5. Databases

We performed our experiments on NN-HMM hybrids using three different databases: ATR's database of isolated Japanese words, the CMU Conference Registration database, and the DARPA Resource Management database. In this chapter we will briefly describe each of these databases.

5.1. Japanese Isolated Words

Our very first experiments were performed using a database of 5240 isolated Japanese words (Sagisaka et al 1987), provided by ATR Interpreting Telephony Research Laboratory in Japan, with whom we were collaborating. This database includes recordings of all 5240 words by several different native Japanese speakers, all of whom are professional announcers; but our experiments used the data from only one male speaker (MAU). Each isolated word was recorded in a soundproof booth, and digitized at a 12 kHz sampling rate. A Hamming window and an FFT were applied to the input data to produce 16 melscale spectral coefficients every 10 msec.

Because our computational resources were limited at the time, we chose not to use all 5240 words in this database; instead, we extracted two subsets based on a limited number of phonemes:

- **Subset 1** = 299 words (representing 234 unique words, due to the presence of homophones), comprised of only the 7 phonemes a,i,u,o,k,s,sh (plus an eighth phoneme for silence). From these 299 words, we trained on 229 words, and tested on the remaining 70 words (of which 50 were homophones of training samples, and 20 were novel words). Table 5.1 shows this vocabulary.
- **Subset 2** =1078 words (representing 924 unique words), comprised of only the 13 phonemes a,i,u,e,o,k,r,s,t,kk,sh,ts,tt (plus a 14th phoneme for silence). From these 1078 words, we trained on 900 words, and tested on 178 words (of which 118 were homophones of training samples, and 60 were novel words).

Using homophones in the testing set allowed us to test generalization to new samples of known words, while the unique words allowed us to test generalization to novel words (i.e., vocabulary independence).

aa	ikou	ooku	kakoi	ku *	koushou	**sasai**	**shisso**	shousoku
ai	ishi **	oka	kakou *	kui	kousou	sasu **	shakai	shoku
aiso	**ishiki**	**okashii**	kasa	kuiki	kousoku	sasoi	shaku	shokki
au *	isha	okasu	kasai	kuu	kokuso	sasou	shako	su *
ao	ishou	oki	kashi	kuuki	koshi	sakka	shashou *	suisoku
aoi	isu	oku **	kashikoi	kuukou	koshou	sakkaku	shuu	suu *
aka *	ikka	okosu	kashu	kuusou	**koosu**	sakki	**shuui**	sukasu
akai	ikkou	oshii	kasu *	kuki	kosu *	sakku	shuukai	suki *
aki *	issai	oshoku	kasuka	**kusa**	kokka	sassou	shuukaku	suku
aku *	isshu	osu *	**kakki**	kusai	kokkai	sassoku	shuuki	sukuu *
akushu	issho	osoi	kakko	kushi *	kokkaku	shi **	shuusai	sukoshi
asa	isshou	osou	kakkou	ko	kokki	shiai	shuushuu	**sushi**
asai	isso	ka *	ki *	koi **	kokkou	shio *	shuushoku	susu
ashi	issou	kai **	kioku	koishii	sa	shikai *	shukusha	suso
asu	ukai	kaikaku	kikai **	kou *	saiku	shikaku *	shukushou	sou **
akka	uku	kaisai	kikaku *	koui *	saikou	shikashi	shusai	soui
asshuku	ushi	kaishi	kiki	kouka	kaishuu	shiki	shushi	souko
i	**usui**	**kaisha**	kiku **	koukai **	**saisho**	**shikisai**	shushoku	sousa *
ii	uso	kaishaku	kikou	koukou *	saisoku	shiku	shou *	sousaku *
iu	o	kaishou	kisaku	koukoku	sao	shikou *	shouka *	soushiki
ika	oi	kau *	kishi	kousa	saka	shisaku	shoukai	soushoku
iasu	oishii	kao	kisha	**kousai**	sakai	shishuu	shouki	soko
iki **	ou *	kaoku	kishou *	kousaku *	sakasa	shishou	**shouko**	soshi
ikiiki	ooi *	kaku ***	kisuu	koushi *	saki	shisou	shousai	soshiki
ikioi	oou	kakusu	kiso *	koushiki	saku ***	shikkaku	shoushou	**soshou**
iku	ookii	**kako**	**kisoku**	koushuu *	sakusha	shikki	shousuu	sosokkashii

Table 5.1: Japanese isolated word vocabulary (Subset 1 = 299 samples including homophones; 234 unique words). The testing set (70 words) consisted of 50 homophones (starred words) and 20 novel words (in bold).

5.2. Conference Registration

Our first experiments with continuous speech recognition were performed using an early version of the CMU Conference Registration database (Wood 1992). The database consists of 204 English sentences using a vocabulary of 402 words, comprising 12 hypothetical dialogs in the domain of conference registration. A typical dialog is shown in Table 5.2; both sides of the conversation are read by the same speaker. Training and testing versions of this database were recorded with a close-speaking microphone in a quiet office by multiple speakers for speaker-dependent experiments. Recordings were digitized at a sampling rate of 16 kHz; a Hamming window and an FFT were computed, to produce 16 melscale spectral coefficients every 10 msec.

Since there are 402 words in the vocabulary, this database has a perplexity[1] of 402 when testing without a grammar. Since recognition is very difficult under such conditions, we created a word pair grammar (indicating which words can follow which other words) from the textual corpus. Unfortunately, with a perplexity of only 7, this word pair grammar soon proved too easy — it's hard to identify significant improvements above 97% word accuracy.

1. Perplexity is a measure of the branching factor in the grammar, i.e., the number of words that can follow any given word.

A:	Hello, is this the office for the conference?
B:	Yes, that's right.
A:	I would like to register for the conference.
B:	Do you already have a registration form?
A:	No, not yet.
B:	I see. Then I'll send you a registration form.
B:	Could you give me your name and address?
A:	The address is five thousand Forbes Avenue, Pittsburgh, Pennsylvania, one five two three six.
A:	The name is David Johnson.
B:	I see. I'll send you a registration form immediately.
B:	If there are any questions, please ask me at any time.
A:	Thank you. Goodbye.
B:	Goodbye.

Table 5.2: A typical dialog in the Conference Registration database.

Therefore, we usually evaluated recognition accuracy at a perplexity of 111, by testing only the first three dialogs (41 sentences) using a reduced vocabulary without a grammar.

The Conference Registration database was developed in conjunction with the Janus Speech-to-Speech Translation system at CMU (Waibel et al 1991, Osterholtz et al 1992, Woszczyna et al 1994). While a full discussion of Janus is beyond the scope of this thesis, it is worth mentioning here that Janus is designed to automatically translate between two spoken languages (e.g., English and Japanese), so that the above dialog could be carried out between an American who wants to register for a conference in Tokyo but who speaks no Japanese, and a Japanese receptionist who speaks no English. Janus performs speech translation by integrating three modules — speech recognition, text translation, and speech generation — into a single end-to-end system. Each of these modules can use any available technology, and in fact various combinations of connectionist, stochastic, and/or symbolic approaches have been compared over the years. The speech recognition module, for example, was originally implemented by our LPNN, described in Chapter 6 (Waibel et al 1991, Osterholtz et al 1992); but it was later replaced by an LVQ-based speech recognizer with higher accuracy. Most recently, Janus has been expanded to a wide range of source and destination languages (English, Japanese, German, Spanish, Korean, etc.); its task has broadened from simple read speech to arbitrary spontaneous speech; and its domain has changed from conference registration to appointment scheduling (Woszczyna et al 1994).

5.3. Resource Management

In order to fairly compare our results against those of researchers outside of CMU, we also ran experiments on the DARPA speaker-independent Resource Management database (Price et al 1988). This is a standard database consisting of 3990 training sentences in the domain of naval resource management, recorded by 109 speakers contributing roughly 36 sentences each; this training set has been supplemented by periodic releases of speaker-independent testing data over the years, for comparative evaluations. Some typical sentences are listed in Table 5.3. The vocabulary consists of 997 words, many of which are easily confusable, such as what/what's/was, four/fourth, any/many, etc., as well as the singular, plural, and possessive forms of many nouns, and an abundance of function words (a, the, of, on, etc.) which are unstressed and poorly articulated. During testing, we normally used a word pair grammar[1], with a perplexity of 60.

