How efficient is the ASO algorithm? The total computational effort for both optimization and training is always greater than the effort for training of a known optimal handtuned architecture. This is not surprising because in this comparison it is assumed that the optimal handtuned architecture is already known, which hardly ever is true. How many training runs does an experienced engineer need to find an equivalent architecture? Observations of colleagues suggest that the number of trials can grow as large as 50, even if the training runs need several days of computation time[1]. Such a high number of trials is only feasible in a research environment and not in a development environment. If we assume that a well-trained engineer needs 5 trials to develop a good architecture and that each trial requires the same computational effort as the best architecture, then the comparison strongly favors the ASO algorithm (see Table 3).

TABLE 3. The relative computational effort for the optimization by the ASO algorithm and training compared to training of a handtuned architecture assuming that 5 trials are needed by the human developer to develop an equivalent architecture.

Task	Relative computational effort of ASO optimization/training
single on-line handwritten digits	29.2%
single on-line handwritten capital letters	27.6%
segmented spoken alphabet recognition	26.4%
connected spoken alphabet recognition	24.6%

5. CONCLUSIONS

The results with the ASO algorithm suggest that the algorithm can construct efficient architectures in a single training run that achieve comparable or better recognition accuracies than manually tuned architectures (see also [1], [2], and[3]). In particular, it has been shown that

- The computational efficiency of the ASO algorithm compares favorably with the computational effort of manual tuning under realistic assumptions. The high optimization efficiency is possible because the allocation of resources starts as early as possible.
- The performance in all tested applications (two on-line handwriting recognition tasks and two speech recognition tasks, see [3]) was comparable or better than the performance of handtuned architectures.
- The ASO algorithm can be used for systems from ~5,000 trainable parameters to systems with ~20,000 trainable parameters. The ASO algorithm should scale well for even larger systems.

The results obtained with the ASO algorithm suggest that automatic optimization algorithms on top of todays speech recognizers can make speech technology easily adaptable to new, customized domains. Automatic optimization algorithms can make the already complex speech technology (that is likely to become even more complex) transparent to developers of software that *includes* speech recognition as *one* feature. If software developers start including speech recognition in their products, there is a chance that end users get introduced and used to this exciting new input modality. The same applies to other new input modalities like on-line handwriting, gesture recognition, or lipreading.

ACKNOWLEDGMENTS

This research was partly funded by grant 413-4001-01IV101S3 from the German Ministry of Science and Technology (BMFT) as a part of the VERBMOBIL project and by a grant from Landesschwerpunktprogramm Neuroinformatik of Baden Würtemberg, Germany. The views and conclusions contained in this document are those of the authors. The authors would like to thank A. Waibel, S. Fahlman, S. Manke, and M. Finke for helpful comments.

REFERENCES

[1] Bodenhausen, U. and Manke, S. Connectionist Architectural Learning for High Performance Character and Speech Recognition. In: *Proceedings ICASSP-93*, Minneapolis, April 1993.

[2] Bodenhausen, U., and Waibel, A. Tuning By Doing: Flexibility Through Automatic Structure Optimization In: Proceedings Eurospeech 93, Berlin, September 1993

[3] Bodenhausen, U. Automatic Structuring of Neural Networks for Spatio-Temporal Real-World Applications, Doctoral Thesis, University of Karlsruhe, 1994

[4] Chen, E. C., and Lippmann, R. P. A Boundary Hunting Radial Basis Function Classifier Which Allocates Centers Constructively. In: *Advances in Neural Information Processing Systems 5*, 1993

[5] Haffner, P., Franzini, M., and Waibel, A. Integrating Time Alignment and Neural Networks for High Performance Continuous Speech Recognition. In *Proceedings of the ICASSP-91*.

[6] Hild, H., and Waibel, A., Connected Letter recognition with a Multi-State Time Delay Neural Network, Neural Information Processing Systems 5, 1993

[7] Lee, K. F. Speech Recognition for Pesonal Computing In: Proceedings 1993 IEEEWorkshop on Automatic Speech Recognition Snowbird, Utah, USA, December 1993

[8] Wilpon, J. G. Applications of Speech Recognition Technology in Telecommunications In: Proceedings 1993 IEEE Workshop on Automatic Speech Recognition Snowbird, Utah, USA, December 1993

1. The MS-TDNN system that was used for performance comparison was probably trained more than 50 times. However, no detailed records are available.

Both systems were trained with standard Back-Propagation and were validated and tested with exactly the same data sets. The results are summarized in Table 1. The manually tuned MS-TDNN system performs comparably (97.5% vs. 97.4%) as the MS-TDNN/ASO system. The small difference between these two results is surprising given the fact that 1.) no phoneme labels were used, 2.) a simpler duration control was used and 3.) the great amount of manual tuning that was applied to the other system over several years.

TABLE 1. Performance comparisons on the connected spoken alphabet recognition task (500 training sentences, 100 validation sentences, and 400 test sentences) under comparable conditions.

