**Table 1: Speech Recognition Performances (Alphabet Recognition)**

| | training | testing |
|---|---|---|
| manually optimized MSTDNN architecture with DTW | 94.3% | 85.0% |
| manually optimized MSTDNN with gaussian smoothing of the DTW path | 98.9% | 88.0% |
| automatically optimized MSTDNN architecture with standard DTW | 97.1% | 85.0% |
| automatically optimized MSTDNN with gaussian smoothing of the DTW path | 99.5% | 91.7% |

**Table 2: Handwritten Character Recognition Performances (Digit Recognition)**

| | training | testing |
|---|---|---|
| manually optimized MSTDNN architecture without hidden units | 98.3% | 96.5% |
| automatic optimization of the window size and the number of state units | 99.6% | 98.0% |
| automatically optimized architecture with gaussian smoothing of the DTW path | 100% | 99.5% |
| TDNN architecture proposed by [4] on the same data | 100% | 95.5% |
| TDNN architecture manually optimized for the same data | 100% | 98.5% |

## 5. SUMMARY AND CONCLUSIONS

The results on three different tasks show that the ASO algorithm can achieve equal or better results than handtuned architectures without any tuning to the particular task. Table 3 shows that the MSTDNN network optimized by ASO can adapt to different amounts of training data. The handtuned architecture performed equally well for the amount of data that it was optimized for, but did not generalize as well for more data and failed to learn a small subset completely for various learning rates and momentums.

The results suggest that the ASO algorithm is able to optimize MSTDNN type networks for real world applications with varying amounts of training data effectively. Preliminary experiments with regularization techniques like weight decay and weight elimination [9] together with the ASO algorithm have been encouraging. In future, the algorithm will be applied to contiunous speech recognition and continuous (cursive) handwritten character recognition tasks.

**Table 3: Handwritten Character Recognition Performances (Capital Letters) depending on training set size**

| number of training patterns | TDNN architecture manually optimized for 1170 training patterns | automatically optimized MSTDNN architecture |
|---|---|---|
| 520 | no convergence | 81.5% |
| 1170 | 88.5% | 88.5% |
| 1560 | 90.5% | 91.3% |

## Acknowledgements

The authors gratefully acknowledge the support of the McDonnel-Pew Foundation (Cognitive Neuroscience Program) and would like to thank Alex Waibel for lots of helpful discussions.

## REFERENCES

[1] Waibel, A., Hanazawa, T., Hinton, G., Shiano, K., and Lang, K. Phoneme Recognition using Time-Delay Neural Networks. *IEEE Transactions on Acoustics, Speech and Signal Processing*, March 1989

[2] Haffner, P., Franzini, M., and Waibel, A. Integrating Time Alignment and Neural Networks for High Performance Continuous Speech Recognition, *ICASSP* 91

[3] Haffner, P., and Waibel, A. Time-Delay Neural Networks Embedding Time Alignment: A Performance Analysis, *Eurospeech* 91

[4] Guyon, I., Albrecht, P., Le Cun, Y., Denker, W., Hubbard, W. Design of a Neural Network Character Recognizer for a Touch Terminal, *Pattern Recognition*, 24(2), 1991

[5] Baum, E.B., and Haussler, D., What Size Net Gives Valid Generalization?, *Neural Computation*, 1: 151-160

[6] Moody, J. The Effective Number of Parameters: An Analysis of Generalization and Regularization in Nonlinear Learning Systems. in: *Advances in Neural Information Processing Systems* 4, 1991.

[7] Sakoe,H., and Chiba, S., Dynamic Programming Algorithm Optimization for Spoken Word Recognition, *IEEE Transactions on Acoustics, Speech and Signal Processing*, (26): 43-49, 1978

[8] Bodenhausen, U., and Waibel, A., Application Oriented Automatic Structuring of Time-Delay Neural Networks for High Performance Character and Speech Recognition, *ICNN Proceedings, San Francisco*, March 1993

[9] Weigend, A.S., Hubermann, B.A., and Rumelhart, D.E., Predicting the Future: A Connectionist Approach, *TR SSL-90, Xerox Science Laboratory, Palo Alto, CA, 1990*

Fig. 2: The weights from the input units to the state units for the recognition of the handwritten capital letters after training. Negative weights are displayed by white blobs, positive weights by black blobs. Eight input units are used to represent the features recorded from the touch sensitive tablet (see [4]). The weights for the character “A” are displayed at the left bottom. “A” is modeled by two state units. Each of these state units gets input from input windows with size 13 (time delays from 0 to 12). The character “B” is modeled by three states, “C” is modeled by one state etc.

of the output units can be computed in three different ways: The simplest way is to give each state unit an equal share of the time slice that the output unit respresents. The second possibility is to use *Dynamic Time Warping* (DTW)  [7] to find the best path through the activation matrix of the state units. The third possibility is to smooth the DTW path by gaussian functions positioned according to the DTW segmentation [8]. Smoothing of the DTW path allows the states to model the transitions between two states more accurately. If each state specializes on different parts of the spectrogram, then the transition between these parts may not be modeled by any of them. Smoothing allows both states to partially represent the transition.

