# SPARSE CONNECTION AND PRUNING IN LARGE DYNAMIC ARTIFICIAL NEURAL NETWORKS

*Nikko Ström*
Department of Speech, Music and Hearing
KTH (Royal Institute of Technology), Stockholm, Sweden
Tel. +46 8 790 75 63, FAX: +46 8 790 78 54, E-mail: nikko@speech.kth.se

## ABSTRACT

This paper presents new methods for training large neural networks for phoneme probability estimation. A combination of the time-delay architecture and the recurrent network architecture is used to capture the important dynamic information of the speech signal. Motivated by the fact that the number of connections in fully connected recurrent networks grows super-linear with the number of hidden units, schemes for sparse connection and connection pruning are explored. It is found that sparsely connected networks outperform their fully connected counterparts with an equal or smaller number of connections. The networks are evaluated in a hybrid HMM/ANN system for phoneme recognition on the TIMIT database. The achieved phoneme error-rate, 28.3%, for the standard 39 phoneme set on the core test-set of the TIMIT database is not far from the lowest reported. All training and simulation software used is made freely available by the author[1], making reproduction of the results feasible.

## 1. INTRODUCTION

It is well-known that artificial neural networks can be successfully used for phoneme recognition problems, e.g., [1,2,3]. The phoneme recognition rate on the TIMIT database reported in [3] is several percent higher than that of all other systems – a large difference for this type of test. Still, for the last years, the research activities on ANNs for speech recognition have not by far been as intensive as for the prevailing HMM paradigm. Besides the general preference for a well established technology and the somewhat disturbing lack of equally excellent results reported for ANNs on the word-level, a possible reason for the mild interest in ANN solutions can be problems with the network training, i.e., efficiently and robustly determining the parameters of the networks. For example, the recurrent network used in [3] is trained with special hardware and a rather complex training heuristic is used with several *ad hoc* parameters to be determined empirically.

Recurrent connections are not the only path to good results, but other existing solutions have different problems. Another architecture is used with good results in [2]. There, time-delay windows [4] are used instead to capture the temporal cues of the speech signal. The absence of recurrent connections makes the training algorithm more stable, but very large networks are used to achieve good results, and therefore the available computing resources limits the performance.

In contrast to the existing high performing ANN solutions, the well-established Maximum Likelihood (ML) training paradigm for the standard HMM (e.g. [5]) has a robust, theoretically well established training scheme and must be considered a safer route to a functioning speech recognition system today. The HMM paradigm also has the advantage of a large mature body of easily available software.

The aim of this paper is to describe new methods for robust training of large, high performance ANNs using limited computing resources, i.e., reduce the problems of contemporary ANN solutions. Further, the software tool-kit used for training and evaluating the neural networks is made freely available[1] to promote further development in the field.

## 2. HMM/ANN HYBRID RECOGNITION

The phoneme recognizer used in the evaluations on the TIMIT database, is a hybrid HMM/ANN (see for example [4]), where the phoneme output activities of the ANN are interpreted as the *a posteriori* phoneme probabilities $p(c_i \mid o)$, and the observation probabilities, $p(o \mid c_i)$, used in the HMM framework, are derived from the *a posteriori* probabilities using Bayes's rule.

$$p(o|c_i) = \frac{p(c_i|o)}{p(c_i)}\, p(o) \qquad \textbf{(1)}$$

The *a priori* class frequencies $p(c_i)$ are estimated off-line from the training data and the unconditioned observation probability, $p(o)$, is constant for all classes and therefore dropped in the computations.

Basically, a one-state Markov model is used for each phoneme and the transition probabilities are estimated by ML of the durations of the training database. In addition to this weak duration model, a phoneme-dependent minimum duration constraint is imposed by adding extra nodes to the HMM and putting the self-loop on only the last node. The minimum durations are selected such that 5% of the phones in the training data are shorter than the minimum. This fraction was chosen, after some experimenting, to optimize recognition performance. A phoneme bi-gram grammar is used for the transition probabilities between phonemes.

