens which represents equally all the classes to be discriminated. It is important, therefore, that the training set be properly balanced with respect to variables such as sex, dialect, and sentence origin (i.e., TI vs MIT). It is also necessary to be able to exclude or include particular talkers in order to create a test set in which talkers are different from those in the training set.

To accomplish this, software was written[5] which created a directory file for each phoneme. This file contained the following information about examples of the phoneme wherever they occurred in the data base: phoneme label, dialect, sex, speaker label, sentence label, 'adc' file start address, stop address, token length, left context phoneme label, left context token length, right context phoneme label, and right context token length.

A second program scanned the directory files to select phonemes according to specified criteria, obtained the digitized phoneme data from the appropriate 'adc' file, and created the spectrogram representation starting at the beginning of the left context phoneme and ending at the end of the right context phoneme. The above header information and the spectrogram data were appended to a data file for a particular phoneme class .

One data file for each phoneme was used for training networks to recognize that phoneme class. Two more data files for each phoneme were also created to test the generalization performance of trained networks. One set of phoneme data files (Test Set A) was used for periodic testing during training to identify deleterious effects of over-learning. The other (Test Set B) was used to provide an independent assessment of generalization performance. Each test set was obtained from an independent set of speakers.

Each of the three data sets were obtained from a different group of speakers to permit

[5]The initial version of the software was written for the MacIntosh computer in collaboration with D. Graf at the Computing Research Lab, Bell Northern Research. The software was subsequently adapted at CRC to the MS-DOS platform.

Table 1: Phonemes in the TIMIT Data Base

	Phoneme	Pronunciation		Phoneme	Pronunciation
Stops:			Vowels:		
	b	bee		iy	beet
	d	day		ih	bit
	g	gay		eh	bet
	p	pea		ey	bait
	t	tea		ae	bat
	k	key		aa	bott
	dx	muddy		aw	bout
	q	bat		ay	bite
Affricates:				ah	but
	jh	joke		ao	bought
	ch	choke		oy	boy
Fricatives:				ow	boat
	s	sea		uh	book
	sh	she		uw	boot
	z	zone		ux	toot
	zh	azure		er	bird
	f	fin		ax	about
	th	thin		ix	debit
	v	van		axr	butter
	dh	then		ax-h	suspect
Nasals:			Others:		
	m	mom		pau	pause
	n	noon		h#	non-speech
	ng	sing		cl	closures
	em	bottom			
	en	button			
	eng	washington			
	nx	winner			
Semivowels Glides					
	l	lay			
	r	ray			
	y	yacht			
	hh	hay			
	hv	ahead			
	el	bottle			

speaker-independent evaluations of performance. The labels for speakers in Test Set A were *mklw0, mdwd0, madc0, fsjs0, mjls, fgmb0, fpad0, mrcs0, fgwr0,* and *fcmh1,* while those in Test Set B were *fsah0, fslb1, mjeb0, fgcs0, mjws0, fkdw0, fjsa0, mlih0, mjrk0, mses0,* and *mrlk0.* The remainder of the speakers made up the training set. The frequencies of phonemes in each set were distributed equally across either sex and across the eight dialects. For consistency with others[21, 27], only data from the *si* and *sx* sentence types were included.

3.3 Speech Signal Processing

The time domain speech data sampled at 16 KHz was transformed to a frequency domain representation via a fast fourier transform routine[6] following a procedure similar to that described by Lang, Waibel, and Hinton [19]. A sequence of 320 points (20 msec) was pre-emphasized with a 6 dB/octave filter and processed by a Hamming window. Then the last 64 points from the 320 point window were folded into the first 64 points, resulting in a 256 point vector. These 256 values were processed by a 128 point complex fourier transform which treats even samples as real values and odd samples as imaginary. The resulting 128 values obtained from the FFT were transformed by the function $20 * \log_{10}(r^2 + i^2)$. The next frame of data was obtained by sliding the Hamming window forward in time by 64 points (4 msec) and repeating the process.

The output resulting from the above process was reduced to 16 values from 128 by forming weighted sums of the 128 values. That is, a 16 channel, third octave filter bank [1] was simulated by forming inner products of the output vector with each of 16 weights vectors[7]. The center frequencies of the simulated filters were 125, 185, 283, 400, 500, 630, 800, 1000, 1250, 1600, 2000, 2500, 3150, 4000, 5000, and 6300 Hz, forming an approximate mel scale filter bank. To reduce the amount of data even further, pairs of successive frames from the simulated filter bank were averaged to give a temporal resolution of eight msec.

4 Experiments with Simple Network Architectures

Network architectures were compared with regard to their ability to generalize to test data not in the training set. Training continued until generalization performance failed to improve or began to decline. The phonemes *w*, *l*, and *ow* were often used in classification experiments, since the pattern of confusions in early experiments indicated that these had some overlap in the input space.

Because the experiments were exploratory and because training times usually were exten-

[6]Appreciation is extended to G. Gagnon and A. Vincent for discussions during an early stage of the project concerning the application of fourier analysis.

[7]The shape of each filter was triangular on a linear scale, with a bandwidth of .2316 times the center frequency, and 3 dB points at half the bandwidth from the center frequency on each side.

sive, the values of more than one variable often were allowed to differ between experiments. Therefore, comparisons among experiments should be made with caution. For example, no attempt was made to control for the total number of weights in the different network architectures, and often the sizes of the training sets differed due to limited training data or limited memory for storing the training set[8]. However, the experiments are considered adequate for discovering clear advantages or disadvantages in network architectures. Only representative experiments are reported.

Each training token consisted of 14 frames (112 msec) of spectrogram data (unless otherwise noted); these included 3 frames of the left context phoneme and 11 frames from the phoneme to be classified. Data from the right context phoneme was also included to complete the token when necessary. If there were insufficient data in the phoneme and its left and right context, the token was completed with zero values.

4.1 The Multilayer Network

The layered network structure (Figure 1) described in Section 2 has come to be called a multilayer network to distinguish it from the perceptron model which has no hidden layers. The perceptron is limited in that it can distinguish only sets of patterns that are linearly separable. The hidden layer(s) in the multilayer network removes this restriction by allowing hidden units to respond to different combinations of input values which the output units can then combine to yield a non-linear segmentation of the input space.

A multilayer network with 224 input nodes (14 frames), 10 hidden nodes, and three output nodes was trained to discriminate among *w*, *l*, and *ow*. The training set consisted of 300 patterns per class. The best performance on the test set in terms of percent correct was obtained after 1000 iterations through the training set with *w* (91.7%), *l* (67.3%), and *ow* (80.0%). Overall, generalization performance was 75.5% correct (overall performance was not the average of the class accuracy rates since the class frequencies in the test data were not equal).

In a more ambitious experiment, a network with 240 input nodes (15 frames), 20 hidden nodes, and 39 output nodes was trained to discriminate among 39 phoneme classes. This number of classes was obtained by merging some of the phonemes in Table 1 as indicated by the brackets in the following list defining the classes: *p, t, k, b, d, g, (m,em), (n,nx,en), (ng,eng), s, z, ch, th, f, (sh,zh), jh, dh, v, (l,el), r, y, (hh,hv), w, eh, (ao,aa), (uw,ux), (er,axr), ay, ey, aw, (ax,ah,ax-h), (ix,ih), ae, uh, oy, iy, ow, (pau,h#,cl)*. The maximum training set size that could be accomodated in memory was 100 patterns per class. Best generalization performance of 44.8% correct was achieved after 1200 passes through the training set. Additional improvement was obtained by replacing the training set every 50 passes with a different set of training data that overlapped the preceding set by 80 percent. After an additional 3200 iterations, overall generalization performance was increased to 47.1

[8]The array processor contained six Mbytes of memory which limited the number of examples that could be used at one time for training the neural networks.

percent.

Because of the limited numbers of speakers in the test sets, the phoneme frequencies were not at all equal. Table 2 lists the 39 classes, with associated frequencies and proportions correct for each test set alone and for the combined test sets.

It is clear from the table that performance varied considerably among classes. Accuracies for the classes in the combined test sets ranged from 0.0 to 83.3 percent.

4.2 A Recurrent Neural Network

A recurrent network structure removes the restrictions on interconnectivity among nodes within a layer, and the learning algorithm must usually be modified to enable node outputs to settle to a stable state. However, a restricted form of the recurrent network model, often attributed to Elman, may still be trained with the feedforward learning algorithm [7, 5]. It is useful for encoding the trajectory of a pattern through time. Therefore, it seems a natural candidate for representing temporally varying information such as speech. This network is a multilayer network in structure, but is distinguished by the nature of the data presented to the input units (Figure 2). Part of the input array represents the input data at the current time step. The remaining input units, however, are used to represent the activation levels of the hidden units at the previous time step. Therefore, as learning proceeds, the states of the hidden units are affected by the external data applied to the input, as well as the previous activation levels of the hidden units. As a result, the states of the hidden units in a trained network are differentially affected by different pattern sequences, and different sequences can produce different output trajectories.