ARE THERE TWO CARRIERS IN YELLOW SEA WITH TRAINING RATING MORE THAN C1
HOW MANY NUCLEAR SURFACE SHIPS ARE WITHIN FIFTY NINE MILES OF CONIFER
SET UNIT OF MEASURE TO METRIC
DRAW THE TRACK OF MISHAWAKA
WHAT IS COPELAND'S FUEL LEVEL AND FUEL CAPACITY
WHAT WAS ARKANSAS'S READINESS THE TWENTY NINTH OF JUNE
ADD AN AREA
DOES SASSAFRAS HAVE THE LARGEST FUEL CAPACITY OF ALL SIBERIAN SEA SUBMARINES
WAS MONDAY'S LAST HFDF SENSOR LOCATION FOR THE HAWKBILL IN MOZAMBIQUE CHANNEL
DO ANY SHIPS THAT ARE IN BASS STRAIT HAVE MORE FUEL THAN HER
EDIT THE ALERT INVOLVING AJAX
WHAT SHIPS WENT TO C2 ON EQUIPMENT AFTER TWELVE JULY
WILL THE EISENHOWER'S EQUIPMENT PROBLEM BE FIXED BY TWENTY THREE JANUARY
WHEN DID SHERMAN LAST DOWNGRADE FOR ASUW MISSION AREA
REDRAW FIJI IN LOW RESOLUTION
CLEAR ALL DATA SCREENS
HOW MANY LAMPS CRUISERS ARE IN MOZAMBIQUE CHANNEL
CLEAR THE DISPLAY
WHAT WAS PIGEON'S LOCATION AND ASUW AREA MISSION CODE TWENTY FIVE DECEMBER
DIDN'T ENGLAND ARRIVE AT MANCHESTER YESTERDAY

Table 5.3: Typical sentences from the Resource Management database.

From the training set of 3990 sentences, we normally used 3600 for actual training, and 390 (from other speakers) for cross validation. However, when we performed gender-dependent training, we further subdivided the database into males, with 2590 training and 240 cross validation sentences, and females, with 1060 training and 100 cross validation sentences. The cross validation sentences were used during development, in parallel with the training sentences. Official evaluations were performed using a reserved set of 600 test sentences (390 male and 210 female), representing the union of the Feb89 and Oct89 releases of testing data, contributed by 30 independent speakers.

1. Actually a word-class pair grammar, as all sentences in this database were generated by expanding templates based on word classes.

6. Predictive Networks

Neural networks can be trained to compute smooth, nonlinear, nonparametric functions from any input space to any output space. Two very general types of functions are *prediction* and *classification*, as shown in Figure 6.1. In a predictive network, the inputs are several frames of speech, and the outputs are a prediction of the next frame of speech; by using multiple predictive networks, one for each phone, their prediction errors can be compared, and the one with the least prediction error is considered the best match for that segment of speech. By contrast, in a classification network, the inputs are again several frames of speech, but the outputs directly classify the speech segment into one of the given classes.

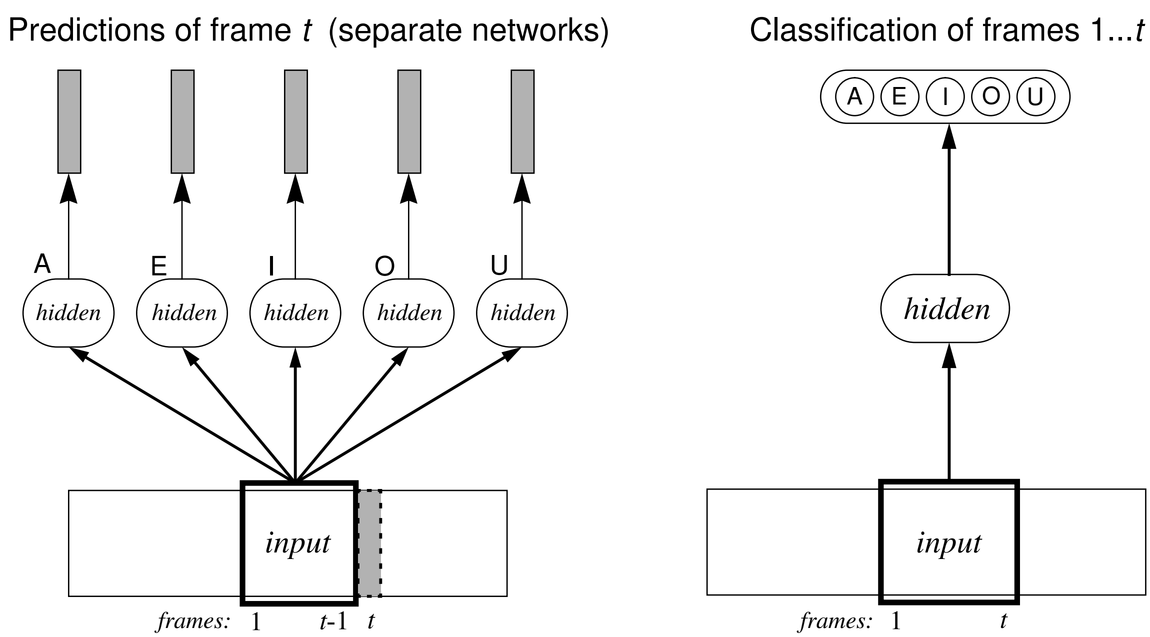

Figure 6.1: Prediction versus Classification.

In the course of our research, we have investigated both of these approaches. Predictive networks will be treated in this chapter, and classification networks will be treated in the next chapter.

6.1. Motivation... and Hindsight

We initially chose to explore predictive networks for a number of reasons. The principal reason was scientific curiosity — all of our colleagues in 1989 were studying classification networks, and we hoped that our novel approach might yield new insights and improved results. On a technical level, we argued that:

1. Classification networks are trained on binary output targets, and therefore they produce quasi-binary outputs, which are nontrivial to integrate into a speech recognition system because binary phoneme-level errors tend to confound word-level hypotheses. By contrast, predictive networks provide a simple way to get non-binary acoustic scores (prediction errors), with straightforward integration into a speech recognition system.
2. The temporal correlation between adjacent frames of speech is explicitly modeled by the predictive approach, but not by the classification approach. Thus, predictive networks offer a dynamical systems approach to speech recognition (Tishby 1990).
3. Predictive networks are *nonlinear* models, which can presumably model the dynamic properties of speech (e.g., curvature) better than linear predictive models.
4. Classification networks yield only one output per class, while predictive networks yield a whole frame of coefficients per class, representing a more detailed acoustic model.
5. The predictive approach uses a separate, independent network for each phoneme class, while the classification approach uses one integrated network. Therefore:
   - With the predictive approach, new phoneme classes can be introduced and trained at any time without impacting the rest of the system. By contrast, if new classes are added to a classification network, the entire system must be retrained.
   - The predictive approach offers more potential for parallelism.

As we gained more experience with predictive networks, however, we gradually realized that each of the above arguments was flawed in some way:

1. The fact that classification networks are trained on binary targets does not imply that such networks yield binary outputs. In fact, in recent years it has become clear that classification networks yield estimates of the posterior probabilities $P(class|input)$, which can be integrated into an HMM more effectively than prediction distortion measures.
2. The temporal correlation between N adjacent frames of speech and the $N+1$st predicted frame is modeled just as well by a classification network that takes $N+1$ adjacent frames of speech as input. It does not matter whether temporal dynamics are modeled explicitly, as in a predictive network, or implicitly, as in a classifica-

tion network.

3. Nonlinearity is a feature of neural networks in general, hence this is not an advantage of predictive networks over classification networks.

4. Although predictive networks yield a whole frame of coefficients per class, these are quickly reduced to a single scalar value (the prediction error) — just as in a classification network. Furthermore, the modeling power of any network can be enhanced by simply adding more hidden units.

5. The fact that the predictive approach uses a separate, independent network for each phoneme class implies that there is no discrimination between classes, hence the predictive approach is inherently weaker than the classification approach. Moreover:

   - There is little practical value to being able to add new phoneme classes without retraining, because phoneme classes normally remain stable for years at a time, and when they are redesigned, the changes tend to be global in scope.
   - The fact that predictive networks have more potential for parallelism is irrelevant if they yield poor word recognition accuracy to begin with.

Unaware that our arguments for predictive networks were specious, we experimented with this approach for two years before concluding that predictive networks are a suboptimal approach to speech recognition. This chapter summarizes the work we performed.

6.2. Related Work

Predictive networks are closely related to a special class of HMMs known as an *autoregressive HMMs* (Rabiner 1989). In an autoregressive HMM, each state is associated not with an emission probability density function, but with an autoregressive function, which is assumed to predict the next frame as a function of some preceding frames, with some residual prediction error (or noise), i.e.:

$$x_t = F_k \left(X_{t-p}^{t-1}, \theta_k \right) + \varepsilon_{t,k} \tag{62}$$

where F_k is the autoregressive function for state k, X_{t-p}^{t-1} are the p frames before time t, θ_k are the trainable parameters of the function F_k, and $\varepsilon_{t,k}$ is the prediction error of state k at time t. It is further assumed that $\varepsilon_{t,k}$ is an independent and identically distributed (iid) random variable with probability density function $p_\varepsilon(\varepsilon|\lambda_k)$ with parameters λ_k and zero mean, typically represented by a gaussian distribution. It can be shown that

$$
\begin{aligned}
P(X_1^T, Q_1^T) &\approx P(X_{p+1}^T, Q_{p+1}^T \,|\, X_1^p, Q_1^p) \\
&= \prod_{t=p+1}^{T} p_\varepsilon(x_t - F_{k_t}(X_{t-p}^{t-1}, \theta_{k_t}) \,|\, \lambda_{k_t}) \cdot p(q_t | q_{t-1})
\end{aligned}
\tag{63}
$$

This says that the likelihood of generating the utterance X_1^T along state path Q_1^T is approximated by the cumulative product of the prediction error probability (rather than the emission probability) and the transition probability, over all time frames. It can further be shown that during recognition, maximizing the joint likelihood $P(X_1^T, Q_1^T)$ is equivalent to minimizing the cumulative prediction error, which can be performed simply by applying standard DTW to the local prediction errors

$$
\left\| x_t - F_k(X_{t-p}^{t-1}, \theta_k) \right\|^2 \tag{64}
$$

Although autoregressive HMMs are theoretically attractive, they have never performed as well as standard HMMs (de La Noue et al 1989, Wellekens 1987), for reasons that remain unclear. Predictive networks might be expected to perform somewhat better than autoregressive HMMs, because they use nonlinear rather than linear prediction. Nevertheless, as will be shown, the performance of our predictive networks was likewise disappointing.