[1] The software and documentation of the NICO tool-kit, used in the ANN simulations in this paper, can be downloaded from the home page: http://www.speech.kth.se/NICO/index.html

The *a posteriori* phoneme probabilities are estimated by the output units of an ANN with 10 ms frame resolution. A feature vector is computed from each frame including the twelve first Mel cepstrum coefficients and the log energy. The input to the ANN is the feature vector and its first and second time derivatives. More details about the signal processing, the estimation of the HMM parameters and the dynamic decoding is given in [6].

## 3. ANN STRUCTURE AND TRAINING

### 3.1 Recurrent connections and time delay windows

Dynamic features of speech such as formant trajectories, are not captured by the short time spectrum used as input to the ANN. Therefore, phonetic classification can be greatly enhanced by considering also the neighboring spectra. A step in this direction was taken in [4] when time-delay neural networks (TDNN) were introduced. Here we will use the term TDNN for all architectures where units are connected to lower layers with time-delayed connections so that the activities depend on the activities of lower layer units within a finite time-delay window. The first experiments with TDNN successfully showed improved classification of stop consonants where the dynamics is of great importance [4]. Later it has been successfully utilized in full-fledged hybrid HMM/ANN speech recognition systems [2].

Recurrent neural networks (RNN) is a different course to utilizing the context in the classification and are currently the most successful architecture for phoneme recognition [3]. In RNN, units in the same layer are connected to each other with a time-delay of one. This approach differs from TDNN in that the activity of a unit depends recursively on activities at all previous times.

TDNNs and RNNs have much in common; in particular, both use time-delayed connections to incorporate context into the classification. If the connections of an RNN can have multiple time-delays instead of just one time step, the resulting network has all the modeling power of both architectures. This unified architecture, RTDNN, introduced in [7], is used in this study.

In the RTDNN framework we attempt to preserve the best features of the TDNN and RNN structures. The concept of look-ahead connections is borrowed from TDNN. It simplifies ANN design because it makes delayed targets unnecessary and the dynamic structure more intuitive. Look-ahead connections force some unit activities to be computed ahead of others, but the network is still a feed-forward network as long as no unit's activity at a particular time depends on its own activity.

### 3.2 Back-propagation through time

The networks are trained using a variation of the back-propagation algorithm. The error gradient is computed using back-propagation through time in a similar manner as in [3].

A way to visualize back-propagation through time is to draw the spatial dimension of the network in one dimension, i.e., line up all units in one column. Then *unfold* the network in the time dimension by drawing one column of units for each time point. Figure 1 shows a very simple example of such an unfolded network. The unfolded network is similar to a network with no delays but as many layers as there are time points. The difference is that the connection weights are shared by all connections that correspond to the same connection in the original network. Back-propagation through time is equivalent to standard back propagation with this additional weight sharing constraint.

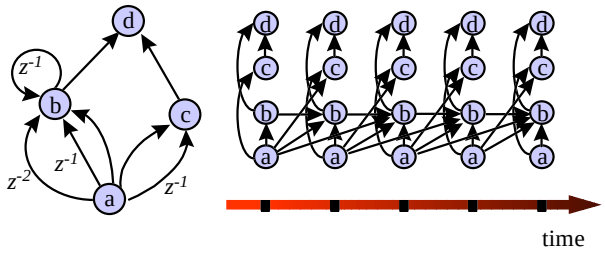


**Figure 1.** A simple dynamic network (left) and the same network unfolded in time (right) where the units a-d are duplicated for each time-point. Connections labeled $z^{-x}$ are delayed $x$ time points.

### 3.3 Weight updating scheme

The error function is defined by setting the target for the output unit of the correct phoneme for each frame to 1.0, and all other targets to 0.0. The cross-entropy error function is used.