The recurrent network described above was trained to discriminate among the 39 classes. The input layer of 36 nodes accomodated one frame of data and the state of 20 hidden nodes. Thirty-nine output units represented the classes. The training set consisted of 128 patterns per class. Best generalization performance of 34.3% was obtained after 3800 passes through the training data.

4.3 The Time Delay Neural Network

A multilayer network typically represents the temporal dimension by mapping it onto a spatial input array. A consequence of this is that the network becomes very sensitive to the precise postioning of relevant features in the input array during training. That is, if a feature in different patterns of the same class appears at different positions in the input array, the effect is to smear the information presented to the network, and to reduce its sensitivity to the feature.

The Time Delay Neural Network (TDNN) structure, schematically presented in Figure 3, was specifically developed to avoid this problem [18]. Each input node is presented

Table 2: Generalization Performance of the Multilayer Network

Class	Set A Freqency	Set A % Correct	Set B Frequency	Set B % Correct	Combined % Correct
p	50	60.0	64	70.3	65.8
t	78	48.7	72	44.4	46.7
k	94	69.1	88	68.2	68.7
pau	592	46.8	572	44.6	45.7
dx	30	76.7	40	65.0	70.0
b	46	63.0	44	84.1	73.3
d	42	23.8	62	37.1	31.7
g	18	55.6	6	66.7	58.3
m	86	55.8	83	56.6	56.2
n	152	27.6	172	30.2	29.0
ng	25	60.0	18	66.7	62.8
s	128	65.6	102	62.7	64.3
z	66	60.6	54	66.7	63.3
ch	12	75.0	6	100.0	83.3
th	14	50.0	14	64.3	57.1
f	44	72.7	42	69.0	70.9
sh	25	80.0	36	77.8	78.7
jh	24	70.8	30	66.7	68.5
dh	52	40.4	46	45.7	42.9
v	46	54.3	42	40.5	47.7
l	110	57.3	112	44.6	50.9
r	106	54.7	98	60.2	57.4
y	14	78.6	16	56.3	66.7
hh	40	65.0	22	68.2	66.1
w	48	50.0	58	58.6	54.7
eh	66	25.8	66	15.2	20.5
ao	66	47.0	68	41.2	44.0
uw	32	25.0	32	43.8	34.4
er	72	56.9	80	51.3	53.9
ay	48	70.8	42	57.1	64.4
ey	34	52.9	44	52.3	52.6
aw	14	50.0	4	75.0	55.6
ax	106	25.5	118	19.5	22.3
ix	244	31.1	278	21.6	26.1
ae	50	62.0	40	37.5	51.1
uh	4	25.0	14	21.4	22.2
oy	10	0.0	6	0.0	0.0
iy	94	56.4	100	62.0	59.3
ow	30	33.3	26	46.2	39.3
OVERALL		49.0		46.4	47.7

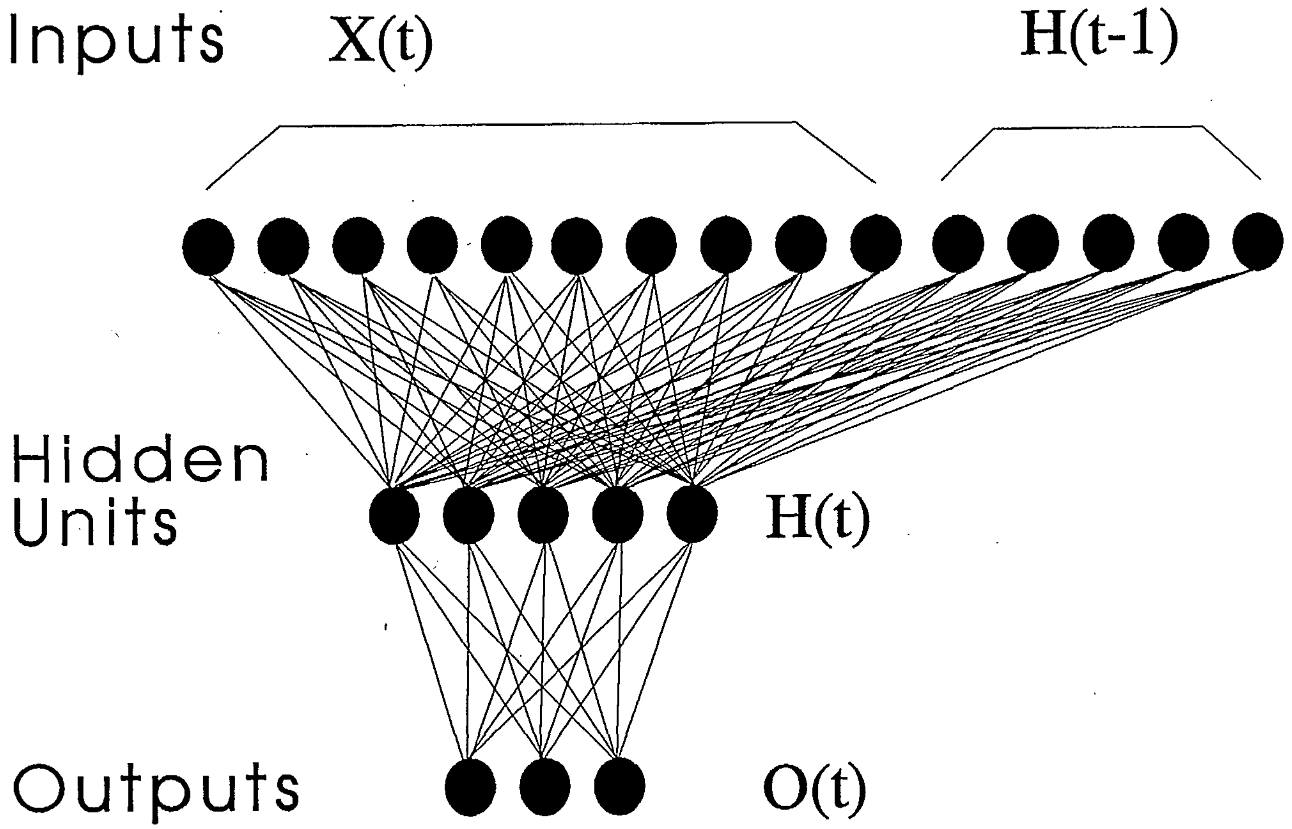


Figure 2: A recurrent network structure

Table 3: Classifying phoneme subsets with a TDNN

Classes	Percent correct	Overall percent correct
w, l, ow	90.9, 54.1, 85.3	69.4
y, uw, iy, ih	75.6, 71.8, 71.2, 61.0	68.2
ae, eh, ey	59.0, 68.0, 76.2	68.3
ao, ay, aw, oy	58.4, 68.9, 50.0, 55.6	61.6
ah, ih, uh	79.4, 75.3, 66.7	76.4
dh, v	80.5, 86.7	83.3
f, th	77.6, 82.6	78.7
ch, jh, sh	67.4, 69.4, 80.7	73.3
s, z	82.6, 71.0	78.0
r, er	89.3, 74.5	84.5
m, n	71.1, 58.8	61.8

sequentially with the value of a pattern element at several sucessive points in time. Since the same weights are adapted at each time step, they are trained to respond to the significant event wherever it occurs in the sequence. The outputs of the first layer of the network form the inputs to a second layer of weights. Here again, an input unit receives data from the successive time steps in the first layer, and the same weights (different from the first layer) are adapted at each time step. Finally, the outputs of the second layer's units are integrated across time to yield a final classification decision.

An important advantage of the TDNN structure, compared to the simpler multilayer network, is the stability of its outputs while a stream of data is passed through its input window. The TDNN will signal the presence of a feature as it moves into the window until it moves out again. The multilayer network, on the other hand, often responds erratically until the feature is properly aligned within the input window.

The dimensions of the TDNN structure employed were similar to those recommended by Waibel et al. [33], and differed only in the network input width at a given delay due to the difference in frame rates. Their network input spanned 30 msec which corresponded to three spectrogram frames, while our network input spanned 32 msec or four frames. There were eight output nodes per delay at the first layer of the TDNN, and the input for the second processing layer spanned the result of five first layer time steps.

Each of a number of TDNN networks learned to discriminate among a set of similar classes using 200 patterns per class. Each network processed tokens consisting of 12 frames of data, and training was terminated after 5000 passes through the data. The test performance results for each network are presented in Table 3.