At the same time that we began our experiments, similar experiments were performed on a smaller scale by Iso & Watanabe (1990) and Levin (1990). Each of these researchers applied predictive networks to the simple task of digit recognition, with encouraging results. Iso & Watanabe used 10 word models composed of typically 11 states (i.e., predictors) per word; after training on five samples of each Japanese digit from 107 speakers, their system achieved 99.8% digit recognition accuracy (or 0.2% error) on testing data. They also confirmed that their nonlinear predictors outperformed linear predictors (0.9% error), as well as DTW with multiple templates (1.1% error).

Levin (1990) studied a variant of the predictive approach, called a *Hidden Control Neural Network*, in which all the states of a word were collapsed into a single predictor, modulated by an input signal that represented the state. Applying the HCNN to 8-state word models, she obtained 99.3% digit recognition accuracy on multi-speaker testing. Note that both Levin's experiments and Iso & Watanabe's experiments used non-shared models, as they focused on small vocabulary recognition. We also note that digit recognition is a particularly easy task.

In later work, Iso & Watanabe (1991) improved their system by the use of backward prediction, shared demisyllable models, and covariance matrices, with which they obtained 97.6% word accuracy on a speaker-dependent, isolated word, 5000 Japanese word recognition task. Mellouk and Gallinari (1993) addressed the discriminative problems of predictive networks; their work will be discussed later in this chapter.

6.3. Linked Predictive Neural Networks

We explored the use of predictive networks as acoustic models in an architecture that we called *Linked Predictive Neural Networks* (LPNN), which was designed for large vocabulary recognition of both isolated words and continuous speech. Since it was designed for large vocabulary recognition, it was based on shared phoneme models, i.e., phoneme models (represented by predictive neural networks) that were linked over different contexts — hence the name.

In this section we will describe the basic operation and training of the LPNN, followed by the experiments that we performed with isolated word recognition and continuous speech recognition.

6.3.1. Basic Operation

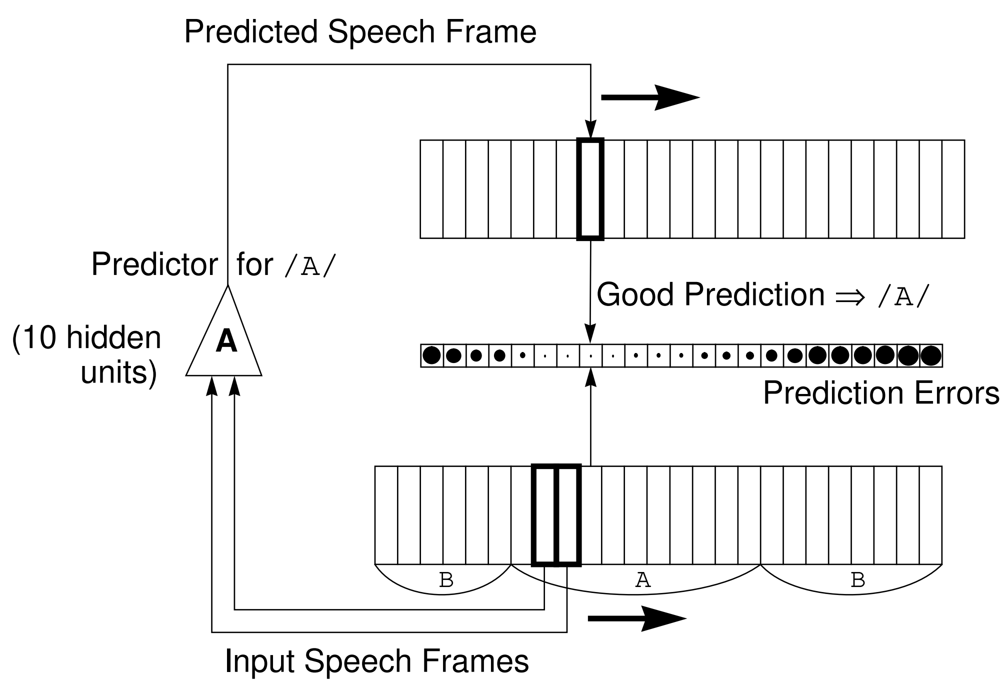

Figure 6.2: Basic operation of a predictive network.

An LPNN performs phoneme recognition via prediction, as shown in Figure 6.2(a). A network, shown as a triangle, takes K contiguous frames of speech (we normally used K=2), passes these through a hidden layer of units, and attempts to predict the next frame of speech. The predicted frame is then compared to the actual frame. If the error is small, the network is considered to be a good model for that segment of speech. If one could teach the network to make accurate predictions only during segments corresponding to the phoneme /A/ (for instance) and poor predictions elsewhere, then one would have an effective /A/ phoneme recognizer, by virtue of its contrast with other phoneme models. The LPNN satisfies this condition, by means of its training algorithm, so that we obtain a collection of phoneme recognizers, with one model per phoneme.

The LPNN is a NN-HMM hybrid, which means that acoustic modeling is performed by the predictive networks, while temporal modeling is performed by an HMM. This implies that the LPNN is a state-based system, such that each predictive network corresponds to a state in an (autoregressive) HMM. As in an HMM, phonemes can be modeled with finer granularity, using sub-phonetic state models. We normally used three states (predictive networks) per phoneme, as shown in subsequent diagrams. Also, as in an HMM, states (predictive networks) are sequenced hierarchically into words and sentences, following the constraints of a dictionary and a grammar.

6.3.2. Training the LPNN

Training the LPNN on an utterance proceeds in three steps: a forward pass, an alignment step, and a backward pass. The first two steps identify an optimal alignment between the acoustic models and the speech signal (if the utterance has been presegmented at the state level, then these two steps are unnecessary); this alignment is then used to force specialization in the acoustic models during the backward pass. We now describe the training algorithm in detail.

The first step is the forward pass, illustrated in Figure 6.3(a). For each frame of input speech at time *t*, we feed frame(*t-1*) and frame(*t-2*) in parallel into all the networks which are linked into this utterance, for example the networks a_1, a_2, a_3, b_1, b_2, and b_3 for the utterance "aba". Each network makes a prediction of frame(*t*), and its Euclidean distance from the actual frame(*t*) is computed. These scalar errors are broadcast and sequenced according to the known pronunciation of the utterance, and stored in column(*t*) of a prediction error matrix. This is repeated for each frame until the entire matrix has been computed.

The second step is the time alignment step, illustrated in Figure 6.3(b). The standard Dynamic Time Warping algorithm (DTW) is used to find an optimal alignment between the speech signal and the phoneme models, identified by a monotonically advancing diagonal path through the prediction error matrix, such that this path has the lowest possible cumulative error. The constraint of monotonicity ensures the proper sequencing of networks, corresponding to the progression of phonemes in the utterance.

The final step of training is the backward pass, illustrated in Figure 6.3(c). In this step, we backpropagate error at each point along the alignment path. In other words, for each frame we propagate error backwards into a single network, namely the one which best predicted that frame according to the alignment path; its backpropagated error is simply the difference between this network's prediction and the actual frame. A series of frames may backpropagate error into the same network, as shown. Error is accumulated in the networks until the last frame of the utterance, at which time all the weights are updated.

This completes the training for a single utterance. The same algorithm is repeated for all the utterances in the training set.