The classic weight updating scheme (with some modifications) is still a good choice for problems with a large amount of training data. It can be written

$$\Delta w_{ij}^{(0)} = 0 \qquad \textbf{(2)}$$

$$\Delta w_{ij}^{(n)} = \gamma \Delta w_{ij}^{(n-1)} + \eta \frac{\partial E}{\partial w_{ij}}$$

$$w_{ij}^{(n+1)} = w_{ij}^{(n)} + \Delta w_{ij}^{(n)}$$

where superscript ($n$) indicates a parameter after weight update number $n$, $\gamma$ and $\eta$ are the gain and momentum parameters respectively and $E$ is the error function. The reason that this simple scheme is competitive, is that it can be repeatedly applied before all samples of the training data (one epoch) have been processed, to speed up the convergence. In the case of standard back-propagation, the weights can be updated after every training pattern – so called pattern updating.

The picture becomes more complicated for back-propagation through time because the derivatives depends not only on the current pattern but on the whole sequence of patterns. We have adopted the approximate scheme to update the weights every $N$ frames, i.e., approximate the derivative based on sub-sequences of the training data. This method is also used in [3]. The approximation is worse if the weights are updated

frequently, but on the other hand it is desirable to update frequently, to speed up training. In the simulations, the weights are updated every 20-30 frames (*N* is random distributed **R**[20;30]). This is intuitively reasonable as it corresponds to the length of a syllable.

To reach a minimum of *E*, the gain parameter must be gradually decreased during the training. We have combined this with cross-validation in a manner similar to [2]. The idea is to decrease the gain parameter when the objective function fails to decrease on a validation set. To be more specific, the training data is partitioned into a training set and a smaller validation set. The training set is used for the actual back-propagation training, and after each epoch, the error function *E* is computed for the validation set too (without weight updating). If *E* fails to decrease during an epoch, the gain parameter $\gamma$ is multiplied by a constant factor $\alpha < 1$. In this study $\alpha$ is always 0.5, $\eta$ is 0.7, and the initial value of $\gamma$ is $10^{-5}$.

### 3.4 Network topology

In all simulations, a single hidden layer with recurrent connections with the delays one, two and three frames are used. The 39 input units are connected with a skewed time-delay window, of -1 to +5 frames, to the hidden units, and the 61 output units are connected with a window of -1 to +1 frames to the hidden units. This yields (with *H* hidden units)

$$39\times H\times 7 + H\times H\times 3 + H\times 61\times 3 = 456H + 3H^2 \quad \textbf{(3)}$$

connections. This number grows rapidly with *H* – for example, 300 hidden units gives 406.800 connections.

## 4. PRUNING AND SPARSE CONNECTION

It is our experience that the performance of the ANNs is a monotonously increasing function of the number of hidden units. The reason that the increased number of free parameters does not hurt performance by over-adaptation to the training data, is that the weight updating scheme is controlled by a validation data set. Thus, enlarging the hidden layer is a rather safe way to raise performance. The problem is that the number of connections grows very fast with the number of units which makes training and evaluation unfeasible. However, inspection of trained fully connected networks have shown that a large part of all connections have very small weights. This inspired us to study connection pruning methods and sparse connection schemes to be able to work with large networks with a manageable number of connections.

Connection pruning is a method that reduces the runtime of trained networks. The most well-known method is optimal brain damage (OBD) [9]. An overview of this and other methods is found in [8]. In our experiments a coarse pruning criterion is used – we simply remove all weights smaller (in magnitude) than a threshold. It is important that the network is retrained after pruning. This training converges much faster than the original training, and the retraining corrects most of the errors due to the simplistic pruning criterion.

Unfortunately, pruning has no impact on training time because it is applied after training. To reduce training time we have experimented with *sparsely connected* networks. Before training, there is no available information about which connections are salient, so a random set of connections must be selected. Of course, this is in general not the optimal set, but as will be seen in section 5, the resulting ANN may still be competitive.