System description	%Word accuracy
Manually tuned MS-TDNN, trained with phoneme labels, handtuned number of states per phoneme, number of hidden units and window sizes, Gaussian modeling of duration constraints	97.5%
MS-TDNN optimized by ASO, trained with letter labels only, automatic optimization of the number of states per letter, number of hidden units and window sizes, simple duration control	97.4%

4. SCALING OF ASO DEPENDING ON THE SIZE OF THE SYSTEM

The ASO algorithm was developed and tested with classification tasks with 10 (on-line handwritten digits) to 27 classes (connected spoken letter recognition) so far. The smallest system (for on-line handwritten digits) was initialized with 10 state units (one state unit for each digit), no hidden units and an input window of one frame, each consisting of 8 features. This makes a total of ((8 * 10) + 10) = 90 independently trainable weights for the initial architecture[1]. After optimization of the architecture, the average architecture had an average number of 2.6 states per digit, 3.4 hidden units and a window width of 13.2 for the direct weights from the input to the state units, a width of 8.7 for the windows from the input to the state units, and an average width of 7.8 for the weights from the hidden units to the state units. The average number of independently trainable parameters was 3956 in 10 different optimization runs. This was the smallest problem that the ASO algorithm was applied to so far.

1. The bias weights are left out in this comparison for simplicity.

When the ASO algorithm was applied to the connected spoken letter recognition task with 27 classes (26 letters plus silence), the system was again initialized with one state per letter, no hidden units and input windows of one frame, each consisting of 16 features (spectral coefficients). This makes a total of ((27 * 16) + 27) = 459 independently trainable connections. In an average of five optimization runs, the ASO algorithm constructed networks from 17,000 connections to 26,000 connections, with an average of 19,231 connections. This was the largest problem that the ASO algorithm was applied to so far.

In all applications tested with ASO so far, the generalization performance was comparable or better than the performance of a handtuned system [3].

4.1 Computational Efficiency of ASO

For an evaluation of the efficiency of the ASO algorithm it is necessary to compare the total computational effort for both optimization of the architecture and the training of the weights with the effort for training of an already handtuned architecture. This comparison can only be based on the total computation time on a certain machine. Table 2 shows the relative computational effort for four tasks including the connected letter recognition task from the last section. The handtuned architecture is assumed to be known and the computational effort for training of this architecture is set to 100%. The relative computational effort varies from 146% for the classification of single on-line handwritten digits to 122% for the connected spoken letter recognition task described in the last section. Fortunately, the relative computational effort is high (146%) for the smallest task and lower (123%) for the largest task.

TABLE 2. The relative computational effort for the optimization by the ASO algorithm and training compared to training of a handtuned architecture. The effort for the handtuned architecture (assumed to be known in advance) is set to 100%

Task	# Experiments	Relative computational effort of ASO optimization/training
single on-line handwritten digits	10	146%
single on-line handwritten capital letters	5	138%
segmented spoken alphabet recognition	10	132%
connected spoken alphabet recognition	5	123%

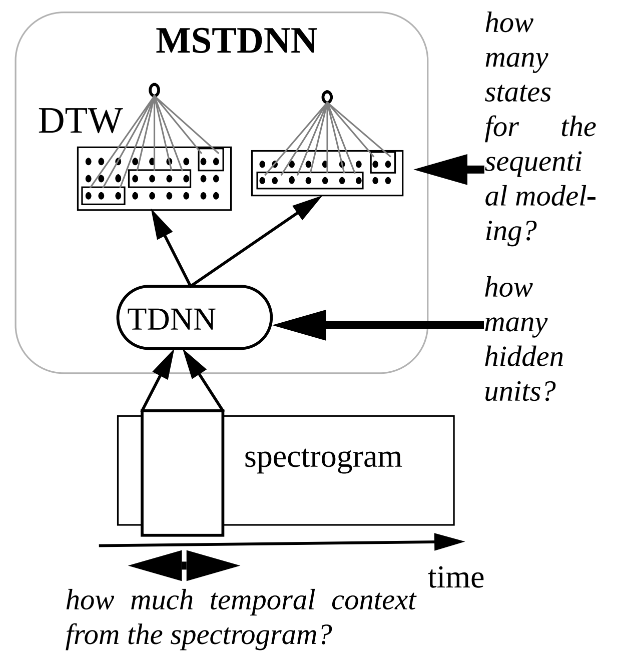
MSTDNN
DTW
how many states for the sequential modeling?
TDNN
how many hidden units?
spectrogram
time
how much temporal context from the spectrogram?

Fig. 1: Overview of the speech recognizer (Multi-State Time Delay Neural Network [MSTDNN] which combines a Time Delay Neural Network [TDNN] with Dynamic Time Warping [DTW]) and the relevant architectural parameters for the optimization process: 1.) How much temporal context is needed from spectrogram? 2.) How many hidden units are necessary for the mapping? 3.) How many states are necessary for the sequential modeling?

2. THE AUTOMATIC STRUCTURE OPTIMIZATION ALGORITHM (ASO)

For the application of neural networks to speech recognition all of the following architectural parameters have to be well adapted to the task and the given amount of training data (see Fig. 1):

- the number of hidden units,
- the size of input windows and
- the number of states that model an acoustic event.