## 4. SIMULATIONS

The ASO algorithm was tested with an alphabet recognition task (aprox. 3000 isolated words by speaker DBS taken from the same database that was used in [2]) and two handwritten character recognition tasks (aprox. 1000 digits and 2000 capital letters, recorded as described in [4]) so far. For the current simulations the hidden layer of the original MSTDNN was left out. Methods that add hidden units between the input units and the state units are currently tested. The results for both manually optimized architectures and automatically optimized architectures are summarized below (see Tables 1-3). Results with different manually optimized architectures (single state TDNNs with hidden layer) are added for the handwritten character recognition task for comparison. Fig.2 shows the connections from the input layer to the state ayer after constructing the network with the ASO algorithm for the recognition of capital letters. Fig. 2 shows that the algorithm constructs a rather inhomogenous architecture that would be hard to find manually.

### 2.2. Task Decomposition

Instead of learning very complex decision surfaces for the classification of events, it may be better to decompose the classification into the recognition of subevents that have to be observed jointly. In many cases the decision surfaces for the recognition of these subevents are much easier to learn. This method is used in many speech recognition systems. For example, the recognition of words can be decomposed into the recognition of sequences of phonemes or phoneme like units. TDNN's have recently been extended to Multi State Time-Delay Neural Networks (MSTDNNs) [2, 3] that allow the recognition of sequences of ordered events that have to be observed jointly. Unfortunately, this also means that (word specific) state sequence topologies have to be found for the given type and amount of training data.

### 2.3. Confusion Matrix Dependant Construction of the Network Architecture

It was frequently observed that application-oriented researchers using neural networks use the confusion matrix of the training data for manual optimization of the network architectures. A certain architecture is trained until the stopping criterium is reached and then the confusion matrix is evaluated. If a structured approach is used (as in many speech recognition systems), the modelling can be refined if too many errors in a certain class are observed. This kind of approach could be very useful for an automatic optimization procedure.

### 2.4. Early Constructive Changes of the Network Architecture

Waiting for a whole training run and then making decisions on the further optimization of the network is computationally very expensive. Our experience shows that it is possible to detect the most important mistakes very early in the training run and change the architecture early in the training run. Starting the training run again is not necessary.

## 3. APPLICATION OF THE ASO ALGORITHM TO MSTDNN ARCHITECTURES

As can be seen, MSTDNN type networks conform with the first two principles of paragraph 2.1 and are also very powerful classifiers [2, 3]. The architecture of these highly structured networks can be optimized in many ways. For best performance, the size of the input windows, the number of hidden units and the (word specific) state sequence topologies are of critical importance for optimal performance. This makes MSTDNNs a suitable candidate for the demonstration of the ASO algorithm.

The ASO algorithm optimizes all relevant parameters of MSTDNN structures for a given amount of training data. The minimal configuration of a MSTDNN consists of an input layer, a state layer and an output layer (see Fig. 1). Let us consider a word recognition task where each output unit represents a word. Each state unit represents a small piece of the utterance like phonemes or sub-phonemes. The network is initialized with a window size of one (one connection between an input unit and a unit of the following layer) and one state unit per output unit. The net input of the output units is computed by integrating the weighted activity of the single or multiple state unit(s) over time. The activation of the output units is given by the sigmoid of the net input.

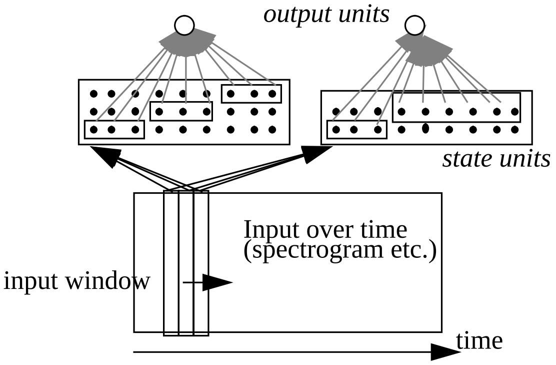

Fig. 1: An example of a simple MSTDNN with an input layer, a state layer and an output layer (consisting of two ouput units). In this example the first output unit is connected with three state units and the second output unit is connected with two state units.

The state units can be regarded as a special type of hidden units because of their very constrained connectivity to the output units.

During training, the size of the input window of the state units as well as the number of state units increases depending on the performance of the corresponding output units. The criterium for the allocation of further resources is derived from the confusion matrix on the training set. At each epoch the mistakes of each output unit are counted. If the counter for output unit j is higher than the mean of all counters, then resources for this particular unit are added. At first, the size of the input window from the input layer to the corresponding state unit is increased by adding one set of random connections. In the next epoch, these new connections are trained together with the already existing connections and the above procedure is applied again.

