bias toward higher performance for the decomposed network. Whether a decomposed network would learn faster than a non-decomposed network when the number of weights in the two networks is approximately equal remains to be tested.

Second, we performed a more extensive search in $\eta$ and $\alpha$ space than [Kamm and Singhal, 1990], resulting in generally higher learning rates and training times of only a few epochs. The range of learning rates explored in the current study included higher rates than those explored in [Kamm and Singhal, 1990], so it is difficult to determine whether the learning speed-up (in terms of arithmetic operations) is attributable to problem decomposition or learning rate. To address this issue, we also trained monolithic 125ms networks for approximately the same number of operations as the decomposed net, using a) the learning parameters used for the 125ms subnetwork and b) the original parameters for the monolithic network. On the 200-sentence-test set, these networks had normalized AHAs of 47.3% and 46.3% and normalized AFAs of 5.9% and 5.1%, respectively. This performance is markedly poorer than performance of the decomposed network, supporting the hypothesis that the problem decomposition strategy was a primary contributor to the learning speed-up.

The problem decomposition strategy facilitated the more thorough search for learning parameters, since it allowed the use of different parameters for training the different subnetworks. In many empirical studies utilizing the back-propagation algorithm, it is difficult to control for the $\eta$ and $\alpha$ parameters. Often, they are held constant over all experimental conditions. However, since optimal parameter values are often problem-dependent, this constraint can result in misleading comparisons if $\eta$ and $\alpha$ are chosen optimally for one network but not another. Our experience in this project supports the fact that the careful selection of network training parameters is critical for successful training (and performance).

## 6 Summary

Problem decomposition was applied to AP-net, a neural network for mapping acoustic spectra to phoneme classes. AP-net was decomposed into subnetworks differing in both output units (assigning phoneme classes to subnetworks based on their durations) and input units (providing different input spans to the subnetworks). Subnetworks were trained individually and then combined into a larger network using "glue" units and fine-tuning of the entire network. This problem decomposition strategy achieved comparable performance to that of a monolithic network, but required an order-of-magnitude fewer arithmetic operations during training. This result provides evidence for the general utility of this problem decomposition methodology for addressing the learning speed problem in back-propagation.

## References


[Fisher *et al.*, 1987] W. M. Fisher, V. Zue, J. Bernstein, and D. Pallett. An acoustic-phonetic data base. *J. Acoust. Soc. Am. Suppl.*, 81(1):S92, Spring 1987.

[Kamm and Singhal, 1990] C. A. Kamm and S. Singhal. Effect of neural network input span on phoneme classification. In *Proceedings of the International Joint Conference on Neural Networks, 1990*, volume 1, pages 195–200, San Diego, 1990. IEEE.

[Pratt *et al.*, 1991] Lorien Y. Pratt, Candace A. Kamm, and Jack Mostow. Direct Transfer of Learned Information among Neural Networks, 1991. (To Appear), Proceedings for the American Association for Artificial Intelligence (AAAI-91).

[Rumelhart *et al.*, 1987] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In David E. Rumelhart and James L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, pages 318–362. MIT Press: Bradford Books, 1987.

[Shavlik *et al.*, 1991] J. W. Shavlik, R. J. Mooney, and G. G. Towell. Symbolic and neural net learning algorithms: An experimental comparison. *Machine Learning*, 6(2):111–143, 1991.

[Waibel *et al.*, 1989] Alexander Waibel, Hidefumi Sawai, and Kiyohiro Shikano. Modularity and scaling in large phonemic neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(12):1888–1898, December 1989.

## 5 Results and Discussion

Table 1 summarizes the conditions used to train networks in this study. For comparison, conditions for the monolithic (non-decomposed) network are also shown.

| Network | Parameter Pairs Tried | $\eta$ | $\alpha$ | Epochs | Operations per Pattern | Total Operations |
|---|---|---|---|---|---|---|
| 125-ms Monolithic AP-net | 4 | .1 | .10 | 300 | 32486 | $10.62 \times 10^{11}$ |
| Decomposed AP-net | | | | | | |
| 35ms subnet | 6 | 2.5 | .10 | 7 | 8250 | $.06 \times 10^{11}$ |
| 65ms subnet | 5 | 1.2 | .20 | 3 | 12705 | $.04 \times 10^{11}$ |
| 125ms subnet | 7 | 3.0 | .01 | 10 | 32486 | $.35 \times 10^{11}$ |
| after glue training | 2 | 2.0 | .50 | 9 | 46485 | $.46 \times 10^{11}$ |
| after fine tuning | 1 | 2.0 | .50 | 2 | 88199 | $.19 \times 10^{11}$ |
| Total | | | | | | $1.10 \times 10^{11}$ |