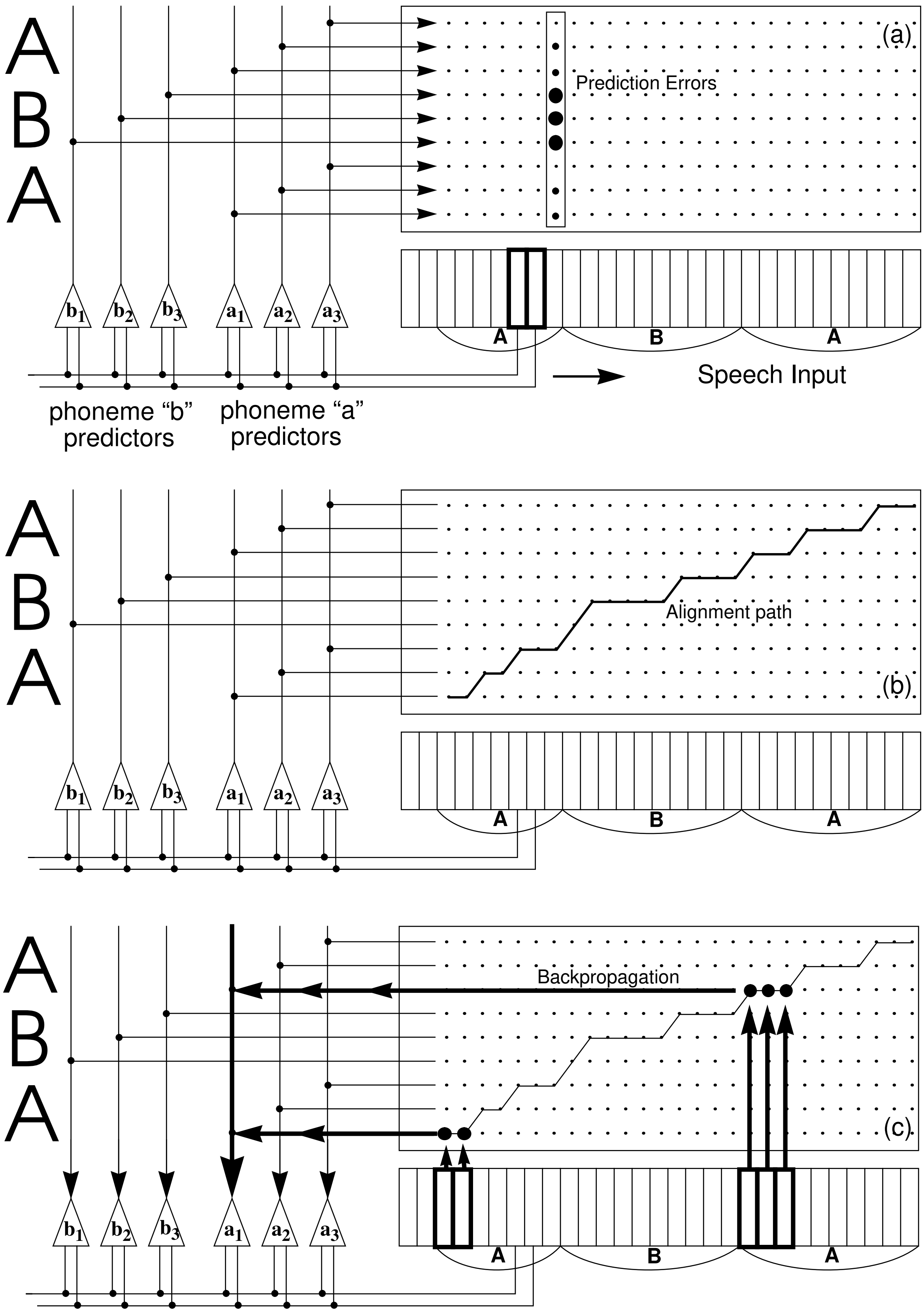

Figure 6.3: The LPNN training algorithm: (a) forward pass, (b) alignment, (c) backward pass.

It can be seen that by backpropagating error from different segments of speech into different networks, the networks learn to specialize on their associated segments of speech; consequently we obtain a full repertoire of individual phoneme models. This individuation in turn improves the accuracy of future alignments, in a self-correcting cycle. During the first iteration of training, when the weights have random values, it has proven useful to force an initial alignment based on average phoneme durations. During subsequent iterations, the LPNN itself segments the speech on the basis of the increasingly accurate alignments.

Testing is performed by applying standard DTW to the prediction errors for an unknown utterance. For isolated word recognition, this involves computing the DTW alignment path for all words in the vocabulary, and finding the word with the lowest score; if desired, next-best matches can be determined just by comparing scores. For continuous speech recognition, the One-Stage DTW algorithm (Ney 1984) is used to find the sequence of words with the lowest score; if desired, next-best sentences can be determined by using the N-best search algorithm (Schwartz and Chow 1990).

6.3.3. Isolated Word Recognition Experiments

We first evaluated the LPNN system on the task of isolated word recognition. While performing these experiments we explored a number of extensions to the basic LPNN system. Two simple extensions were quickly found to improve the system's performance, hence they were adopted as "standard" extensions, and used in all the experiments reported here.

The first standard extension was the use of duration constraints. We applied two types of duration constraints during recognition: 1) hard constraints, where any candidate word whose average duration differed by more than 20% from the given sample was rejected; and 2) soft constraints, where the optimal alignment score of a candidate word was penalized for discrepancies between the alignment-determined durations of its constituent phonemes and the known average duration of those same phonemes.

The second standard extension was a simple heuristic to sharpen word boundaries. For convenience, we include a "silence" phoneme in all our phoneme sets; this phoneme is linked in at the beginning and end of each isolated word, representing the background silence. Word boundaries were sharpened by artificially penalizing the prediction error for this "silence" phoneme whenever the signal exceeded the background noise level.

Our experiments were carried out on two different subsets of a Japanese database of isolated words, as described in Section 5.1. The first group contained almost 300 samples representing 234 unique words (limited to 8 particular phonemes), and the second contained 1078 samples representing 924 unique words (limited to 14 particular phonemes). Each of these groups was divided into training and testing sets; and the testing sets included both homophones of training samples (enabling us to test generalization to new samples of known words), and novel words (enabling us to test vocabulary independent generalization).

Our initial experiments on the 234 word vocabulary used a three-network model for each of the eight phonemes. After training for 200 iterations, recognition performance was perfect for the 20 novel words, and 45/50 (90%) correct for the homophones in the testing set. The fact that novel words were recognized better than new samples of familiar words is due to the fact that most homophones are short confusable words (e.g., “kau” vs. “kao”, or “kooshi” vs. “koshi”). By way of comparison, the recognition rate was 95% for the training set.

We then introduced further extensions to the system. The first of these was to allow a limited number of “alternate” models for each phoneme. Since phonemes have different characteristics in different contexts, the LPNN’s phoneme modeling accuracy can be improved if independent networks are allocated for each type of context to be modeled. Alternates are thus analogous to context-dependent models. However, rather than assigning an explicit context for each alternate model, we let the system itself decide which alternate to use in a given context, by trying each alternate and linking in whichever one yields the lowest alignment score. When errors are backpropagated, the “winning” alternate is reinforced with backpropagated error in that context, while competing alternates remain unchanged.

We evaluated networks with as many as three alternate models per phoneme. As we expected, the alternates successfully distributed themselves over different contexts. For example, the three “k” alternates became specialized for the context of an initial “ki”, other initial “k”s, and internal “k”s, respectively. We found that the addition of more alternates consistently improves performance on training data, as a result of crisper internal representations, but generalization to the test set eventually deteriorates as the amount of training data per alternate diminishes. The use of two alternates was generally found to be the best compromise between these competing factors.

Significant improvements were also obtained by expanding the set of phoneme models to explicitly represent consonants that in Japanese are only distinguishable by the duration of their stop closure (e.g., “k” versus “kk”). However, allocating new phoneme models to represent diphthongs (e.g., “au”) did not improve results, presumably due to insufficient training data.

Table 6.1 shows the recognition performance of our two best LPNNs, for the 234 and 924 word vocabularies, respectively. Both of these LPNNs used all of the above optimizations. Their performance is shown for a range of ranks, where a rank of K means a word is considered correctly recognized if it appears among the best K candidates.

Vocab size	Rank	Testing set		Training set
		Homophones	Novel words	
234	1	47/50 (94%)	19/20 (95%)	228/229 (99%)
	2	49/50 (98%)	20/20 (100%)	229/229 (100%)
	3	50/50 (100%)	20/20 (100%)	229/229 (100%)
924	1	106/118 (90%)	55/60 (92%)	855/900 (95%)
	2	116/118 (98%)	58/60 (97%)	886/900 (98%)
	3	117/118 (99%)	60/60 (100%)	891/900 (99%)

Table 6.1: LPNN performance on isolated word recognition.

For the 234 word vocabulary, we achieved an overall recognition rate of 94% on test data using an exact match criterion, or 99% or 100% recognition within the top two or three candidates, respectively. For the 924 word vocabulary, our best results on the test data were 90% using an exact match criterion, or 97.7% or 99.4% recognition within the top two or three candidates, respectively. Among all the errors made for the 924 word vocabulary (on both training and testing sets), approximately 15% were due to duration problems, such as confusing "sei" and "seii"; another 12% were due to confusing "t" with "k", as in "tariru" and "kariru"; and another 11% were due to missing or inserted "r" phonemes, such as "sureru" versus "sueru". The systematicity of these errors leads us to believe that with more research, recognition could have been further improved by better duration constraints and other enhancements.