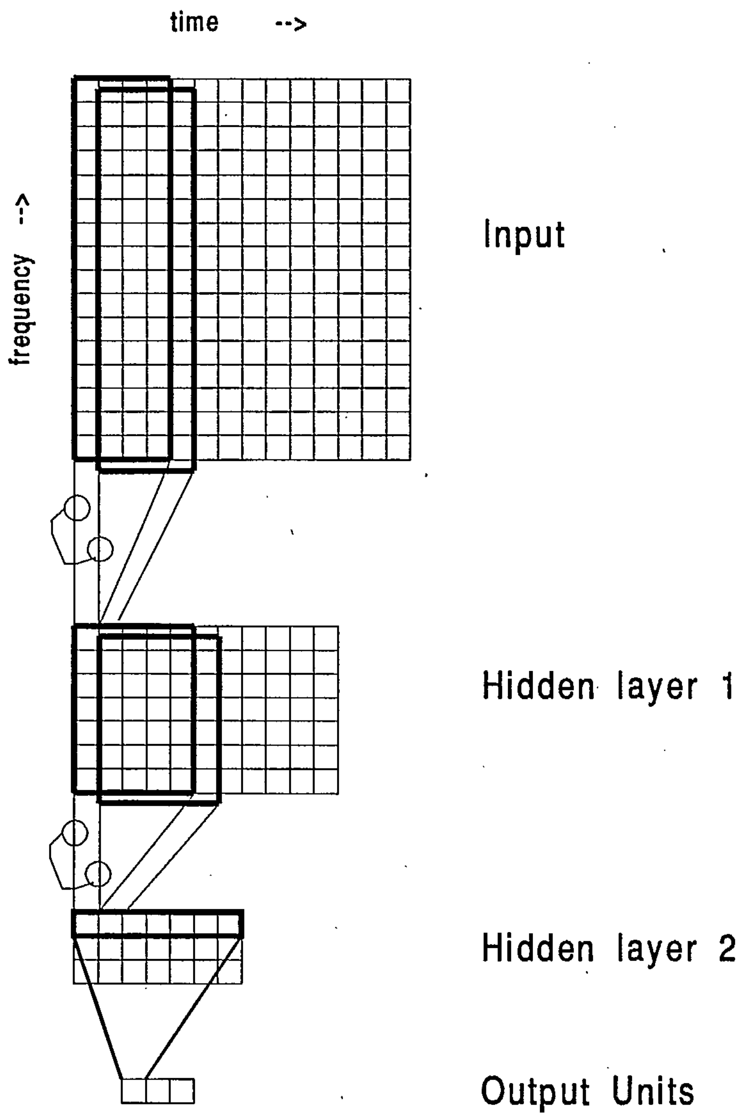


Figure 3: The Time Delay Neural Network structure

5 Experiments with Modular Architectures

A difficulty with the back propagation model is that it does not scale up well because of cross-talk within a single network structure. Crosstalk occurs when a hidden unit receives conflicting error information from different output units [16], and larger networks provide more opportunities for this to occur. Another problem with large networks is that there are more parameters to adjust, and the number of training examples required for accurate adjustment increases proportionately with the number of parameters. Therefore, it would seem advantageous to have smaller network modules learn to handle parts of the problem, and then to combine the decisions of the individual modules.

However, simply observing the biggest output from all the modules trained with examples from disjoint regions of the input space, cannot reliably determine the decision of the system as a whole. By way of analogy, a function determined from a finite number of data points by regression analysis may be used to predict a value from a new point drawn from the same distribution as the original set. However, if the data point were drawn from a different distribution where the function should have had a different form, the predicted value would be quite incorrect. The function would extrapolate into a region where it did not apply. Similarly, a back propagation network which uses a "semi-linear" activation function such as the logistic (Equation 2), will also give an unreliable classification when presented with examples from a region of the input space not sampled during training. That is, the network will interpolate along the function learned, but may not extrapolate accurately into unexplored regions.

Therefore, a way must be found to identify the part of the input space to which an unknown sample belongs in order to attend to the network module which has meaningful output. Several network architectures, described below, have been proposed to accomplish this. Also described are new variations on some of the architectures which were developed in hopes that these would improve learning and generalization performance. The rationale for each modification is explained, the changes to the learning algorithms are described, and the effects on performance noted.

5.1 The Meta-Pi Network

Hampshire and Waibel [13] presented an architecture consisting of a number of networks each already trained to classify stimuli from a restricted region of the input space. They proposed an additional network, called a "Meta-Pi" network, to modulate the output of each sub-network to produce a global output. In the generic architecture shown in Figure 4, the trained networks are the "subnets", and the Meta-Pi network is referred to as an "input region identifier". To obtain a global output, the outputs of each subnet are multiplied by an output of the Meta-Pi network and same-class outputs are summed. The winning classification is determined from the global outputs using a winner-take-all strategy. The Meta-Pi network, in effect, learns which subnet or combination of subnets to use to classify a particular input sample. The approach assumes that at least one of the subnets correctly classifies the input, since the linear combining function which computes the global output cannot invert the outputs of the individual subnets.

The Meta-Pi learning algorithm was derived for the case where the subnets all attempt to make the same classification decision (the conjoint classification task), and for the case where the subnets are trained to make different classification decisions (the disjoint classification task).

For the conjoint task, the nth global output is the sum of the nth output (ρ_n) from each of the K sub-networks weighted by the kth output of the Meta-Pi module corresponding to that sub-network. That is,

$$O_n = \frac{1}{\mu} \sum_k \rho_{kn} M_k \tag{3}$$

where

$$\mu = \sum_k M_k$$

The error derivative for the kth output of the Meta-Pi module is defined as

$$\frac{\partial E}{\partial M_k} = \frac{\partial E}{\partial \vec{O}} \frac{\partial \vec{O}}{\partial M_k}$$

From Equation 3

$$\frac{\partial O_n}{\partial M_k} = \frac{1}{\mu}(\rho_{kn} - O_n)$$

If the error for the global output is defined as

$$E = \frac{1}{2} \sum_n (O_n - D_n)^2$$

where D_n is the desired output, then

$$\frac{\partial E}{\partial O_n} = O_n - D_n$$

Therefore,

$$\frac{\partial E}{\partial M_k} = \frac{1}{\mu} \sum_n (O_n - D_n)(\rho_{kn} - O_n)$$

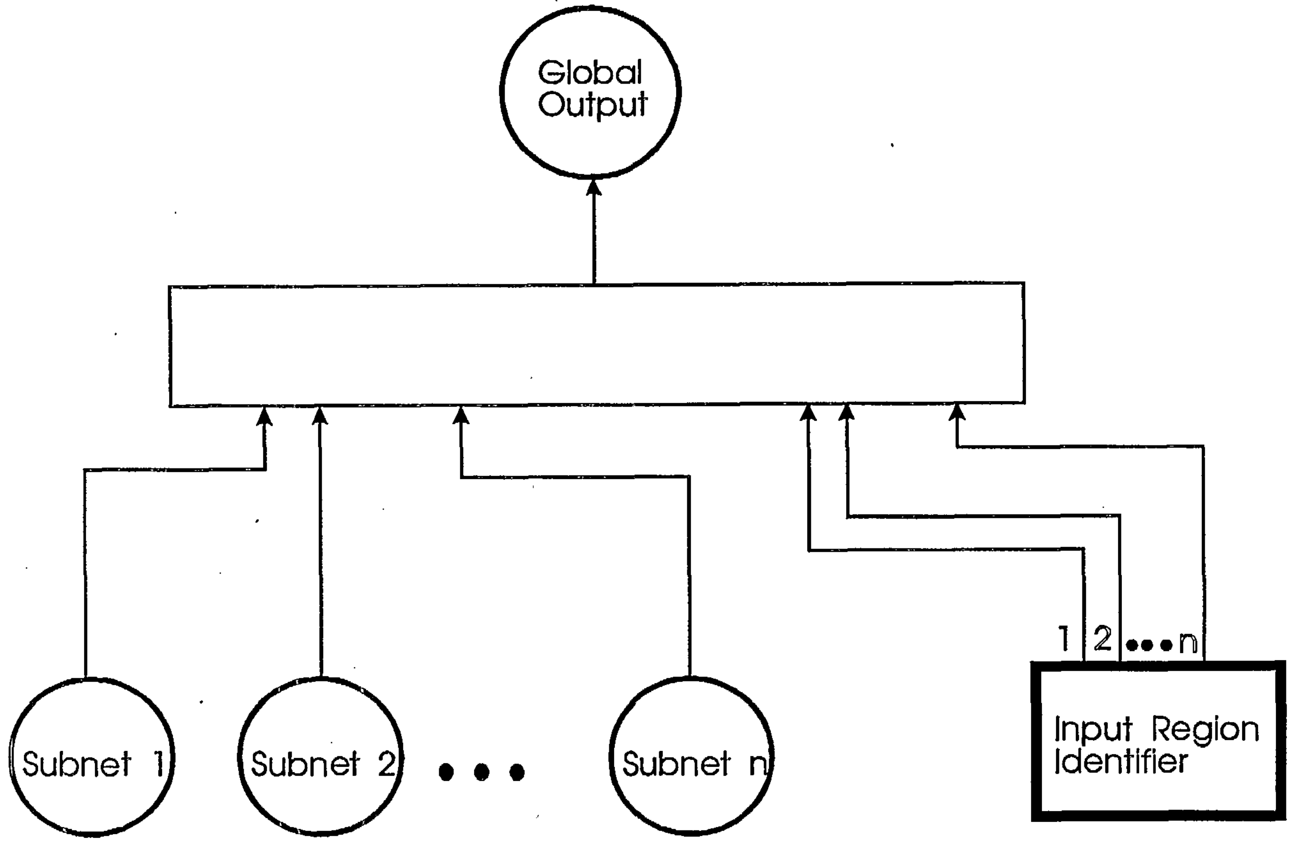


Figure 4: A generic modular architecture

An example of the conjoint task was analysed by Hampshire and Waibel [13]. Each of six subnets was trained with data from a different speaker to discriminate among the phonemes *b*, *d*, and *g*. Then, the Meta-Pi network was trained to weight the three outputs of the six subnets so that the weighted sum of the subnet outputs correctly classified the sample. After training this architecture, a new sample could be classified without first identifying the speaker and the appropriate subnet. Instead, the Meta-Pi network took the responsibility of weighting the outputs of all the subnets on the basis of speech characteristics such as absolute position of the formants.