Table 1. Training Operations

As can be seen above, three subnetworks were trained, then glue training was performed, followed by fine tuning. The second column shows the number of parameter pairs ($\eta$, $\alpha$) explored. The $\eta$ and $\alpha$ values shown in the third and fourth columns were selected for training the network. The number of operations per pattern (column six) was calculated as described in Section 4. The total operation count was obtained by multiplying the operations per pattern by the number of patterns (108983) times the number of epochs used for training.

The resulting performance scores of each of the decomposed networks (as tested on the 200-sentence test set, containing 112970 patterns), along with the scores for the 125-ms monolithic (non-decomposed) network, are shown in Table 2.

| Network | top-3 normalized AHR | top-3 normalized AFA | top-3 AHR | top-3 AFA |
|---|---|---|---|---|
| 125-ms Monolithic AP-net | 61.7 | 3.1 | 55.6 | 2.5 |
| Decomposed Subnetworks | | | | |
| 35ms subnet | 70.1 | 7.4 | 69.0 | 2.1 |
| 65ms subnet | 65.2 | 5.8 | 62.0 | 2.4 |
| 125ms subnet | 33.1 | 4.3 | 33.0 | 4.4 |
| Full Decomposed AP-net | | | | |
| after glue training | 56.5 | 4.0 | 45.0 | 3.1 |
| after fine tuning | 62.4 | 3.5 | 55.6 | 2.5 |

Table 2. Classification Performance

The top-3 normalized AHR and normalized AFA measures are calculated as described in Section 4. Unnormalized AHR and AFA measures are shown for purposes of comparison. If we measure overall performance as the top-3 normalized AHR minus the top-3 normalized AFA, then the final network (after glue training and fine tuning) has essentially equivalent performance to the 125ms monolithic network, in only $\frac{1.1 \times 10^{11}}{10.62 \times 10^{11}} \approx 10\%$ of the operations. This is a significant savings over the training time for the monolithic network.

We also performed tests on an independent set of 850 sentences that were not in either the training set or the test set used for cross-validation. This set yielded a top-3 normalized AHR of 62.6 for the 125ms monolithic AP-net, and a score of 65.7 for the decomposed network.

Given the magnitude of the difference between learning times of the monolithic vs. the decomposed network and given the fact that problem decomposition has shown significant learning speed improvement in more controlled studies, it is probable that the savings can be at least partially attributed to the problem decomposition methodology. However, there are a couple of other differences between the monolithic and decomposed AP-net experiments that could not be thoroughly controlled for, due to the very large size of this network.

First, the decomposed network had many more weights than the monolithic AP-net. The larger number of weights might make modelling the training data a simpler task, and, as a consequence, might result in a

training, and for fine tuning by running a few training epochs and choosing the parameters that generated the lowest error score.

Subnetworks were trained until there was no substantial improvement in performance over time on the testing data. Weights from subnetwork epochs with the best score were used to initialize those subnets in the full decomposed network. Glue training was then performed, and when the performance score leveled out, fine-tuning was performed.

## 4 Measurement Techniques

**Measurement of AP-net Network Performance** As discussed in [Kamm and Singhal, 1990], determining an appropriate performance measure for AP-net is not a straightforward process. The values reported in that paper included the top-3 average hit rate (AHR) and top-3 average false alarm (AFA). The top-3 hit rate for each output class was defined as the number of test patterns where the output unit corresponding to the target (i.e., true) class had one of the three highest activations (over all output classes), divided by the number of patterns where that class was the target. The top-3 AHR was the average of the top-3 hit rate for the 46 phoneme classes. The top-3 false alarm rate for output class $i$ was computed as the number of patterns where (a) the target phoneme is not in class $i$, and (b) the output unit corresponding to class $i$ has one of the three highest activations, and (c) the output unit corresponding to the target phoneme does *not* have one of the three highest activations, divided by the total number of patterns where the target phoneme is *not* in class $i$. The top-3 AFA was defined to be the average of the false alarm rates for the 46 output classes.