If the size of the input window of a state unit converges and the corresponding output unit still makes more mistakes than the average unit, then a new state unit is added. The size of the input window of the 'old' state unit is halved to avoid a dramatic increase of the number of trainable parameters. The 'new' state unit receives input from an input window of the same size as the 'old' state unit, but with random connections. From now on, the output unit receives input from both state units.

The allocation of resources is controlled by a simple scheme:

- Adding more resources is easy if the number of connections is small compared to the number of training patterns and gets harder with an increasing number of connections. This avoids hard upper bounds for the network resources.
- All resources that are added to the network are initialized randomly. This reduces the risk that the new resources disturb the learning process. A side-effect is that noise is added which prevents the network from getting stuck in local minima. This noise is reduced afterwards because the new connections are trained together with the already existing connections.
- The maximal size of the input windows also depends on the number of states that model a word. If a word is modeled by many states the state units don't need such a large input window as a state unit that models a whole word.

In case of more than one state unit per output unit the inputs

# CONNECTIONIST ARCHITECTURAL LEARNING FOR HIGH PERFORMANCE CHARACTER AND SPEECH RECOGNITION

*Ulrich Bodenhausen and Stefan Manke*

Computer Science Department, University of Karlsruhe, 7500 Karslruhe 1, FRG, and

School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213

## ABSTRACT

Highly structured neural networks like the Time-Delay Neural Network (TDNN) can achieve very high recognition accuracies in real world applications like handwritten character and speech recognition systems. Achieving the best possible performance greatly depends on the optimization of all structural parameters for the given task and amount of training data. We propose an Automatic Structure Optimization (ASO) algorithm that avoids time-consuming manual optimization and apply it to Multi State Time-Delay Neural Networks, a recent extension of the TDNN. We show that the ASO algorithm can construct efficient architectures in a single training run that achieve very high recognition acuracies for two handwritten character recognition tasks and one speech recognition task.

## 1. INTRODUCTION

Time-Delay Neural Networks (TDNN) [1] with shifted input windows have been successfully applied to speech recognition and handwritten character recognition tasks [2,3,4]. The main feature of the TDNN architecture is the use of a highly structured connectivity between the units of the network. This structured connectivity reduces the number of trainable parameters and also ensures a translation invariant recognition. One reason for the introduction of structure to the network is the relationship between the number of trainable parameters, amount of training data and generalization (see [5, 6] and others). Networks with too many trainable parameters for the given amount of training data learn well, but do not generalize well. This phenomenon is usually called overfitting. With too few trainable parameters, the network fails to learn the training data and performs very poorly on the testing data. Imposing structure into the network can increase the generalization performance by reducing the number of trainable parameters [1].

The use of a highly structured approach leads to the problem of finding the best possible structure for the given task and amount of training data[1]. In order to achieve optimal performance without time-consuming manual optimization of the architecture, we propose an Automatic Structure Optimization (ASO) algorithm that automatically optimizes the structure and the total number of parameters synergetically and also considers the current amount of training data. Rather than starting with a distributed internal representation, the structure of the network is constructed by adding units and connections in order to selectively improve certain parts of the network. At the beginning of the training run the internal representation is completely local and gets more and more distributed in the following optimization process. Only a concept for structuring the network has to be specified before training. The concept for structuring the network is derived from (simple) knowledge about the task (such as invariances).

In this paper we describe an ASO algorithm and apply it to the optimization of Multi State Time-Delay Neural Networks (MSTDNNs), an extension of the TDNN that have been proposed recently [2, 3]. These networks allow the recognition of sequences of ordered events that have to be observed jointly. For example, in many speech recognition systems the recognition of words is decomposed into the recognition of sequences of phonemes or phoneme like units. In handwritten character recognition the recognition of characters can be decomposed into the joined recognition of characteristic strokes etc..

The combination of the proposed ASO algorithm with the MSTDNN was applied successfully to speech recognition and handwritten character recognition tasks with varying amounts of training data.

## 2. CONCEPT OF THE ASO ALGORITHM

The proposed algorithm is based on four principles:

- built-in invariances
- task decomposition
- confusion matrix dependant construction of the network
- early constructive changes of the network architecture.

### 2.1. Built-in Invariances

If there is any knowledge about the task, it should be built into the structure of the network. For speech and handwritten character recognition, a classifier that is robust against temporal distortions is highly desirable. This can be achieved by using shifted input windows over time as in the Time-Delay Neural Network [1]. Shifting the window reduces the number of weights and ensures that the hidden abstractions that are learned are invariant under translations in time.

[1] See Table 2 for an example: Although our data was recorded exactly the same way as proposed in [4], the architecture proposed in [4] did not fit perfectly because of the different number of training characters.