6.3.4. Continuous Speech Recognition Experiments

We next evaluated the LPNN system on the task of continuous speech recognition. For these experiments we used the CMU Conference Registration database, consisting of 200 English sentences using a vocabulary of 400 words, comprising 12 dialogs in the domain of conference registration, as described in Section 5.2.

In these experiments we used 40 context-independent phoneme models (including one for silence), each of which had the topology shown in Figure 6.4. In this topology, similar to the one used in the SPICOS system (Ney & Noll 1988), a phoneme model consists of 6 states, economically implemented by 3 networks covering 2 states each, with self-loops and a certain amount of state-skipping allowed. This arrangement of states and transitions provides a tight temporal framework for stationary and temporally well structured phones, as well as sufficient flexibility for highly variable phones. Because the average duration of a phoneme is about 6 frames, we imposed transition penalties to encourage the alignment path to go straight through the 6-state model. Transition penalties were set to the following values: zero for moving to the next state, *s* for remaining in a state, and *2s* for skipping a state, where *s* was the average frame prediction error. Hence 120 neural networks were evaluated during each frame of speech. These predictors were given contextual inputs from two past-frames as well as two future frames. Each network had 12 hidden units, and used sparse connectivity, since experiments showed that accuracy was unaffected while computation could be significantly reduced. The entire LPNN system had 41,760 free parameters.

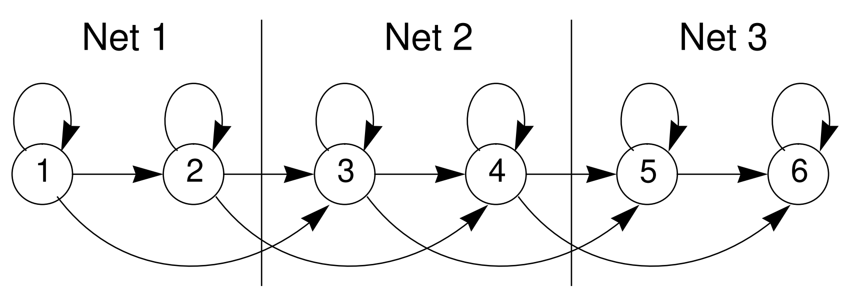

Figure 6.4: The LPNN phoneme model for continuous speech.

Since our database is not phonetically balanced, we normalized the learning rate for different networks by the relative frequency of the phonemes in the training set. During training the system was bootstrapped for one iteration using forced phoneme boundaries, and thereafter trained for 30 iterations using only "loose" word boundaries located by dithering the word boundaries obtained from an automatic labeling procedure (based on Sphinx), in order to optimize those word boundaries for the LPNN system.

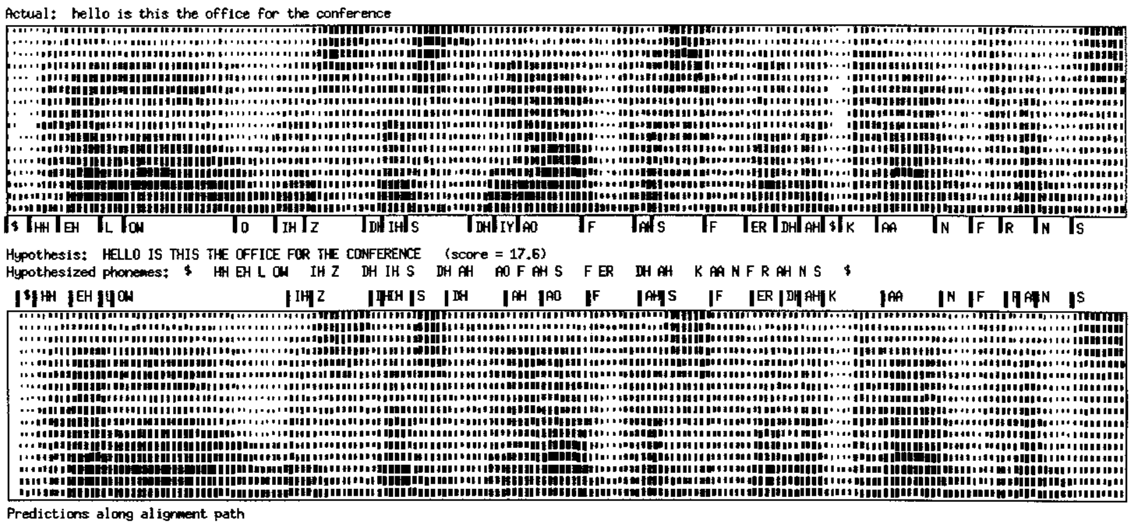

Figure 6.5: Actual and predicted spectrograms.

Figure 6.5 shows the result of testing the LPNN system on a typical sentence. The top portion is the actual spectrogram for this utterance; the bottom portion shows the frame-by-frame predictions made by the networks specified by each point along the optimal alignment path. The similarity of these two spectrograms indicates that the hypothesis forms a good acoustic model of the unknown utterance (in fact the hypothesis was correct in this case).

Speaker-dependent experiments were performed under the above conditions on two male speakers, using various task perplexities (7, 111, and 402). Results are summarized in Table 6.2.

	Speaker A			Speaker B		
Perplexity	7	111	402	7	111	402
Substitutions	1%	28%	43%	4%	28%	46%
Deletions	1%	8%	10%	2%	12%	14%
Insertions	1%	4%	6%	0%	4%	3%
Word Accuracy	97%	60%	41%	94%	56%	37%

Table 6.2: LPNN performance on continuous speech.

6.3.5. Comparison with HMMs

We compared the performance of our LPNN to several simple HMMs, to evaluate the benefit of the predictive networks. First we studied an HMM with only a single Gaussian density function per state, which we parameterized in three different ways:

$M_{16}V_0$: Mean has 16 coefficients; variance is ignored (assumed unity).

$M_{16}V_{16}$: Mean has 16 coefficients; variance has 16 coefficients.

$M_{32}V_0$: Mean has 32 coefficients (including deltas); variance is ignored.

The Gaussian means and variances in each case were derived analytically from the training data. Table 6.3 shows the results of these experiments. It can be seen that the last configuration gave the best results, but the LPNN outperformed all of these simple HMMs.

System	HMM-1 mixture			LPNN
	$M_{16}V_0$	$M_{16}V_{16}$	$M_{32}V_0$	
Substitutions	41%	35%	30%	28%
Deletions	12%	16%	13%	8%
Insertions	10%	5%	2%	4%
Word Accuracy	37%	44%	55%	60%

Table 6.3: Performance of HMMs using a single gaussian mixture, vs. LPNN.

Next we increased the number of mixture densities, from 1 to 5 to 10, where each of the Gaussians was parameterized as in $M_{32}V_0$ above, and evaluated each of these HMMs. We also compared these results against a discriminative LVQ based system developed by Otto Schmidbauer (1992), in which Learned Vector Quantization is used to automatically cluster speech frames into a set of acoustic features, which are subsequently fed into a set of neural network output units which compute the emission probability for HMM states. The results of this comparison are shown in Table 6.4. We see that an LPNN is easily outperformed by an HMM with 5 or more mixture densities, and the discriminative LVQ system outperforms everything. We attribute the inferior performance of the LPNN primarily to its lack of discrimination; this issue will be discussed in detail at the end of this chapter.

	perplexity		
System	7	111	402
HMM-1		55%	
HMM-5	96%	70%	58%
HMM-10	97%	75%	66%
LVQ	98%	80%	74%
LPNN	97%	60%	40%

Table 6.4: Word accuracy of HMM-*n* with *n* mixture densities, LVQ, and LPNN.

Finally, we measured the frame distortion rate of each of the above systems. In an LPNN, frame distortion corresponds to the prediction error. In an HMM, it corresponds to the distance between the speech vector and the mean of the closest gaussian in the mixture. In the LVQ system, it corresponds to the quantization error, i.e., the distance between the input vector and the nearest weight vector of any hidden node. Table 6.5 shows that the LPNN has the least distortion rate of any of these systems, despite its inferior word accuracy. This suggests that the training criterion, which explicitly minimizes the frame distortion rate, is inconsistent and poorly correlated with the ultimate goal of word recognition accuracy. We will further discuss the issue of consistency in the next chapter (Section 7.4).

System	Avg. Frame Distortion
HMM-1	0.15
HMM-5	0.10
HMM-10	0.09
LVQ	0.11
LPNN	0.07

Table 6.5: The LPNN has minimal frame distortion, despite its inferior word accuracy.

6.4. Extensions

In our attempts to improve the accuracy of our LPNN system, we investigated several extensions to the basic system. This section describes those architectural extensions, and presents the results of those experiments.