Hampshire and Waibel also proposed a variation of the learning algorithm for the case where each subnet is trained to make different rather than the same classifications. In this case, the above definition of the global output is modified to give k times n outputs instead of n outputs.

$$O_{kn} = \rho_{kn} \frac{M_k}{\mu} \tag{4}$$

From this definition

$$\frac{\partial O_{kn}}{\partial M_k} = \frac{1}{\mu}(\rho_{kn} - O_{kn})$$

and the error gradient, $\frac{\partial E}{\partial M_k}$, is modified accordingly.

Their proposal to test the algorithm was to train one subnet to discriminate among *b*, *d*, and *g*, and a second subnet to discriminate among *p*, *t*, and *k*. These sets of stop consonants differ in the degree of voicing present. Therefore, the Meta-Pi network would be expected to weight the subnet outputs in proportion to the amount of voicing present in the input sample. In this case, the number of global outputs is the total number of subnet outputs. Hampshire and Waibel [13] did not empirically investigate this form of the Meta-Pi learning algorithm. Since combining disjoint classifiers is the problem of interest here, the performance of this form of the Meta-Pi algorithm was investigated. Several experiments were performed with this architecture to weight the outputs of separate networks each trained to make different classification decisions.

In the first experiment, a single TDNN was trained to discriminate among the phonemes *s*, *z*, *r*, and *er*. Then, two more networks were trained to discriminate between *s* and *z*, and between *r* and *er*, respectively. The latter two networks became part of the Meta-Pi structure and its classification accuracy may be compared with that of the first TDNN. The results of the generalization tests, displayed in Table 4, show that the Meta-Pi architecture's performance was roughly equivalent to the TDNN network that learned to classify all four categories. Therefore, the Meta-Pi module, itself a TDNN network, successfully learned to distinguish the part of the input space containing *s* and *z* from the part containing *r* and *er*.

Table 4: S, Z, R, ER Experiment

Architecture	S	Z	R	ER	Overall
TDNN	81.0	74.1	85.8	69.1	79.1
TDNN	82.6	71.0			78.0
TDNN			89.3	74.5	84.5
Meta-Pi Net	82.2	69.8	82.2	74.5	78.3

Table 5: Experiment with Voiced and Unvoiced Stop Consonants

Architecture	P	T	K	B	D	G	Overall
TDNN	75.0	78.1	71.0				74.7
TDNN				80.2	69.0	76.5	73.4
Meta-Pi Net	36.6	40.8	26.6	61.3	39.4	49.0	40.2

A potentially more difficult problem for the Meta-Pi network module was suggested by Hampshire and Waibel; that is, discriminating between voiced and unvoiced stop consonants. In this experiment, two TDNN networks were trained to discriminate among the voiced and unvoiced stop consonants, respectively. Then the outputs from these networks were weighted by the Meta-Pi module to produce a global output. The results of the generalization tests, displayed in Table 5, show a quite drastic decline in classification accuracy based on the global output as compared to the performance of the two separate TDNNs. Apparently, the regions of input space containing the voiced versus the unvoiced stop consonants were difficult to discriminate.

Another way to decompose the stop consonants is in terms of place of articulation. Three TDNNs were each trained to discriminate between two classes - *p* and *b*, *t* and *d*, and *k* and *g*, respectively. As before, a Meta-Pi module weighted the outputs of the three networks to produce a global output. Table 6 shows results of generalization tests on the three TDNNs and on the Meta-Pi system. The individual TDNNs did well, but it was clear on closer examination that each discriminated primarily on the basis of the voicing cue. Consequently, the network trained with *p* and *b* would discriminate almost as well between *t* and *d*. The Meta-Pi module, which had learned to weight the other networks' outputs on the basis of place of articulation, did somewhat better than in the preceding experiment. However, performance was still considerably lower than that of the individual TDNNs.

Table 6: Experiment Using Place of Articulation of Stop Consonants

Architecture	P	T	K	B	D	G	Overall
TDNN	80.4			87.7			83.9
TDNN		79.9			75.6		78.0
TDNN			82.2			78.4	81.8
Meta-Pi Net	49.1	65.7	49.7	70.8	52.0	52.9	56.9

5.2 The System of Experts

Nowlan and Hinton [24, 23] proposed a learning procedure for a system consisting of a number of separate networks, each of which learns to classify a subset of the training cases. At the same time, a "gating" network learns to weight the outputs of these "expert" networks, and causes the expert networks to compete with each other for control of particular regions of the input space. In the generic modular architecture shown in Figure 4, the expert networks are the "subnets" and the gating network is the "input region identifier". The gating network of a trained system may be considered a manager who knows which of the experts is best qualified to deal with the region of the input space from which a sample of interest is drawn.

The expert networks learn via a supervised learning algorithm which requires that the desired outputs be specified. However, the gating network's learning is unsupervised, and adaptation depends on the relative performances of the expert networks. The expert networks all receive the same input data which may, however, be different from the data presented to the gating network.

The model has been evaluated using a number of relatively small problems and performed as desired [23]. However, when a large number of classes are to be discriminated and the input space is large and complex, the model is computationally slow on current serial machines. This is because many candidate networks must be available during training (even though only a few may ultimately become experts), and because each candidate must have the structural potential for dealing with all of the classes. Nevertheless, because of its elegance, the model was considered an interesting candidate for decomposing a large task into smaller sub-tasks.

The objective function to be maximized for the system of k experts is the log probability of generating the desired output vector under the mixture of Gaussians model employed [23].

$$L = -\log \sum_{k} (p_k e^{-\frac{1}{2}\|\mathbf{d}-\mathbf{y_k}\|^2/\sigma^2}) \tag{5}$$

where **d** is the desired output vector, **y** is the output of the kth expert network, p_k is the gate network output corresponding to the kth expert network, and

$$\| \mathbf{d} - \mathbf{y_k} \|^2 = \sum_i (d_i - y_{ki})^2$$

For convenience, let us define

$$L = -\log U$$

and

$$U = \sum_k (p_k e^{M_k})$$

where, for the kth expert net,

$$M_k = -\frac{1}{2} \sum_i (d_i - y_{ki})^2 / \sigma^2$$

Then the error gradient for the ith output unit of the kth expert with respect to its input is

$$\frac{\partial L}{\partial x_{ki}} = \frac{\partial L}{\partial U} \frac{\partial U}{\partial M_k} \frac{\partial M_k}{\partial y_{ki}} \frac{\partial y_{ki}}{\partial x_{ki}}$$

Since

$$\frac{\partial L}{\partial U} = -\frac{1}{U}$$

$$\frac{\partial U}{\partial M_k} = p_k e^{M_k}$$

$$\frac{\partial M_k}{\partial y_{ki}} = (d_i - y_{ki})/\sigma^2$$

and, for the logistic activation function (Equation 2),

$$\frac{\partial y_{ki}}{\partial x_{ki}} = y_{ki}(1 - y_{ki})$$

then

$$\frac{\partial L}{\partial x_{ki}} = -\frac{1}{U} p_k e^{M_k} (d_i - y_{ki}) y_{ki} (1 - y_{ki}) / \sigma^2$$

The error gradient for the ith gating network output is derived after first choosing the "softmax" activation function for the output units; that is, where x_i is the input to the ith output unit,

$$p_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

The objective function then becomes

$$L = -\log \sum_i \left(\frac{e^{x_i}}{\sum_j e^{x_j}} \right) e^{M_i}$$

After some manipulations, keeping in mind that the index *j* includes *i* in the above softmax definition[9],

$$\frac{\partial L}{\partial x_i} = -\frac{1}{U} p_i e^{M_i} + p_i$$

The error gradients with respect to the inputs for the gating network and all expert networks are now propagated back through the networks as before to derive the error gradients with respect to all the weights.

At the beginning of learning, the gating network should consider all experts equally probable for any input sample. This is obtained by initializing to zero all the weights leading to the output units.

A system of experts architecture with 10 multilayer "expert" networks and one multilayer gating network was trained to discriminate the *s* and *z* phonemes. All networks had 240 inputs and 10 hidden nodes. The expert networks each had two outputs while the gating network had 10 outputs corresponding to the number of expert networks. The training set consisted of 200 samples per class.

