# LEARNING THE ARCHITECTURE OF NEURAL NETWORKS FOR SPEECH RECOGNITION

*Ulrich Bodenhausen* *Alex Waibel*

School of Computer Science, Carnegie Mellon University
Pittsburgh, Pennsylvania 15213-3890, USA

**ABSTRACT**

Multi-layered neural networks have been recently proposed for speech recognition systems. In many approaches a set of trainable connections with different time-delays enables the networks to discover temporal relationships between acoustic-phonetic events [1, 2, 3]. In these networks, weights are automatically trained but the architecture of the network (time-delays, number of connections and number of units) have to be predetermined by laborious experiments [3, 4]. The Tempo 2 network proposed in this paper automatically learns the architecture as well. The application to phoneme classification shows that this adaptive architecture can approach the performance of a carefully handtuned TDNN and leads to more compact networks. The learning rule may also be useful to learn temporal and rhythmic relationships in speech.

## I. INTRODUCTION

The TDNN [2] and many other neural networks applied to speech recognition use spectrograms as input to the network. The spectrograms are computed with a certain frame-rate and fed into a neural network where each input unit represents one spectral frequency. The frame-rate of the spectrogram determines how often the activities of the input units of the TDNN are updated. Time-delays are used to let the units of the following layer capture some temporal context of the activations. These time-delays are implemented as hat-shaped input windows. For example, each connection in the TDNN transmits an input-window that covers exactly one time-frame. A unit j that is connected with unit i by a connection with the delay n is only receiving information about the activity of unit i n time-steps ago (see Fig 1). A set of connections with consecutive time-delays is used to let each unit gather a certain amount of temporal context (see Fig. 2). With this configuration it is necessary to specify:

- *how much* temporal context should each unit receive;
- *how many* independently trainable weights are needed to capture the temporal dynamics within this given temporal context;
- *what* temporal context should a unit receive (how far should each unit look back into the past).

In the standard TDNN, the units of the first hidden layer receive a temporal context of three consecutive time-frames via three

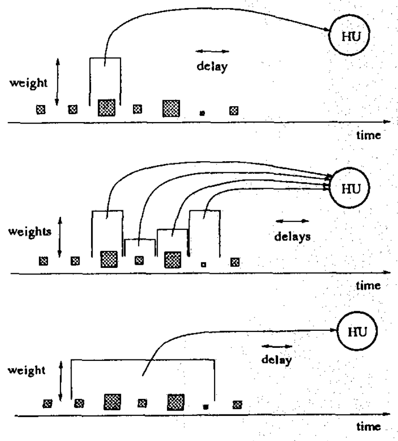

Figure 1: Different choices of input windows: Top: A single hat-shaped input window that covers one timeframe. Middle: A set of hat-shaped input windows with consecutive time-delays cover more temporal context. Bottom: A single hat-shaped input window that covers 5 time-frames. (The boxes represent the activations of the sending unit; a tall box represents a high activity.)

connections with independently trainable weights and the time-delays 0, 1 and 2. The units of the second hidden layer receive a temporal context of five time-frames via five connections with independently trainable weights and the time-delays 0 .. 4.

The choice of the parameters mentioned above directly determines the number of free parameters in the network and therefore affects the amount of data that is needed for training. A network with a larger number of independent parameters needs more training data. It would also be possible to let one connection transmit more than one time-frame (see Fig. 1). This would lead to a smaller number of free parameters in the network but could limit the ability to capture the temporal dynamics of the input token.

Various experiments have been run to determine the optimal architecture for phoneme classification networks [3, 4]. The Tempo 2 network proposed in this work optimizes the architecture by itself during learning. It is a feedforward, multilayer neural network with *adaptive* weights, *adaptive* time-delays, *adaptive* widths of

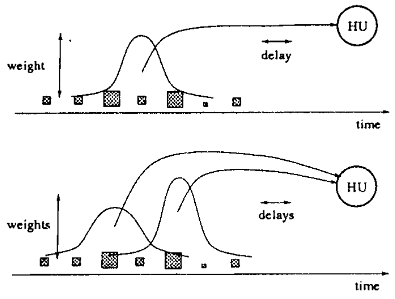

Figure 2: Top: A single gaussian-shaped input window covers a certain amount of temporal context. Bottom: Two gaussian shaped input windows with different weights, delays and widths are able to capture more complex temporal dynamics of the input signal.