A straight-forward, random connection scheme is to consider all connections in a hypothetical, fully connected network and let each be a connection in the actual sparse network with probability $\phi$ (connectivity). The expected number of connections in the sparse network is then $N\phi$, where *N* is the number of connections in the fully connected network. In [10] it is pointed out that the sparse connectivity has the effect of *decoupling* the output units, i.e., all output units are not connected to the same hidden units. Results from several comparative studies are reported, and sparsely connected networks perform as well as, or better than both OBD and fully connected networks.

In RNNs, the number of connections is proportional to the square of the size of the hidden layer (the second term of (3)). Thus, for large hidden layers, a very low connectivity is necessary to reduce connections to a manageable number. This reduces the network's functional capacity more than can be compensated for by the increased number of hidden units.

To overcome the square relationship between the layer size and the number of hidden units, we introduce localized connectivity. In this scheme, recurrent connections between unit $u_1$ and $u_2$ are added with the probability

$$\mu \cdot e^{-d[u_1,u_2]/\sigma} \quad \textbf{(4)}$$

where $d[u_1,u_2]$ is the distance between the units, $\mu$ is the overall connectivity constant, and $\sigma$ is a constant of spread. Distance is defined simply by ordering the hidden units and taking the absolute difference of the ordering numbers. Thus, the self-loop connectivity is $\mu$, and $\sigma$ controls how fast the connectivity decays with distance in the layer. In this study, $\mu$ is always 1.0 and $\sigma$ is varied to control the number of connections.

## 5. TIMIT RECOGNITION RESULTS

An ensemble of sparsely connected ANNs have been trained and evaluated on the TIMIT database. The ANNs have varying connectivity for their three different types of connections: 1) the connections from the input units, 2) the recurrent connections, and 3) the connections to the phoneme output units. The networks were trained on all training utterances except the "sa" sentences, and evaluated on the core test set.

The results are shown in Figure 2. The underlined error-rates clearly show how the fully connected network is outperformed by sparsely connected ANNs with an equal number of connections. A set of networks with 300 hidden units are evaluated to compare different sparse connection schemes. The sweep over varying hidden layer size from 100 to 600 units, with accompanying decrease in error-rate, show the benefit of being able to work with large hidden layers. Because of limited computing resources, we have not yet trained ANNs with more than 600 units, but because the error-rate is constantly decreasing, we expect larger networks to perform even better. The lowest rate, 28.3%, is already better than all reported HMM-based results, e.g., [11] (30.9%), but not quite as low as the currently lowest reported RNN result [3] (26.1%).

The two rightmost columns of Figure 2 are designed to test the idea of [10] that connection to higher layers should be more sparse than lower layers, in order to decouple the output units. Their hypothesis is supported by our experiment and will be put to use in future studies.

In a separate experiment, connection pruning was applied to the networks, and it was found that with a weight threshold of 0.08, the number of connections is reduced by circa 50% for all networks. After retraining, the error-rate had never increased by more than 1%.

## 6. CONCLUSIONS

A robust training scheme for high performing ANNs, with manageable computational demands, have been presented for phoneme probability estimation. In phoneme recognition experiments, a fully connected ANN was clearly outperformed by its sparsely connected counterparts with equal or less connections, and the lowest error-rate achieved, 28.3% is competitive also in comparison with other systems.