Best generalization performance of the system was obtained after 1500 passes through the training set with *s* (81.0%) and *z* (72.2%). Overall performance was 77.5 percent correct. This result is very similar to that obtained with a single TDNN network (Table 3).

A similar experiment was performed to discriminate among *w*, *l*, and *ow*. After 5550 training epochs, generalization performance, which was still improving very slowly, was *w* (90.9%), *l* (56.0%), and *ow* (75.0%). Overall performance was 67.5 percent correct, which is within one percent of that obtained with a single TDNN (Table 3), but seven percent less accurate than the single multilayer network (Section 4.1).

Clearly, performance levels for both sets were not improved over that obtained from a single network classifier. Closer examination of several training runs revealed that each expert network selected by the gating network had learned to turn on only one of its outputs for any input sample. Therefore, the gating network was making all the decisions, and the system's performance depended only on the relative values of the gating network's outputs. Since only one network was doing all the work, the equivalence of performance with that of a single network is not surprising.

The reason for the lack of improvement with the system of experts is not clear. The implementation is thought to be correct since it appeared to perform as expected on the test problems described by Nowlan [23]. One possibility may be that the amount of training data, although adequate for a single network, was simply insufficient for training the several networks that eventually learned to specialize. Insufficient data might have led to grossly imbalanced class frequencies in a given expert's region of input space. Large differences in

[9]The assistance of D. Graham, Canadian Space Agency, in confirming this equation is appreciated.

prior probabilities for the classes would then have adversely biased the network's learning, perhaps to the point of settling in a local minimum which only allowed one output to become activated. Increasing the training set size would help to reduce such an effect. It is also possible that any advantage from using the system of experts architecture might be obtained more easily if disjoint regions of the input space are to be placed in the same class, as was the case in the vowel classification experiments reported by Nowlan. The distribution of classes in the present experiments is difficult to determine due to the high dimensionality of the input space.

If a larger sample size is needed to enable the system of experts to perform properly, the architecture may be considered impractical since currently available computing equipment would take too long to handle all phoneme classes.

5.2.1 Extension to the System of Experts

Calculation of the a posteriori probabilities for each expert network requires an estimate of the variance (i.e., noise) used by the formula for the multivariate Gaussian distribution. Nowlan (personal communication) recommends that training should begin with a variance of about 0.25, and that it should be lowered gradually to about 0.05 as training proceeds.

Instead of using a fixed schedule for modifying the variance, adjusting it adaptively in response to some cost function would enable it to be sensitive to the nature of the input data. In this way, the probability that a given output is generated by a particular expert network could be influenced not only by the gating network output, but also by the variance associated with the input pattern.

Adaptation of the variance can be accomplished by a new network module analagous to the gating network that we will call the "noise" network. The noise network has one output corresponding to each expert network. During training, each output magnitude is an estimate of the variance to be used by a corresponding expert network. Initially, these are approximately 0.5 (with the asymmetric logistic activation function) due to random weight initialization, but can become smaller or larger as learning proceeds. The noise network is adapted like the gating network, but with the error derivative proportional to $\frac{\partial L}{\partial V_k}$, where V_k is the variance for the kth expert network. The derivative is obtained from the objective function.

$$L = -\log U$$

where

$$U = \sum_k \frac{p_k}{\sqrt{V_k}} e^{-\frac{z_k}{2V_k}}$$

$$z_k = \| \mathbf{d} - \mathbf{y_k} \|^2$$

$$V_k = \sigma_k^2$$

Then,

$$\frac{\partial L}{\partial V_k} = \frac{\partial L}{\partial U}\frac{\partial U}{\partial V_k}$$

So

$$\frac{\partial L}{\partial V_k} = -\frac{p_k e^{-\frac{z_k}{2V_k}}}{2U} \cdot \frac{z_k - V_k}{V_k^2\sqrt{V_k}}$$

In tests using the noise network, its outputs tended to become negatively correlated with the outputs of the gating network, as expected. That is, after some training, a network assigned by the gating network to the region of the input space from which the current sample was drawn, was also given a small variance by the noise network.

No obvious improvement in the performance of the system of experts was gained with this modification. Further investigation is required with controlled types of data distributions to assess the benefit of adaptively controlling the variance in this way.

5.3 The Minos Architecture

The Minos architecture proposed by Smieja [30] is another approach for learning the region of input space that is relevant to a particular network module. In this sense, it is similar to the system of experts described above, but the learning rules are somewhat less elegant. It consists of a number of Minos modules which, for our task decomposition problem, each receive the same input data. Each Minos module consists of a worker network and a monitor network (Figure 5). The worker network attempts to learn to classify the input sample, and the monitor network learns how much confidence to have in the worker's decision about that particular input.

When learning to classify a given data sample, outputs are derived for both the worker and monitor of each Minos module. The performance of each worker is then assessed by comparing the similarity of the pattern of its outputs with the target pattern. Smieja recommends normalizing the output and target values so that the respective minima and maxima are equal, then measuring the similarity of output and target with the absolute value metric. The worker with the highest similarity score is chosen to learn the pattern, while all other workers refrain from learning. However, all of the monitors learn. The winning worker's monitor learns to raise its confidence in its partner, while each losing worker's monitor learns to lower its confidence in its partner's performance for that input sample. In this way, a Minos module learns to specialize on a particular region and to communicate that a sample belongs to that region.

To classify an unknown sample after training, outputs are generated by all the monitors, as well as by the worker belonging to the monitor expressing the highest confidence. The final classification decision is obtained from this worker module.

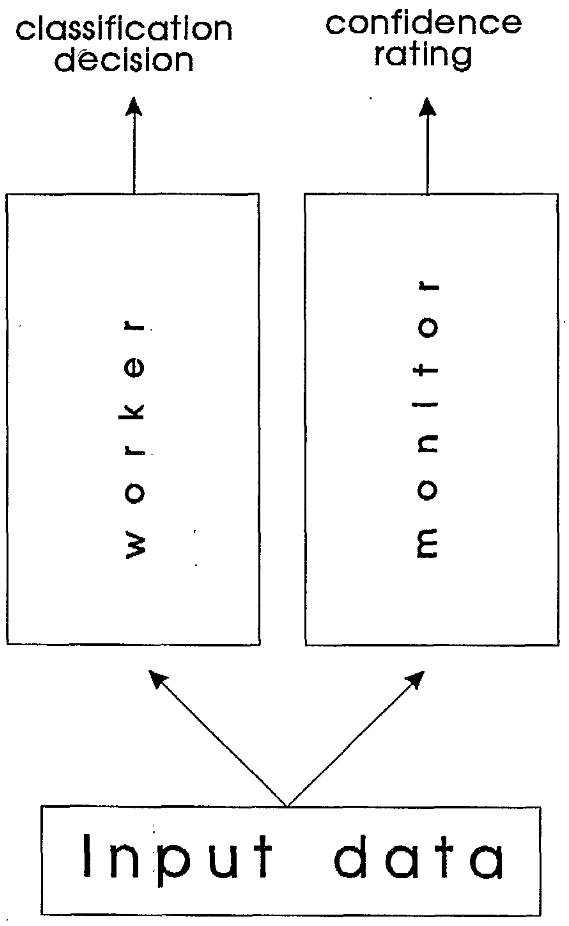


Figure 5: A Minos module

A Minos architecture with 10 modules was trained to discriminate among *w*, *l*, and *ow*. Each worker network had 224 input units, 10 hidden units and three output units. Each of the monitor networks had the same number of input and hidden units, but had only one output unit. A monitor's target during learning was 1.0 to indicate high confidence, and 0.0 for low confidence. After 3200 epochs, the obtained generalization performance was *w* (81.3%), *l* (61.8%), and *ow* (76.7%). Overall performance of 69.1 percent correct was similar to that of the system of experts and about six percent less than that of a single multilayer network. Unlike the system of experts, however, the Minos workers did not exhibit an apparent local minimum that allowed only one output to turn on.

Again, there was no improvement in generalization performance over the single network. Performance may have improved with a larger training set since parts of it were eventually allocated to different Minos modules as they learned to specialize. However, as with the system of experts, a much larger training set size would have produced an unacceptable increase in training time.

5.3.1 Extension to the Minos Architecture

With the procedure described above, a worker module makes weight adjustments or "learns" only if its output array is closest to the target array. Perhaps non-winning workers should be encouraged to increase their distance from the target as well. An objective function that approaches its minimum as the winning worker module's error becomes smaller relative to those of the other workers may be defined as follows.

$$J = \frac{V_w}{\sum_{m \neq w} V_m}$$

where w is the index of the winning worker module, and V_m is the distance of the mth worker's output vector from the target vector. Although many distance measures are possible, a useful one is the sum of the squared differences between output and target after scaling both such that the mid-range of each is zero. That is, for the mth worker module,

$$V_m = \frac{1}{2}\sum_i((y_{mi} - \frac{RO}{2} - MO) - (t_i - \frac{RT}{2} - MT))^2$$

where y_{mi} is the ith output of the module, RO is the range of values in the output array, MO is the minimum of the output array, t_i is the ith target element, RT is the range of values in the target array, and MT is the minimum of the target array.