AP-net generates an output activation vector for each successive 5-ms *frame* of speech (based on 35 ms, 65 ms, or 125 ms of context information). Phoneme durations typically exceed 5 ms, so, in order to obtain a phoneme string from these activation vectors, sequences of similar output activation vectors from AP-net must be postprocessed and compressed into phonemes. Because optimal post-processing strategies are not yet developed, the top-3 score was used as a performance measure, on the assumption that providing information about the top three phoneme-class candidates to the post-processing step would yield higher phoneme recognition performance than providing only information about the top candidate. However, the top-3 AHR and AFA measures used by [Kamm and Singhal, 1990] do not take into account the relative frequency of occurrence of the different phoneme classes in the corpus. To provide a performance estimate that reflects the distribution of phonemes, we calculated a *normalized* top-3 AHR and AFA score by computing a weighted average of phoneme hit rates. Specifically, the proportion of each phoneme in the corpus (or in the subnetwork phoneme set) is multiplied by each hit rate and false alarm score and the result is summed. These normalized averaged scores are generally about 5% higher than the scores reported in [Kamm and Singhal, 1990], reflecting the fact that these networks tended to have higher performance on the more frequently occurring phoneme classes.

**Measurement of AP-net Learning Time** Back-propagation network learning time is usually reported as the number of epochs required for training. However, for accurate comparison of learning times between non-decomposed networks and those for which glue training is performed, a more precise measure is required. This is because during glue training, many network weights remain fixed, requiring fewer calculations on the back-propagation pass of the algorithm. We considered several measures of training time, including counting the number of weight updates in the network. Unfortunately, this measure inaccurately favors learning with fixed weights, because it doesn't count calculations spent during the feedforward pass, when all weight values are used.

A more accurate measure of learning time is to explicitly calculate the number of arithmetic operations performed by the learning algorithm. By examining the equations used by back-propagation to calculate node activations and weight updates, we derived a measure of the number of the operations (addition, subtraction, multiplication, division, exponentiation) executed in an epoch. This measure was then used to compare network learning times.

"glue" units whose purpose was to combine the results from the subnetworks. The following training steps were used.

1. **Subnetwork Training:** Each subnetwork is trained separately, possibly using different $\eta$ and $\alpha$ parameters. The number of hidden units in each subnet can be adjusted independently, and the subnets can be trained in parallel.

2. **"Glue" Training:** The subnetworks are combined and additional "glue" units are added and trained, with subnetwork weights fixed.

3. **Fine Tuning:** All network weights are allowed to change during further training.

The performance obtained by [Waibel *et al.*, 1989] for a decomposed network was slightly better than performance of a monolithic (i.e., non-decomposed) network. In addition, there was a marked decrease in learning time. [Pratt *et al.*, 1991] validated this study on a vowel recognition problem with a different network topology and a different stopping criterion and showed that the learning speed for several decomposed networks was significantly faster than the learning speed of monolithic (non-decomposed) networks.

**The Decomposed AP-net Architecture** The problem decomposition methodology introduced in the previous section can be applied to networks with different topologies. The networks in [Waibel *et al.*, 1989] had two layers of hidden units, while those in [Pratt *et al.*, 1991] had only a single hidden layer. In both cases, subnetworks were connected to all input units; that is, the problem was decomposed by dividing the output units into non-overlapping subsets.

In the current study, we maintained the multiple input spans used in the monolithic AP-net experiments (Section 2). The use of multiple input spans can be viewed as decomposing the problem in terms of the input units. The problem was further decomposed by assigning the output units so that different subnets included different (but overlapping) subsets of output classes. The results reported in [Kamm and Singhal, 1990] showed that the 35ms and 65ms monolithic networks had much higher performance for phonemes with short average duration than for the longest phonemes (diphthongs). Based on this finding, we decomposed the problem on the output units so that the two subnetworks with short-duration input spans "specialized" on subsets of phonemes with short and intermediate average durations. To assign phoneme classes to subnetworks, we analyzed a portion of the DARPA A-P corpus to determine the distribution of durations for each phoneme class. Phonemes which were usually of short duration were included as output classes for the first network; short- and mid-length phoneme classes formed the output classes for the second network; and all phonemes classes were included in the third network. Hidden unit counts for subnetworks were chosen arbitrarily in proportion to the expected difficulty of each learning task.

The decomposed AP-net topology is shown in Figure 2.