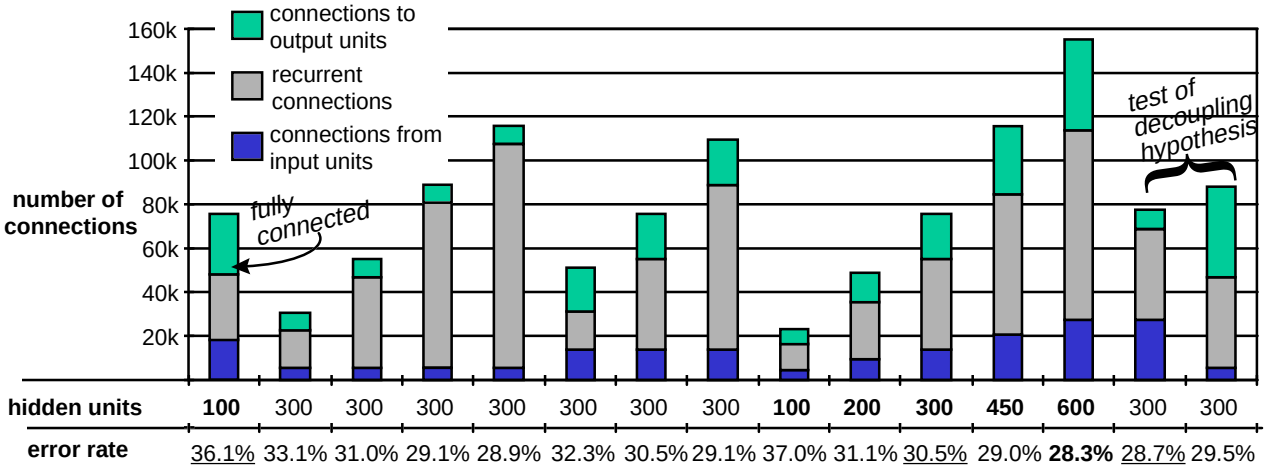


**Figure 2.** Phoneme recognition results for the core test set of the TIMIT database. The 39 symbol set, also used in [3] and [11] is used. Error-rate is defined as the sum of insertions, deletions and substitutions per phone.

## 7. REFERENCES


[1] Elenius K. & Takacs, G., "Acoustic-phonetic recognition of continuos speech by artificial neural networks", *STL-QPSR* **2-3**/1990, pp. 1-44, KTH, Dept. of Speech, Music and Hearing, Stockholm, Sweden, 1990.

[2] Bourlard H. & Morgan N., "Continuous Speech Recognition by Connectionist Statistical Methods", *IEEE trans. on Neural Networks*, **4**(6), pp. 893-909, 1993.

[3] Robinson A.J., "An application of Recurrent Nets to Phone Probability Estimation", *IEEE trans. on Neural Networks*, **5**(2), pp. 298-305, 1994.

[4] Waibel A., Hanazawa T., Hinton G., Shikano K. & Lang K., "Phoneme Recognition Using Time-Delay Neural Networks," *ATR Technical Report TR-006*, ATR, Japan, 1987.

[5] Young S., Jansen J., Odell J., Ollason D. & Woodland P., "HTK – Hidden Markov Toolkit", Entropic Cambridge Research Laboratory, 1995.

[6] Ström "Continuous speech recognition in the WAXHOLM dialogue system", *STL-QPSR* **4**/1996, pp. 67-95, KTH, Dept. of Speech, Music and Hearing, Stockholm, Sweden, 1996.

[7] Ström N., "Development of a Recurrent Time-Delay Neural Net Speech Recognition System", *STL-QPSR* **2-3**/1992, pp. 1-44, KTH, Dept. of Speech, Music and Hearing, Stockholm, Sweden, 1992.

[8] Thimm G. & Fiesler E., "Evaluating pruning methods" , *In 1995 International Symposium on Artificial Neural Networks, Proc. ISANN '95*, pp. A2 20-25, National Chiao-Tung University, Hsinchu, Taiwan, 1995.

[9] Le Cun Y., Denker J. S. & Solla S. A., "Optimal brain damage", *In Advances in Neural Information Processing Systems*, **II**, ed: Touretsky D. S., pp. 589-605, San Mateo, California IEEE, Morgan Kaufmann, 1990.

[10] Ghosh G. & Tumer K., "Structural adaptation and generalization in supervised feed-forward networks", *Journal of Artificial Neural Networks*, **1**(4), pp. 430-458, 1994.

[11] Lamel L. & Gauvain J. L., "High performance speaker-independent phone recognition using CDHMM", *Proc. EUROSPEECH*, pp. 121– 124, 1993.