The function may be rewritten as

$$V_m = \frac{1}{2}\sum_i(y_{mi} - t_i + K)^2$$

where $K = (\frac{RT}{2} + MT) - (\frac{RO}{2} + MO)$.

The gradient with respect to the inputs of the ith output unit of the mth worker module may be defined as

$$\frac{\partial J}{\partial x_{mi}} = \frac{\partial J}{\partial V_m} \frac{\partial V_m}{\partial y_{mi}} \frac{\partial y_{mi}}{\partial x_{mi}}$$

For the winning worker,

$$\frac{\partial J}{\partial x_{wi}} = \frac{1}{\sum_{j \neq w} V_j}(y_{mi} - t_i + K)y_{mi}(1 - y_{mi})$$

For the remaining workers,

$$\frac{\partial J}{\partial x_{mi}} = \frac{-V_w}{(\sum_{j \neq w} V_j)^2}(y_{mi} - t_i + K)y_{mi}(1 - y_{mi})$$

Monitor modules were treated as before; that is, the target was 1.0 for the winning worker's monitor and 0.0 for each of the others, and all learned at every pattern presentation.

The experiment with *w*, *l*, and *ow* classes was repeated using this training algorithm, and the results showed an improvement. Generalization performance was *w* (89.6%), *l* (70.0%), and *ow* (80.0%). Overall performance of 76.6 percent correct was now about one percent better than that of the single multilayer network (Section 4.1). Although this result is better than that obtained with the original Minos training procedure, it does not appear to be a significant improvement over the performance of the simpler multilayer network.

5.4 Helper Networks

The deterministic, back-propagation learning algorithm used to train multilayer networks is a gradient descent algorithm that may or may not arrive at a global minimum of the objective function. If a local minimum is encountered, some training examples will not be learned and generalization performance may suffer.

If one network is incapable of learning the complete training set, it seems reasonable to freeze its weights and train a second "helper" network to make up for the errors of the first. This may be done by setting the target of the helper equal to the first network's error (i.e., target - output). The helper network should avoid the local minimum encountered by the first network because of both the different initial weights and the different error signals arising from the different target. If other local minima are encountered by the helper network, additional helper networks may be added. The complete system output would be the sum of the outputs of all the networks.

Table 7: Confusion Matrix for *W* Class

Input/Output	W	Not W	Overall
W	.88	.12	
Not W	.08	.92	
			.88

Table 8: Confusion Matrix for *L* Class

Input/Output	L	Not L	Overall
L	.89	.11	
Not L	.09	.91	
			.89

For a system of K networks with each network having only one output unit, these considerations resulted in the following objective function and error derivative for each network.

$$E_{pk} = \frac{1}{2}\sum_{k}^{K-1}(t_p - \sum_{i=0}^{k} o_{pi})^2 \qquad (6)$$

$$\frac{\partial E_{pk}}{\partial O_k} = -\sum_{m=k}^{K-1}(t_p - \sum_{i=0}^{m} o_{pi})$$

Since the helper networks' targets can be either positive or negative, their outputs were generated with either the linear activation function or the symmetric logistic (-1.0 to 1.0) activation function.

In an experiment with this architecture, three separate systems of two networks each were trained to respond to the presence of *w*, *l*, and *ow* classes, respectively. Each network had 224 inputs units, 10 hidden units, and 1 output unit. The desired output of each system was 1.0 to signal the presence of the class, and 0.0 to signal its absence (i.e., the presence of any other class). In tests of the systems, 0.5 was chosen as the criterion to separate "on" versus "off" responses. Tables 7, 8 and 9 show the confusion matrices that resulted.

The tables show that the systems of networks discriminate quite highly. However, roughly the same performance was obtained by using only one of the networks (i.e., no helper), and a slight but consistent improvement was obtained using a single larger network with as

Table 9: Confusion Matrix for *OW* Class

Input/Output	OW	Not OW	Overall
OW	.84	.16	
Not OW	.07	.93	
			.84

many weights as the two networks combined. Apparently, no advantage was gained with this data set from using the helper architecture.

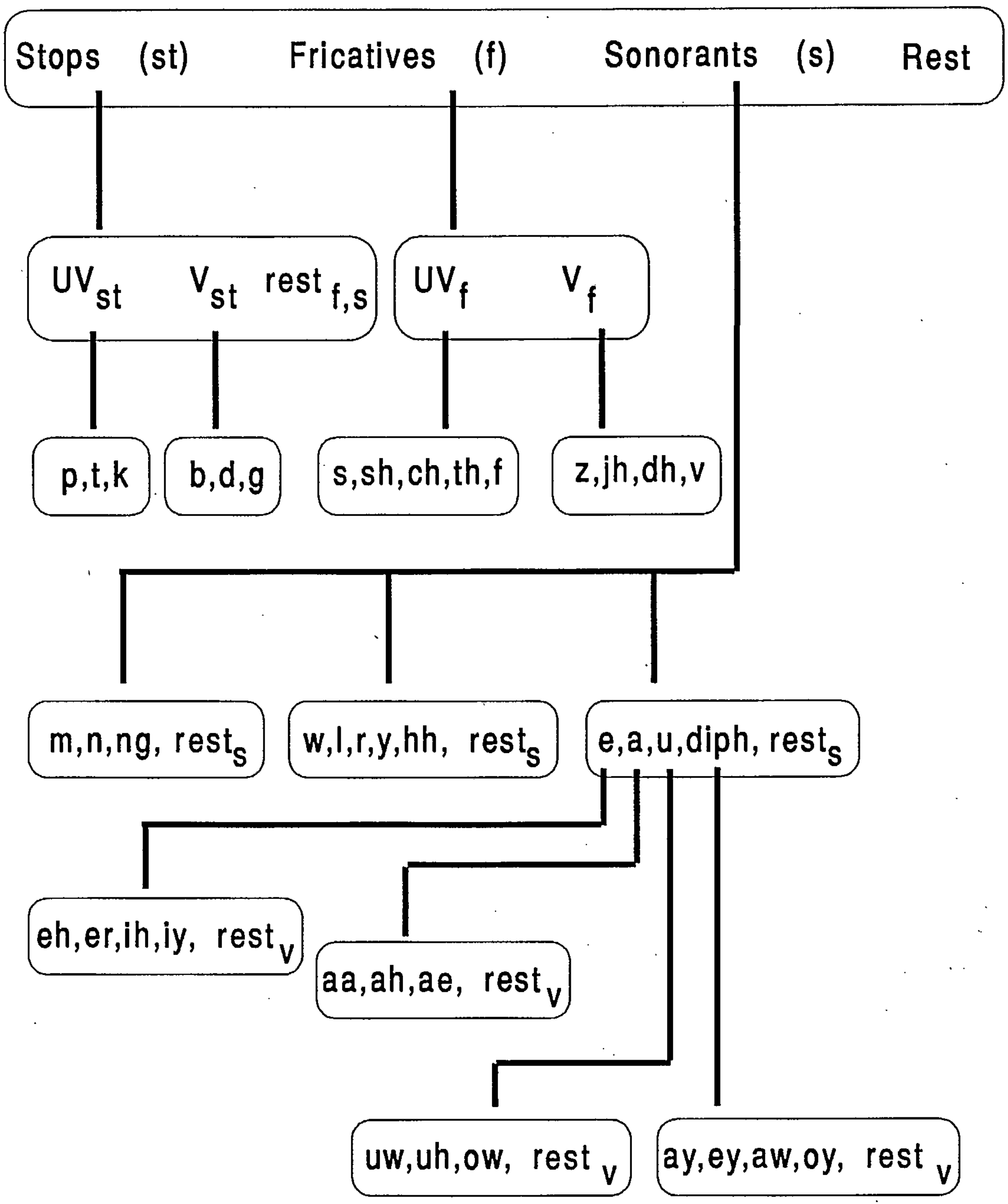
It can be argued that no local minima were encountered during training, so the task was not difficult enough to properly test the helper architecture. Further experiments using data with known characteristics would be of interest. For example, the exclusive-OR problem can not be solved by a network with one layer of weights. However, two such networks, cooperating as described above, may be able to solve it.

5.5 Input Space Transformation Architecture

Table 3 shows that TDNNs are able to map limited regions of the spectrogram space to a phoneme space with greater or lesser degrees of accuracy. It is reasonable to assume that intra-class variability present in the original spectrogram data may be somewhat reduced in the new phoneme space resulting from the transformations. A reduction may be expected because multilayer networks are able to map disjoint regions of input space to the same target. Such reduced variability may enable a single network to classify the complete phoneme set more easily using input data from the various phoneme subspaces rather than from the original spectrogram space. This section describes an experiment which tests this conjecture.