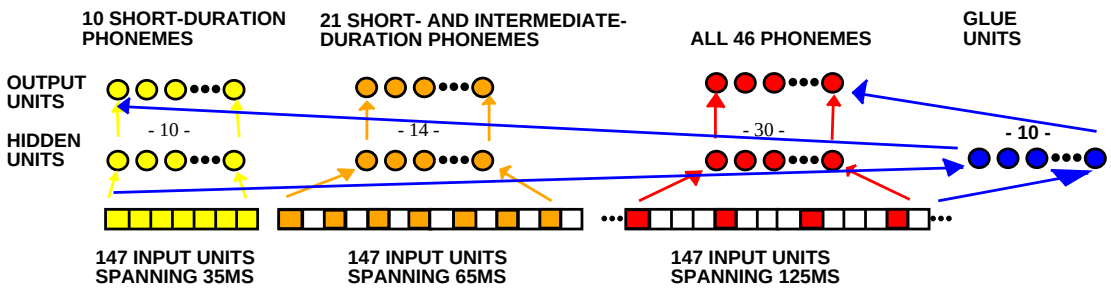


Figure 2: Decomposed AP-net architecture

**Training Methodology** Subnetwork training, glue training, and fine tuning were performed as described in Section 2, using a custom-built simulator, written in Fortran, running on an Alliant FX80 computer.

The subnetwork training data was the same 200-sentence set used to train the networks described in Section 2. We performed a rough search for good training parameters $(\eta, \alpha)$ for each subnetwork, for glue

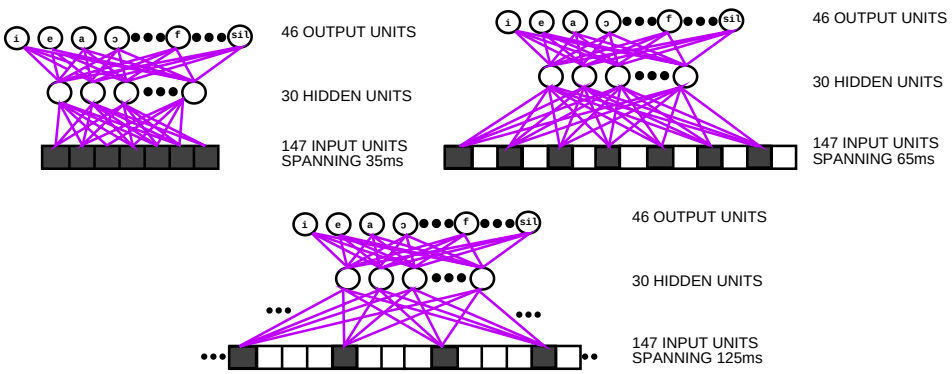


Figure 1: Three AP-net networks with different input spans

**Signal Preprocessing** The training set for these experiments was a portion of the DARPA acoustic-phonetic (A-P) corpus ([Fisher *et al.*, 1987], also called the TI-MIT corpus), which contains recordings of continuous utterances from 630 talkers, along with transcriptions specifying the phonemes uttered. This corpus is used extensively in phoneme recognition research.

A subset of the DARPA A-P corpus consisting of 200 sentences spoken by 10 male and 10 female talkers was used for training. Testing was performed on a disjoint 200-sentence set. Input to the network was a spectro-temporal representation consisting of the scaled outputs of 21 overlapping bandpass filters (with center frequencies from $\approx$ 140 Hz to $\approx$ 7000 Hz and 1-Bark spacing) sampled at seven 5-ms time steps. Network input thus consisted of seven sets of 21 values. This input was obtained from 10-ms frames of speech, using a 5-ms frame shift. This process yielded 108983 training patterns in the training set and 112970 patterns in the testing set.

**Training Procedure** For training, the target vector for each training pattern identified the phoneme that occurred during the center frame of the seven input frames. The output unit corresponding to this phoneme received a target activation of 0.9; all other output units received target activations of 0.1. A modified version of the back-propagation algorithm [Rumelhart *et al.*, 1987] was used for training. The two changes to the learning algorithm were (a) to scale the learning rate on each output unit relative to the frequency of occurrence of its corresponding phoneme class and (b) to disable training of certain output units whenever the training pattern contained a phoneme transition, in order to avoid ambiguous training examples.