6.4.1. Hidden Control Neural Network

A common theme in speech recognition systems is to balance the number of free parameters against the amount of training data available, in order to optimize accuracy on the test set. If there are too many free parameters, the system may learn to perfectly memorize the training set, but will generalize poorly to new data. On the other hand, if there are too few free parameters, the system will learn only the coarse characteristics of the task, and so will attain poor accuracy on both the training set and the testing set. Striking an optimal balance always involves sharing parameters to some extent. In a pure HMM system, this can mean sharing codebooks across all states, sharing distributions across states within a phoneme, merging triphones into generalized triphones, merging distributions via senones, and so on. In a NN-HMM hybrid, many of these techniques can still be used; for example, in the last section we described a 6-state phoneme model that uses only 3 networks, sharing distributions across states.

Another way to share parameters, which is unique to neural networks, is to collapse multiple networks into a single network, modulated by a "hidden control" input signal that distinguishes between the functionality of the separate networks. This idea was initially proposed by Levin (1990), and called a *Hidden Control Neural Network* (HCNN). In the context of speech recognition, this involves collapsing multiple predictive networks into a shared HCNN network, modulated by a hidden control signal that distinguishes between the states, as shown in Figure 6.6. The control signal typically uses a simple *thermometer representation*, comprised of one unit for each state, where successive states are represented by turning on successive units, ensuring a similar representation for adjacent states. In addition to reducing the number of parameters (and thus the amount of memory required), the HCNN can also be computationally more efficient than a set of separate networks, since partial results of redundant forward passes can be cached (although the total number of forward passes remains unchanged).

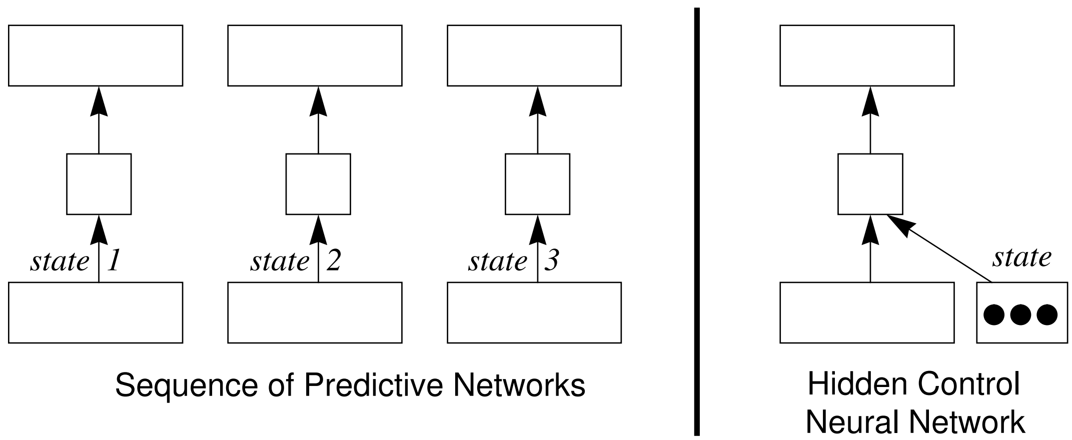

Figure 6.6: A sequence of predictive networks can be replaced by a Hidden Control Neural Network.

We performed a number of experiments with the HCNN, in collaboration with Bojan Petek (1991, 1992). In one set of experiments, we studied the effects of varying degrees of shared structure, as shown in Figure 6.7. These experiments used 2-state phoneme models, rather than 3- or 6-state phoneme models. The first architecture, labeled (a), was a basic LPNN (i.e., no hidden control), in which 80 networks are required to represent 40 phonemes with 2 states each. In (b), hidden control inputs were introduced such that only 40 networks are required for the same task, as each phoneme is modeled by a single network modulated by 2 hidden control input bits which distinguish between the two states. In (c), the hidden control idea is taken to its limit: one big network is modulated by 40 x 2 = 80 hidden control inputs which specify both the phoneme and the state.

Table 6.6 shows the results of these experiments, evaluated on speaker B. Besides testing continuous speech recognition, we also tested excerpted word recognition, in which word boundaries within continuous speech are given; this allowed us to compare the acoustic discriminability of the three architectures more directly. As the table shows, we observed minor differences in performance between architectures (a) and (b): the LPNN was slightly more discriminant, but the hidden control architecture generalized better and ran faster. Meanwhile, architecture (c) did very poorly, presumably because it had too much shared structure and too few free parameters, overloading the network and causing poor discrimi-

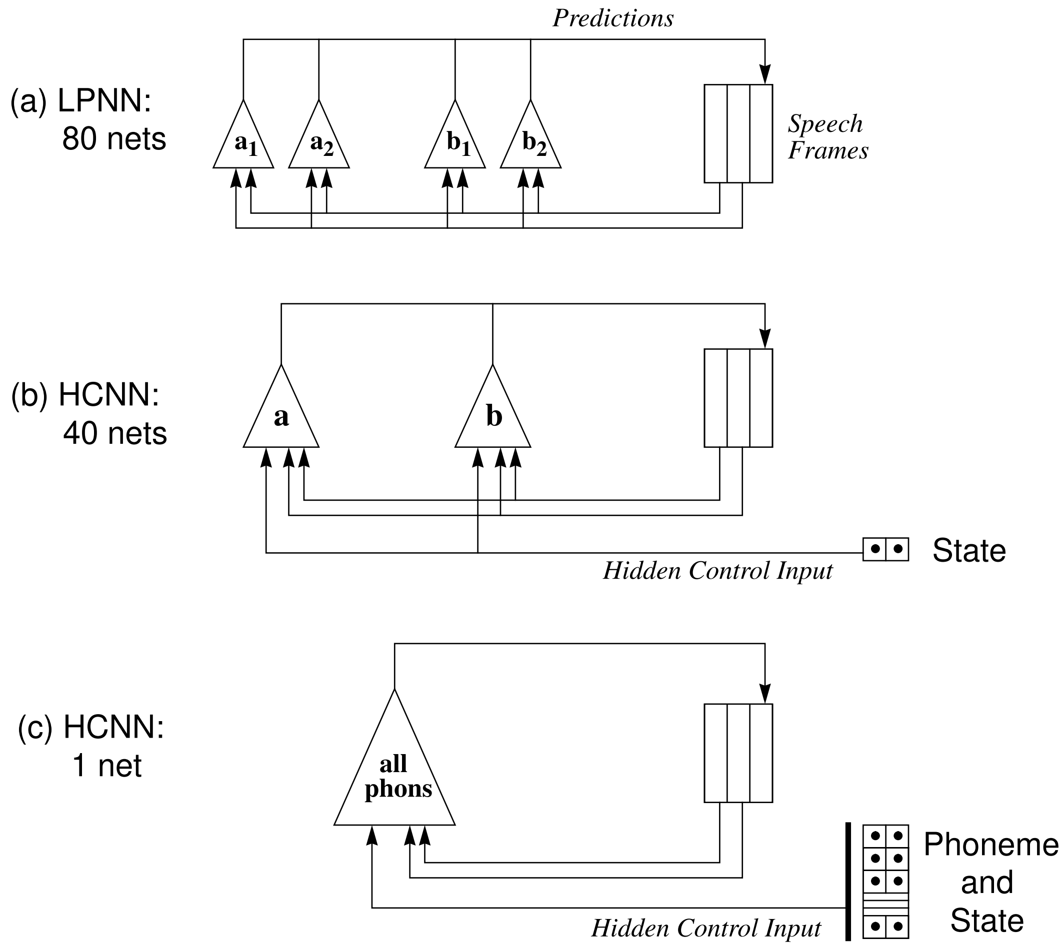

Figure 6.7: Architectures used in Hidden Control experiments.

nation. Hence, we conclude that hidden control may be useful, but care must be taken to find the optimal amount of parameter sharing for a given task.

Architecture	(a)	(b)	(c)
# free parameters (weights)	80960	42080	6466
Word accuracy:			
Excerpted words (P=402)	70%	67%	39%
Continuous speech (P=7)	91%	91%	n/a
Continuous speech (P=402)	14%	20%	n/a

Table 6.6: Results of Hidden Control experiments. Parameter sharing must be used sparingly.

6.4.2. Context Dependent Phoneme Models

The accuracy of any speech recognizer can be improved by using context dependent models. In a pure HMM system, this normally involves using diphone or triphone models, i.e., a phoneme model in the context of one or both of its adjacent phonemes. This increases the specificity and accuracy of the models, but also increases their absolute number by orders of magnitude (e.g., from 40 to 1600 to 64000 models), such that it becomes necessary to cluster them (another form of parameter sharing), to ensure that there is still enough training data per model.

In a NN-HMM hybrid based on predictive networks, context dependent phoneme models could be implemented by using a separate network for each diphone or triphone. However, this would undoubtedly result in a system with too many free parameters, resulting in poor generalization (and an excessive amount of memory and computation). What is desired is again some form of parameter sharing. One potential solution is to use a shared network in which the context is a part of the input signal, as in the HCNN. This approach is appealing because it requires very few additional parameters (diphones require only *c* extra inputs, and triphones require only *2c* extra inputs, for some small value of *c*), and yet it provides a way to distinguish between all the different contexts of a phoneme.