The experiment described in Section 4.1 used a single multilayer network to discriminate among 39 phoneme classes. The confusion matrix provided by that network gave an indication of the similarity among the classes. These observed similarities were used to crudely partition the phoneme space into a number of subspaces so that a given TDNN network module would discriminate only among similar classes. Table 10 lists the discriminations learned by each network, and the domain from which the training set was drawn. Figure 6 shows more clearly that the assignment of input regions to modules was organized somewhat hierarchically. Each network was trained with 300 patterns per class. The word "rest" in the table and the figure refers to a class which groups together the remaining classes in the indicated training set domain.

During training, each network was tested periodically to ensure that generalization accuracy was not decreasing due to overlearning. Training stopped when a decrease was observed.



Note: Subscript 'v' refers to vowels

Figure 6: The hierarchical organization of network domains

Table 10: Domains of Network Modules

Output Dimensions	Training Set Domain
Stops, Fricatives, Sonorant, Rest	(unlimited)
Unvoiced, Voiced Stops, Rest	Stops, Fricatives, Sonorants
Unvoiced, Voiced Fricatives, Rest	Fricatives, Stops, Sonorants
P, T, K	Unvoiced Stops
B, D, G	Voiced Stops
S, SH, CH, TH, F	Unvoiced Fricatives
Z, JH, DH, V	Voiced Fricatives
M, N, NG, Rest	Sonorants
W, L, R, Y, HH, Rest	Sonorants
E, A, U, Diphthongs, Rest	Sonorants
EH, ER, IH, IY, Rest	Vowels
AA, AH, AE, Rest	Vowels
UW, UH, OW, Rest	Vowels
AY, EY, AW, OY, Rest	Vowels

Then a different, validation test set was used to again measure generalization performance, and these results are reported in Table 11.

The outputs of the 14 TDNN modules identified in Table 10 map the voice spectrogram space into a phoneme space. These outputs (except for some of the "rest" categories) became training patterns for a single multilayer network with 51 inputs, 30 hidden units, and 38 output units[10] That is, each pattern in the spectrogram space was transformed by all the TDNN modules into a new input pattern in phoneme space. Figure 7 shows the structure of the complete classifier architecture. The multilayer network was trained with 150 patterns per class[11]. After 8500 passes through the training set, a generalization accuracy of 52.4 percent correct classifications over both test sets was achieved. Table 12 lists the 38 classes, with associated frequencies and proportions correct for each test set alone and for the combined test sets.

At first glance, the above results appear to be an improvement over those obtained in Section 4.1 with the simple multilayer network structure. Notably, however, the largest improvement is for the *silence* class which is also associated with the highest frequency. When this class is ignored in the generalization tests, accuracies of 48.2 percent and 45.8 percent were obtained from the multilayer network and the above modular network, respectively. Thus, the modular network appears to make fewer errors than the multilayer network in identifying the *silence* class, but has slightly more difficulty identifying the other classes.

[10]The number of classes was reduced by one from the earlier experiment (Section 4.1) by merging classes *d* and *dx*. Also, *ao* was merged with *aw* instead of *aa*.

[11]the maximum number allowed by the available memory

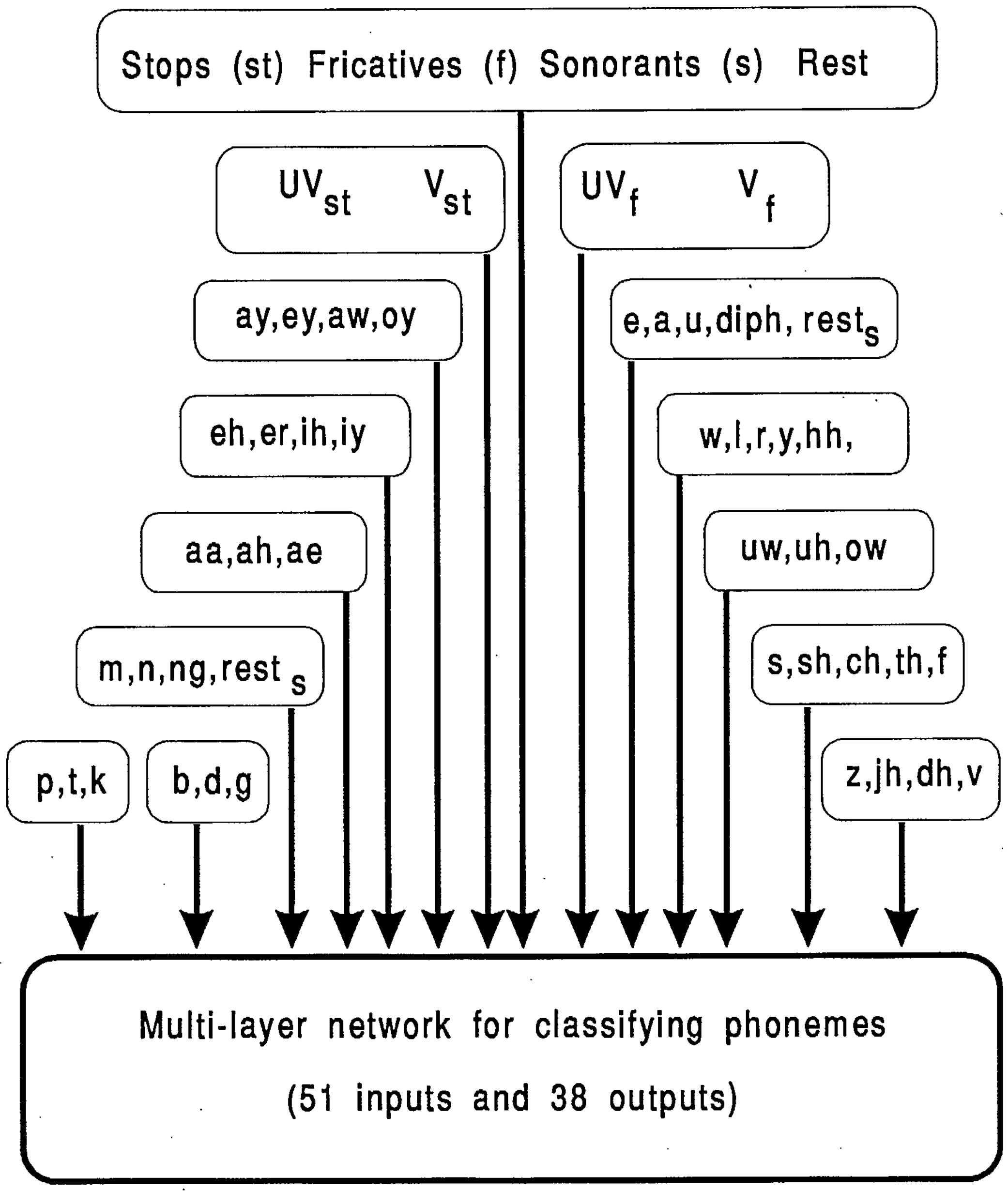


Figure 7: The input space transformation architecture

Table 11: Generalization Test Results

Output Dimensions	Percent Correct Generalization
Stops, Fricatives, Sonorant, Rest	78.7, 78.0, 87.6
Unvoiced, Voiced Stops, Rest	88.4, 73.7, 69.0
Unvoiced and Voiced Fricatives	94.9, 75.0
P, T, K	82.2, 86.1, 80.7
B, D, G	79.5, 81.4, 83.3
S, SH, CH, TH, F	82.4, 80.6, 100.0, 78.6, 81.0
Z, JH, DH, V	74.1, 83.3 80.4, 78.6
M, N, NG, Rest	61.4, 74.4, 61.1, 65.1
W, L, R, Y, HH, Rest	74.1, 61.6, 82.7, 87.5, 86.4, 46.0
E, A, U, Diphthongs, Rest	51.1, 43.6, 62.5, 55.2, 68.0
EH, ER, IH, IY, Rest	56.1, 82.5, 55.0, 82.0, 52.8
AA, AH, AE, Rest	83.8, 71.6, 77.5 49.9
UW, UH, OW, Rest	84.4, 85.7, 69.2, 43.8
AY, EY, AW, OY, Rest	69.0, 86.4, 100.0, 50.0, 55.2
Mean Percent Correct	77.7

6 Conclusions

The problems with scaling up the size of back-propagation networks has resulted in a number of proposals for the design of modular network systems. The objective of this program of research was to evaluate some of these ideas in the context of the real-world problem of speaker-independent speech recognition.

Although labelling of the data in the TIMIT data base appears to be accurate, intra-class variability is still very large. In particular, both spectral shape and duration is known to vary widely among members of a given phoneme class. In contrast, many interesting theoretical ideas are often tested only with well understood and controlled problems. A solution that appears promising under those conditions may not be very useful under much more variable conditions. Usually, the best way to determine the utility of new theoretical insights is by empirical testing. The exploratory work described in this paper was conducted for this purpose.