The networks were trained using back-propagation parameters $\eta$ (learning rate) = 0.1 and $\alpha$ (momentum) = 0.1. These parameters were chosen as the best out of a group of four parameter pairs tried; that is, using these values, squared error decreased monotonically and most rapidly over the first few training epochs. The same parameters were used for all three networks, each of which was trained for 300 epochs.

**Test Results** After training, the weights were fixed and the networks were tested on a set of 200 sentences from speakers not used during training. Using the top-3 normalized average hit rate measure (see Section 4), the best performance for the test sentences was obtained by the 125ms network, with an average hit rate of 61.7% and an average false positive rate of 3.1%. In the current study, we use this performance score as a benchmark for comparison with the network trained using the problem decomposition methodology.

## 3 Problem Decomposition in AP-net

**Introduction** The use of problem decomposition in back-propagation neural networks was introduced by [Waibel *et al.*, 1989], who attempted to extend a network trained to recognize the consonants /b,d,g/ to one that recognized all stop consonants (/b,d,g,p,t,k/) and observed that learning time increased more than linearly with problem complexity. To alleviate this problem, several techniques were used to combine the weights from pre-trained /b,d,g/ and /p,t,k/ subnetworks. The best performance was obtained when the weights on links between the input layer and the first hidden layer of the pre-trained subnetworks were used to initialize a larger network with two hidden layers. The first hidden layer was supplemented by several

# Improving a Phoneme Classification Neural Network through Problem Decomposition

*To appear: Proceedings of IJCNN-91*

L. Y. Pratt
Computer Science Department
Rutgers University
New Brunswick, NJ 08903
pratt@paul.rutgers.edu

C. A. Kamm
Speech Technology Research
Bellcore
445 South Street, Morristown, NJ 07962-1910
cak@thumper.bellcore.com

**Abstract**

[1] In the study of neural networks, it is important to determine whether techniques that have been validated on smaller experimental tasks can be scaled to larger real-world problems. In this paper we discuss how a methodology called *problem decomposition* can be applied to AP-net, a neural network for mapping acoustic spectra to phoneme classes. The network's task is to recognize phonemes from a large corpus of multiple-speaker, continuously-spoken sentences. We review previous AP-net systems and present results from a decomposition study in which smaller networks trained to recognize subsets of phonemes are combined into a larger network for the full signal-to-phoneme mapping task. We show that, by using this problem decomposition methodology, comparable performance can be obtained in significantly fewer arithmetic operations.

## 1 Introduction

The lengthy training times required by neural networks have been identified as one of their major weaknesses in comparison to other methods for learning from examples (cf. [Shavlik *et al.*, 1991]). In this paper, we explore the use of a strategy called *problem decomposition* to improve the learning speed and performance of "**AP**-net", a neural network designed to map **a**coustic spectra to **p**honeme classes. Problem decomposition (also called "modular decomposition") has previously produced both performance improvement ([Waibel *et al.*, 1989]) and significant learning speed improvement ([Pratt *et al.*, 1991]) on less complex tasks than the one attempted by AP-net.

The motivation for problem decomposition is that training time can be significantly decreased by breaking a combinatorial search problem into several sub-problems. For example, if there are ten possible values for each of six neural network weights, then there are $10^6$ distinct combinations of those weights that may need to be examined to find a solution. If we know that a solution involving half the weights can be determined independently of the others, then by breaking the search problem into two different tasks using three weights each, there are only $2 \times 10^3 << 10^6$ weight combinations to explore. Problem decomposition has been studied extensively within the AI community as a means of reducing otherwise intractable search problems. This study demonstrates the application of problem decomposition to larger neural networks than those previously explored.

## 2 Previous work: Monolithic AP-net networks

**Networks with Varying Input Durations** In a previous experiment designed to study effect of network input span on phoneme classification performance ([Kamm and Singhal, 1990]), three AP-net networks were trained. The first network derived its input from seven contiguous 5-ms frames of speech, providing information spanning 35 ms of speech. Each speech frame was represented by 21 input units, yielding $7 \times 21 = 147$ total input units. The second network also received input from seven frames of speech, but these were obtained from every other 5-ms segment of the input signal, yielding an input duration spanning 65 ms. The third network skipped three 5-ms frames between inputs, yielding a 125-ms input duration. All three networks had 30 hidden units and 46 output units, each corresponding to a phoneme class. Their architectures are shown in Figure 1.

[1] Copyright IEEE, 1991