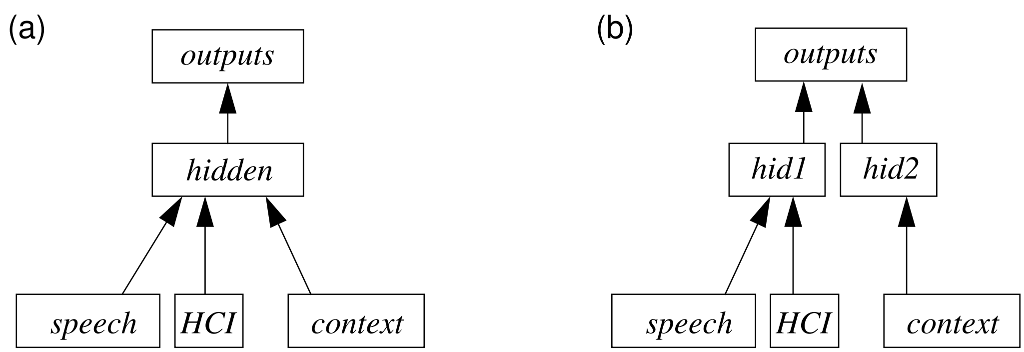

Figure 6.8: Context-dependent HCNN: (a) standard implementation; (b) efficient implementation.

We studied this idea by augmenting our HCNN network to include contextual inputs, as shown in Figure 6.8(a). Our contextual inputs represented only one adjacent phoneme, making this a right-context dependent diphone model (we felt that our database was too small to provide adequate coverage of triphones). We could have represented the 40 possible values of the adjacent phoneme using 40 contextual inputs, but instead we clustered the phonemes by their linguistic features, as proposed by (Rumelhart & McClelland 1986: chapter 18), so that only 10 contextual inputs were necessary. Each phoneme was coded along four dimensions. The first dimension (three bits) was used to divide the phonemes into interrupted consonants (stops and nasals), continuous consonants (fricatives, liquids, and semivowels), and vowels. The second dimension (two bits) was used to subdivide these classes. The third dimension (three bits) classified the phonemes by place of articulation (front, middle, back). Finally, the fourth dimension (two bits) divided the consonants into voiced and unvoiced, and vowels into long and short.

Conceptually there is only a single hidden layer in the predictive network. But in reality, we divided this hidden layer into two halves, as shown in Figure 6.8(b). This allows the forward pass computations on each half of the network to be cached, so that for a given frame and state, the forward pass computations over all contexts can be reduced to a series of output sigmoids using different precomputed net inputs. This saves a considerable amount of redundant computation.

We evaluated the context-dependent HCNN on the CMU Conference Registration database. Our best results are shown in Table 6.7 (for speaker A, perplexity 111). In this evaluation, the predictive network's inputs included 64 speech inputs (2 frames of speech represented by 16 melscale coefficients and 16 delta coefficients), 5 state inputs, and 10 contextual inputs; the network also included 30 hidden units on the speech side, plus 5 hidden units on the context side; and of course 16 outputs representing the predicted speech frame. As in the LPNN, all phonemes used two alternate models, with the best one automatically linked in. In contrast to the LPNN, however, which used a 6-state phoneme model implemented by 3 networks, this context-dependent HCNN used the 5-state phoneme model shown in Figure 6.9 (or a 3-state model for silence), implemented by a single network per phoneme with state inputs. The CD-HCNN achieved much better results than the LPNN (72% vs. 60%), suggesting that the hidden control mechanism provides an effective way to share parameters, and that context dependence improves the specificity of its phoneme models.

System	LPNN	CD-HCNN
Substitutions	28%	20%
Deletions	8%	6%
Insertions	4%	2%
Word accuracy	60%	72%

Table 6.7: Results of LPNN and context-dependent HCNN (speaker A, perplexity 111).

An error analysis at the phoneme level revealed that there was still a phoneme error rate of 20% after training was complete. Most of the confusions involved the phonemes `AH,AE,EH,UH,D,K,N`.

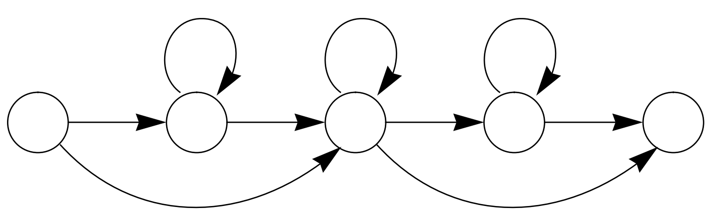

Figure 6.9: 5-state phoneme model used in CD-HCNN experiments.

6.4.3. Function Word Models

Lee (1988) showed that function words — short words like “a”, “of”, “that” — are particularly difficult to recognize, because they have strong coarticulation effects, are very frequent in continuous speech, and are often poorly articulated by the speaker. Inadequate modeling of these words can significantly degrade a system’s overall word accuracy.

We improved the accuracy of our system by introducing function word models. We selected the three words with the highest error rate in our system (“a”, “the”, “you”), and created whole-word models for these, with states indexed by a hidden control inputs. That is, rather than representing these words by a sequence of standard phoneme models, these words got independent models, each of which was represented by a single HCNN with from two to five additional inputs to identify the state of the word (as in Levin 1990). These function word models were also context-dependent; the contextual inputs were arbitrarily set to the initial phoneme of the function word. Note that because of the mutual independence of the predictive networks, there was no need to retrain the other phoneme models when these new function word models were trained.

Evaluating this system under the same conditions as in the previous section, we found that this system achieved 75% word accuracy, which represents 10% fewer errors than the system without function word models.

6.5. Weaknesses of Predictive Networks

Our experience suggests that predictive networks are not very effective for speech recognition. On the Conference Registration database at perplexity 111, we obtained only 60% word accuracy with our basic LPNN, or 75% when the system was enhanced by hidden control inputs, context dependent modeling, and function word modeling. By contrast, a primitive HMM also achieves 75% on this task, and a simple LVQ based system achieves 80% word accuracy.

We have concluded that predictive networks suffer from two weaknesses: (1) a lack of discrimination, and (2) inconsistency between training and testing criteria. It may be possible to correct these problems, but our research stopped short of doing so. We discuss these problems in the following sections.

6.5.1. Lack of Discrimination

Predictive networks are ordinarily trained independently of each other; as a result, there is no discrimination between the acoustic models. This means there is no explicit mechanism to discourage models from resembling each other, which leads to easily confusable phone models, which in turn degrades word recognition accuracy. This weakness is shared by HMMs that are trained with the Maximum Likelihood criterion; but the problem is more severe for predictive networks, because the quasi-stationary nature of speech causes all of the predictors to learn to make a quasi-identity mapping, rendering all of the phoneme models fairly confusable. For example, Figure 6.10 shows an actual spectrogram and the frame-by-frame predictions of the /eh/ model and the /z/ model. Disappointingly, both models are fairly accurate predictors for the entire utterance.

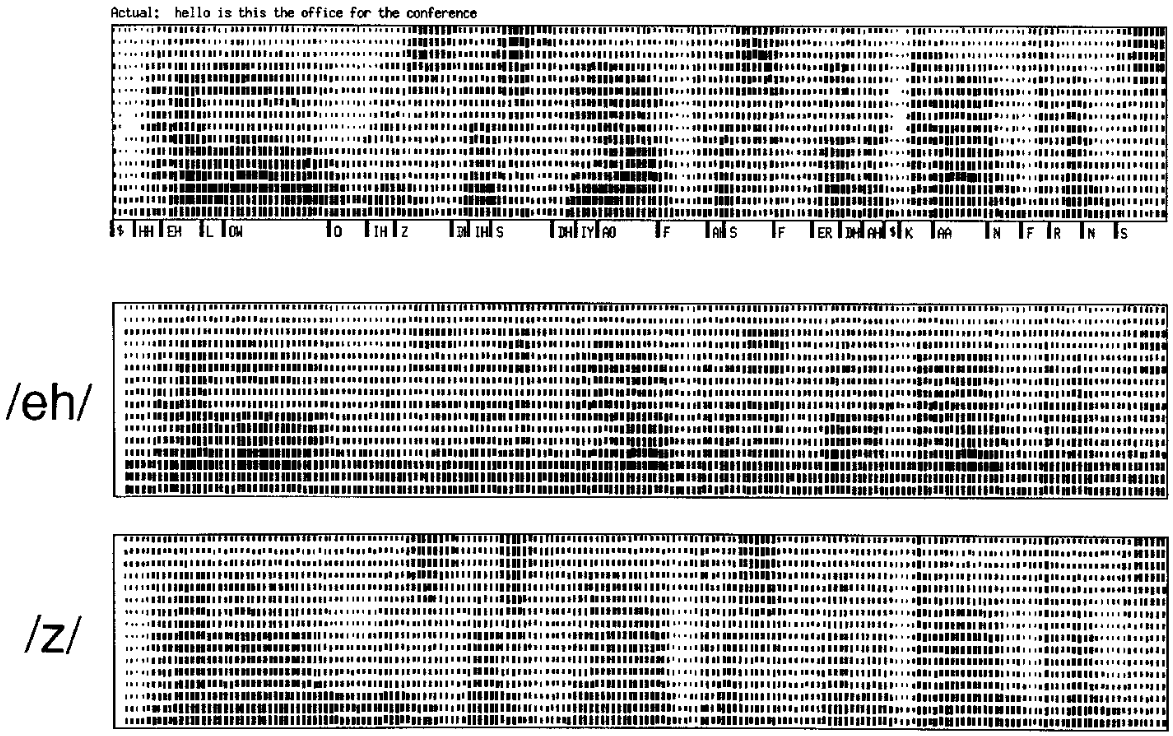

Figure 6.10: Actual spectrogram, and corresponding predictions by the /eh/ and /z/ phoneme models.