The Meta-Pi, system of experts, and Minos architectures all basically attempt to do the same thing; that is, to form criteria useful for identifying regions of the input space. Under the conditions tested here, however, these modular systems could not perform any better than a single network trained to handle the whole input space. This may suggest that the overlap in class distributions was such that clear region boundaries could not be identified by the Meta-Pi module, the system of experts gating module, or Minos monitor modules. Alternatively, the simple network was able to perform optimally with the given data, and

Table 12: Performance of the Input Space Transformation Architecture

Class	Set A Freqency	Set A % Correct	Set B Frequency	Set B % Correct	Combined % Correct
p	50	50.0	64	70.3	61.4
t	78	67.9	72	65.3	66.7
k	94	55.3	88	73.9	64.3
b	46	56.5	44	70.5	63.3
d	72	36.1	102	43.1	40.2
g	18	44.4	6	83.3	54.2
s	128	75.8	102	70.6	73.5
sh	25	68.0	36	72.2	70.5
ch	12	75.0	6	100.0	83.3
th	14	35.7	14	50.0	42.9
f	44	81.8	42	76.2	79.1
z	66	60.6	54	46.3	54.2
jh	24	62.5	30	60.0	61.1
dh	52	36.5	46	47.8	41.8
v	46	32.6	42	45.2	38.6
m	86	33.7	83	39.8	36.7
n	152	30.9	172	28.5	29.6
ng	25	60.0	18	66.7	62.8
w	48	47.9	58	56.9	52.8
l	110	50.9	112	43.8	47.3
r	106	44.3	98	34.7	39.7
y	14	50.0	16	68.8	60.0
hh	40	65.0	22	54.5	61.3
eh	66	15.2	66	7.6	11.4
er	72	65.3	80	61.3	63.2
ix	244	24.6	278	20.5	22.4
iy	94	60.6	100	63.0	61.9
aa	34	41.2	48	54.2	48.8
ax	104	24.0	116	14.7	19.1
ae	50	68.0	40	47.5	58.9
uw	32	46.9	32	59.4	53.1
ow	30	26.7	26	34.6	30.4
uh	4	0.0	14	50.0	38.9
ay	48	47.9	42	45.2	46.7
ey	34	61.8	44	65.9	64.1
aw	46	6.5	24	12.5	8.6
oy	10	100.0	6	33.3	75.0
pau	594	78.6	574	76.5	77.6
OVERALL		52.9		51.8	52.4

no advantage could be gained by using the more complex architectures.

A new modular architecture was developed which did not have a component for identifying the module in which one should have the greatest confidence. Rather, the modules were used to map the input space to a different representation of reduced dimensionality and, presumably, variability. This new representation enabled improved recognition of silence intervals, and performed almost as well as a simple multilayer network classifier at recognizing tokens from the other classes. Because of their similar abilities to identify phoneme tokens, the choice of network to use depends on other factors. The current preference is for the modular network which is composed mostly of TDNN modules, since the TDNN is known to produce less variable output when speech data is passed through its input window.

References

[1] *American standard specification for octave, half-octave, and third-octave band filter sets.* American National Standards Institute, New York, NY, s1.11-1971 edition, 1971.

[2] H. Bourlard and C.J. Wellekens. Links between Markov models and multilayer perceptrons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:1167–1178, 1990.

[3] J.S. Bridle, M.D. Brown, and R.M. Chamberlain. An algorithm for connected word recognition. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 899–902, IEEE, New York, Paris 1982, 1988.

[4] A.C. Clarke. *2001: a space odyssey.* New American Library of Canada, Ltd., Toronto, 1968.

[5] A. Cleeremans, D. Servan-Schreiber, and J.L. McClelland. Finite state automata and simple recurrent networks. *Neural Computation*, 1:372–381, 1989.

[6] S.J. Cox. Hidden Markov models for automatic speech recognition: theory and application. *British Telecom Technical Journal*, 6:105–115, 1988.

[7] J.L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[8] S.E. Fahlman. Fast-learning variations on back-propagation: an empirical study. In D. Touretzky, G. Hinton, and T. Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 38–51, Morgan Kaufmann, San Mateo, Pittsburg 1988, 1988.

[9] M.A. Franzini, K-F. Lee, and A. Waibel. Connectionist Viterbi training: a new hybrid method for continuous speech recognition. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 425–428, IEEE, New York, Albuquerque 1990, 1990.

[10] M.A. Franzini, M.J. Witbrock, and K-F. Lee. Speaker-independent recognition of connected utterances using recurrent and non-recurrent neural networks. In *IEEE*

International Conference on Neural Networks, pages 1–6, IEEE, New York, Washington 1989, 1989.

[11] J.S. Garofolo. *The structure and format of the DARPA TIMIT CD-ROM prototype.* National Institute of Standards and Technology, Gaithersburg, MD, 1988.

[12] J.B. Hampshire and B. Pearlmuter. Equivalence proofs for multi-layer perceptron classifiers and the Bayesian discriminant function. In D.S. Touretzky, J.L. Elman, T.J. Sejnowski, and G. Hinton, editors, *Proceedings of the 1990 Connectionist Models Summer School*, pages 159–172, Morgan Kaufmann, San Mateo, La Jolla, 1990, 1988.

[13] J.B. Hampshire and A.H. Waibel. *The Meta-Pi network: Building distributed knowledge representations for robust pattern recognition.* Technical Report CMU-CS-89-166, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, 1989.

[14] R.A. Heinlein. *The Number of the Beast.* Fawcett Columbine, New York, 1980.

[15] R.A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1:295–307, 1988.

[16] R.A. Jacobs, M.I. Jordan, and A.G. Barto. Task decomposition through competition in a modular connectionist architecture: the what and where vision tasks. *Cognitive Science*, 15:219–249, 1991.

[17] T. Kohonen. The "neural" phonetic typewriter. *Computer*, 21:11–22, 1988.

[18] K.J. Lang and G.E. Hinton. *The development of the time-delay neural network architecture for speech recognition.* Technical Report CMU-CS-88-152, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, 1988.

[19] K.J. Lang, A.H. Waibel, and G.E. Hinton. A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 3:23–43, 1990.

[20] K-F Lee. On large-vocabulary speaker-independent continuous speech recognition. *Speech Communication*, 7:375–379, 1988.

[21] K-F Lee, H-S Hon, and R. Reddy. An overview of the SPHINX speech recognition system. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38:35–45, 1990.

[22] H.C. Leung and V.W. Zue. Some phonetic recognition experiments using artificial neural networks. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 422–425, IEEE, New York, New York 1988, 1988.

[23] S.J. Nowlan. *Competing experts: An experimental investigation of associative mixture models.* Technical Report CRG-TR-90-5, Department of Computer Science, University of Toronto, Toronto, Canada, 1990.

[24] S.J. Nowlan and G.E. Hinton. Evaluation of adaptive mixtures of competing experts. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems*, pages 550–557, Morgan Kaufmann, San Mateo, Denver 1991, 1991.

[25] D.C. Plaut and G.E. Hinton. Learning sets of filters using back-propagation. *Computer Speech and Language*, 2:35–61, 1987.

[26] L.R. Rabiner and B.H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, January:4–16, 1986.

[27] A.J. Robinson and F. Fallside. *Phoneme recognition from the TIMIT database using recurrent error propagation networks.* Technical Report CUED/F-INFENG/TR.42, University of Cambridge, Cambridge, England, 1990.

[28] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In D.E. Rumelhart and J.L. McClelland, editors, *Parallel Distributed Processing*, chapter 8, pages 318–362, MIT Press, Cambridge, 1986.

[29] H. Sawai, A. Waibel, M. Miyatake, and K. Shikano. Spotting Japanese CV-syllables and phonemes using time-delay neural networks. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 25–28, IEEE, New York, Glasgow 1989, 1989.

[30] F.J. Smieja. Multiple network systems (Minos) modules: Task division and module discrimination. In *Proceedings of the 8th AISB90 Conference on Artificial Intelligence*, Leeds 1991, 1991.

[31] K. Torkkola. Automatic alignment of speech with phonetic transcriptions in real time. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 611–614, IEEE, New York, New York 1988, 1988.

[32] W.C. Treurniet, M.J. Hunt, C. Lefevbre, and Z. Jacobson. Phoneme recognition with a neural network: comparisons of acoustic representations including those produced by an auditory model. In *IEEE International Conference on Neural Networks*, page 320, IEEE, New York, San Diego 1988, 1988.

[33] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37:328–339, 1989.

[34] A. Waibel, H. Sawai, and K. Shikano. Modularity and scaling in large phonemic neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37:1888–1898, 1989.

[35] J.G. Wilpon, L.R. Rabiner, and A. Bergh. Speaker-independent isolated words recognition using a 129-word airline vocabulary. *Journal of the Acoustical Society of America*, 72:390–396, 1982.

[36] S.R. Young, A.G. Hauptmann, W.H. Ward, E.T. Smith, and P. Werner. High level knowledge sources in usable speech recognition systems. *Communications of the ACM*, 32:183–194, 1989.

Communications
Research Center
Shirleys Bay
3701 Carling Avenue
P.O. Box 11490
Station H
Ottawa, Ontario
K2H 8S2

Centre de recherches
sur les communications
Shirleys Bay
3701, avenue Carling
Case postale 11490
Succursale H
Ottawa (Ontario)
K2H 8S2