The *Automatic Structure Optimization* (ASO) algorithm ([1], [2], [3]) *automatically adapts* all of these architectural parameters to the given task and amount of training data. The algorithm uses a constructive approach to optimize the architecture of the system for best possible generalization performance for the task and the amount of training data. The ASO algorithm uses the following tuning strategies:

- The confusion matrix on the training data is evaluated to selectively improve certain parts of the neural network.
- The number of states is increased if the acoustic modeling is too complex for the given number of states.
- Hidden units are allocated to specifically solve pairwise confusions (class “A” is confused with class “B” and vice versa) which are caused by inadequate decision boundaries. The approach is similar to the *Boundary Hunting Radial Basis Function* classifier [4], but it allocates hidden units with sigmoid activation functions instead of radial basis functions.

Unlike the human developer, the ASO algorithm starts making decisions about resource allocations very early in the training run, i.e. it is tuning the architecture while the network is learning the task (“tuning by doing”). This allows the algorithm to complete the optimization process in a single training run.

3. RECOGNITION PERFORMANCE WITH ASO

The ASO algorithm was applied to a speaker dependent connected letter recognition task[1]. Letter recognition is not considered a typical customized application, but the task provides a good test for speech recognition systems because of the high confusability of spoken letters. In addition, good manually tuned systems exist for performance comparison. The ASO system was compared with a system developed by Haffner and Hild for the same database [5], [6]. Some of the recent improvements like sentence level training were switched off to allow for comparable experimental conditions[2]. However, some differences between the systems remain:

- The handtuned MS-TDNN was trained with labels for phonemes. In practical customized applications of the ASO algorithms these labels are unknown and the user should not be required to label the data. Thus, the ASO algorithm was used without these phoneme labels in order to investigate its performance under realistic conditions.
- The handtuned MS-TDNN used a better duration control based on Gaussian modeling of letter/phoneme durations [6]. This improved duration modeling was not included in the ASO system.

1. For applications to other tasks and performance measurements, see [1], [2], and [3].

2. The performance of Hermann Hild’s MS-TDNN system with sentence level training is 98.5% on the test set (speaker MJMT).

AUTOMATIC CONSTRUCTION OF NEURAL NETWORKS FOR SPECIAL PURPOSE SPEECH RECOGNITION SYSTEMS

Ulrich Bodenhausen and Hermann Hild

Interactive Systems Laboratories, University of Karlsruhe,

Computer Science Department, ILKD, 76131 Karlsruhe, Germany

ABSTRACT

The successful application of speech recognition systems to new domains greatly depends on the tuning of the architecture to the new task, especially if the amount of training data is small. For example, the application of Multi-Layer Perceptrons (MLPs) to speech recognition requires the optimization of the number of hidden units, the size of the input windows over time and the number of states that model an acoustic event. Previously, we have proposed the Automatic Structure Optimization algorithm (ASO) that optimizes all of the above architectural parameters automatically. In this paper we 1.) present results for the succesful application of the ASO algorithm to connected spoken letter recognition, 2.) show the suitability of the algorithm for various sizes of the system and 3.) analyze the computational efficiency of the automatic optimization process for four different tasks.

1. INTRODUCTION

Despite the aim to develop a general purpose, speaker independent, very large vocabulary, spontaneous speech recognizer there is a considerable number of applications that require the best possible performance on small, well defined, and customized domains. For these applications, manual tuning of the architecture (for example optimization of the number of states, the number of hidden units and the width of the input windows of a neural network) is too costly and not tolerable because each application requires its own optimization. Many of todays speech recognition systems are very powerful, but the complexity of these systems is such that these systems are far from being intuitive and easy to use for the developer. In general it is not possible to develop applications on top of the technology without understanding the underlying speech algorithms. This makes quick prototyping impossible, which is very important for the creation of new products and services [7], [8]. Developers of software that *includes* speech recognition should not be required to invest months or years in an understanding of details of speech recognition technology or in the tuning of these systems.

The current situation of speech technology is rather peculiar: Complex systems have been developed, but the final user is demanding even more general systems (with a much larger vocabulary, even higher word accuracy and less restrictions concerning speaking style) and developers of customized applications are demanding simpler systems that are much easier adaptable to small domains and allow quicker prototyping. The result is that end users do not accept the current off-the-shelf general purpose systems and that developers of customized applications do not dare to include speech recognition into their products.

How can this situation be improved? Further development of general purpose systems demands a considerable effort and expertise in research (acoustics, search, language, ...) as well as the availability of considerable computational resources. It is important to pursue these developments because the current results suggest that real general purpose speech recognition will actually be possible in the future. In the meantime it is important to make end users accustomed to this new technology, both to its advantages and peculiarities. This can be done by small, customized applications or applications where speech is not the primary input modality. This paper

- summarizes the development of an automatic optimization algorithm on top of state-of-the-art speech technology which make this technology easily usable for developers of customized applications;
- presents results from the successful application of this algorithm to connected letter recognition;
- discusses the suitability of the algorithm depending on the size of the system;
- presents an empirical analysis of the computational efficiency of the automatic optimization process for several tasks compared to manual tuning.