There are many ways to partially compensate for this lack of discrimination. For example, we can use more input frames (as long as they are temporally close enough to be relevant to the predicted frame), thereby making each network behave less like an identity mapper, so they can be more easily distinguished. Or we can introduce alternate phone models, or context dependent models, or function word models, or a number of other improvements. However, while each of these techniques may improve the performance of the system, they do not address the lack of discrimination between models, so performance will always remain suboptimal.

What is really needed is a way to discriminate between models, by applying *positive training* to the correct model while somehow applying *negative training* to all of the incorrect models. However, it is not immediately clear what kind of negative target makes sense for a predictive network. In hindsight, we see that there are two general types of target that could be used during both positive and negative training: (1) a vector, analogous to the predicted frame; or (2) a scalar, corresponding to some transformation of the predicted frame (or of all predicted frames). In our research, we studied the first approach, but we had not yet thought of the second approach; it now appears that the second approach may have more promise. The remainder of this section will describe these two approaches.

6.5.1.1. Vector targets

If a prediction-based system uses vectors as training targets, then by definition the target for positive training is the actual frame at time *t*, but there is no obvious target vector for negative training. In our attempts to perform negative training, we studied two possible strategies, neither of which was successful.

The first strategy was to use the actual frame at time *t* as a target for both positive and negative training, but to perform gradient *descent* for the correct network and gradient *ascent* for all incorrect networks. This rather naive approach failed because the force of discrimination is proportional to the prediction error of the network, such that negative training is weakest for the most confusable model, and becomes stronger for all models that have already been pushed away. This is an unstable dynamic, which inevitably throws the models into chaos.

The second strategy returned to using gradient descent for both positive and negative training, but tried to supply a target vector for negative training that would distinguish each model from the others. We observed that each network receives positive training in only a small region of acoustic space (i.e., those frames corresponding to that phoneme), and consequently for any other input frames it will compute an undefined output, which may overlap with the outputs of other predictors. If the network has learned to approximate an identity mapping in its defined region, then it will also tend to approximate an identity mapping in its undefined region. To discourage this behavior, we applied negative training throughout this undefined region, using a target that differed from the actual frame at time *t*. We chose the negative target vector to be the average of all the positive frames associated with that network; this clearly avoids identity mappings, because, for example, the `A` model is trained to predict an average `A` frame whenever the input frames belong to `B`. Unfortunately, this technique failed because each network learned to compute an essentially constant output, corresponding to the average frame of its associated phone. This happened, naturally, because the network was trained to map virtually any input to a constant output, except for a few positive predictions, which also happened to resemble that constant output. We tried to sensitize our networks by using a smaller learning rate for negative training (to emphasize the positive samples) and by increasing the size of our networks (so they could learn more detailed mappings); but our efforts were not successful.

From our experience we have concluded that discriminative training in a predictive system is at best nontrivial and probably infeasible using vector targets.

6.5.1.2. Scalar Targets

An interesting alternative involves redefining the boundaries of a predictive network, so that the associated prediction error (i.e., the Euclidean distance between the predicted frame and the actual frame) is computed internally to the network by a special post-processing layer. With this perspective, the output of the network is a scalar, equal to the Euclidean distance

$$d = \sum_{i}^{N} (y_i - t_i)^2 \tag{65}$$

which lies in the range [0..N] if each frame has N coefficients in the range [0..1]. This can be transformed to a class membership function by inverting and normalizing it, using

$$z = \exp(-d) \tag{66}$$

so that a perfect prediction yields $z = 1$, and a poor prediction yields $z \approx 0$. Now discriminative training is simply a matter of training the correct network on the target $T = 1$, and all of the incorrect networks on the target $T = 0$. Of course, error must now be backpropagated through the equations in the post-processing layer. If we assume the squared error criterion

$$E = \frac{1}{2}(z - T)^2 \tag{67}$$

for each network, then at the frame prediction layer we have

$$\begin{aligned} \frac{\partial E}{\partial y_i} &= \frac{\partial E}{\partial z} \cdot \frac{\partial z}{\partial d} \cdot \frac{\partial d}{\partial y_i} \\ &= (z - T) \cdot (-z) \cdot (2(y_i - t_i)) \end{aligned} \tag{68}$$

Note that this backpropagation equation involves two types of targets: T at the class membership layer, and t_i at the frame prediction layer. It can be seen that this learning rule causes the system to discriminate between models because $(z - T)$ is negative for the correct network and positive for all incorrect networks.

A variation of this approach is to first normalize the scalar outputs so that they sum to 1, like probabilities:

$$p^j = \frac{z^j}{\sum_k z^k} \qquad \text{so} \qquad \sum_j p^j = 1 \tag{69}$$

where superscripts indicate the network index. As before, the correct network is trained on the target $T = 1$, and all the incorrect networks are trained on the target $T = 0$. This time, if the error measure is still

$$E = \frac{1}{2}\sum_j (p^j - T^j)^2 \tag{70}$$

then at the frame prediction layer we have

$$
\begin{aligned}
\frac{\partial E}{\partial y_i^k} &= \sum_j \frac{\partial E}{\partial p^j} \cdot \frac{\partial p^j}{\partial z^k} \cdot \frac{\partial z^k}{\partial d^k} \cdot \frac{\partial d^k}{\partial y_i^k} \\
&= \sum_j \left[p^j - T^j\right] \cdot \left[\frac{\delta_{jk} \sum_k z^k - z^j}{\left(\sum_k z^k\right)^2}\right] \cdot \left[-z^k\right] \cdot \left[2\left(y_i^k - t_i\right)\right] \\
&= \sum_j \left[p^j - T^j\right] \cdot \left[p^k\left(p^j - \delta_{jk}\right)\right] \cdot \left[2\left(y_i^k - t_i\right)\right]
\end{aligned}
\tag{71}
$$

Mellouk and Gallinari (1993) used such normalized scalar targets to introduce discrimination into a predictive system that resembled our basic LPNN. Although their system was a continuous speech recognizer, they evaluated its performance on phoneme recognition. In their preliminary experiments they found that discriminative training cut their error rate by 30%. A subsequent test on the TIMIT database showed that their phoneme recognition rate of 68.6% was comparable to that of other state-of-the-art systems, including Sphinx-II.

Normalized outputs are somewhat more discriminative than non-normalized outputs, since normalized outputs are mutually constrained so that when the correct one increases, all of the incorrect ones will decrease. This property might be called *implicit discrimination*. By contrast, *explicit discrimination* involves contrasting positive and negative training, i.e., training the correct model to output a 1 while training all incorrect models to output a 0. Note that these two modes of discrimination operate independently of each other. Explicit discrimination probably has more impact than implicit discrimination, since it involves greater contrast. Nevertheless, it may be best to combine both types of discrimination, as Mellouk and Gallinari have done.

We conclude that a predictive system *can* become discriminative by transforming the vector outputs to scalar outputs, so that the network can be trained on targets of 0 and 1. However, we did not empirically investigate this approach in our research.

6.5.2. Inconsistency

The second major problem with predictive networks is that their standard training criterion is inconsistent with the testing criterion. That is, predictive networks are trained to make accurate predictions of speech frames, but the testing criterion is something completely different, i.e., word recognition accuracy. We hope against hope that these two criteria are strongly correlated, but in fact we find that the LPNN's excellent frame predictions translate to poor word recognition accuracy. Evidently there is only a weak correlation between frame prediction and word recognition.

Training and testing could be made more consistent by extending the architecture to support *word level training*. This would involve introducing a word level unit for each word in the vocabulary, connecting it to the prediction error layer along its associated DTW alignment path, and backpropagating error down from the word layer using a target of 1 for the correct word and 0 for incorrect words. We will discuss the technique of word level training in greater detail in the next chapter, in the context of classification networks.

In conclusion, predictive networks suffer from two major weaknesses, i.e., a lack of discrimination, and inconsistency between the training and testing criteria. We have discussed some potential remedies for each of these problems. Rather than actually pursuing these remedies in our research, however, we chose to move on and study classification networks, because they support discrimination much more naturally, and they appeared likely to give superior results. Our research with classification networks is described in the next chapter.