gaussian shaped input-windows (see Fig. 2) and a variable number of connections. A unit j in the network is activated by input from a gaussian input-window over time centered around (t - d) and standard deviation $\sigma$, where d (the time-delay) and $\sigma$ (the width of the window) are to be learned. This means that the center and widths of each input-window can be adjusted by learning rules. The gaussian input-window allows the derivation of learning rules for delays and widths in the same way that is used for the weights in a standard Back-Propagation network[5]. The architecture is able to adapt to the time-scale of the input tokens and there is no need to specify the architectural constraints mentioned above:

- The width $\sigma$ of the gaussian shaped input window determines *how much* temporal context each connection transmits.
- The number of connections is allowed to increase during learning. Thus, the training algorithm determines *how many* independently trainable weights are needed to capture the temporal dynamics within the required temporal context .
- The adaptive time-delays determine *what* temporal context each unit receives (that means how far each unit looks back into the past).

Adaptive time-delays were already realized in the previously proposed Tempo 1 network [6, 7] which was tested with sequences of pictures and rhythms.

## II. THE TEMPO 2 ALGORITHM

In the Tempo 2 network a unit j at time t is activated by input from a gaussian shaped input-window centered around (t-d) and standard deviation $\sigma$, where d (the time-delays) and $\sigma$ (the width of the input-window) are to be learned. (Other windows are possible. The function describing the shape of the window has to be smooth.) The input of unit j at time t, $x(t)_j$, is

$$x(t)_j = \sum_{\tau=0}^{t} \sum_{k} y_k(\tau)\theta(\tau, t, d_{jk}, \sigma_{jk})w_{jk}$$

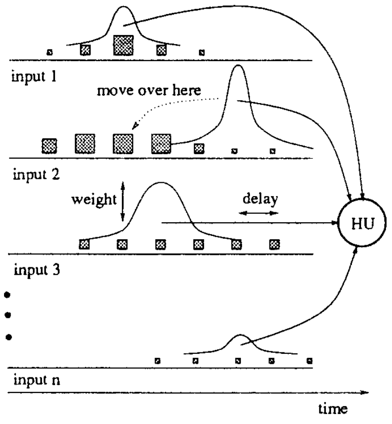

Figure 3: The input to one unit in the Tempo 2 network.

with $\theta(\tau, t, d_{jk}, \sigma_{jk})$ representing the gaussian input window given by

$$\theta(\tau, t, d_{jk}, \sigma_{jk}) = \frac{1}{\sqrt{2\pi}\sigma_{jk}} e^{(\tau - t + d_{jk})^2 / 2\sigma_{jk}^2}$$

where $y_k$ is the output of the previous sending unit and $w_{jk}$, $d_{jk}$ and $\sigma_{jk}$ are the weights, delays and widths on its connections, respectively.

This approach is partly motivated by neurophysiology and mathematics. In the brain, a spike that is sent by a neuron via an axon is not received as a spike by the receiving cell. Rather, the postsynaptic potential has a short rise and a long tail. Let us assume a situation with two neurons. Neuron A fires at time t-d, where d is the time that the signal needs to travel along the connection and to activate neuron B. Neuron B is activated mostly at time t, but the postsynaptic potential will decrease slowly and neuron B will get some input at time t+1, some smaller input at time t+2 and so on. Functionally, a spike is smeared over time and this provides some "local memory".

For our simulations we simulate this behavior by allowing the receiving unit to be activated by the weighted sum of activations around an input centered at time t-d. If the sending unit ("neuron A") was activated at time t-d, then the receiving unit ("neuron B") will be activated mostly at time t, will be less activated at time t+1, and so on. In our case, the input-window function also allows the receiving unit to be (less) activated at times t-1, t-2 etc.. This symmetric 'behavior' enables us to formulate a learning rule that can increase and decrease time-delays.

The gaussian input-window has the advantage that it provides some robustness against temporally misaligned input tokens. By looking at Fig. 2 or Fig. 3 it is obvious that small misalignments of the input signal do not change the input of the receiving unit significantly. The robustness is dependent on the width of the window. Therefore a wide window would make the input of the receiving unit more robust against signals shifted in time, but would also reduce the time-resolution of the unit. This is another

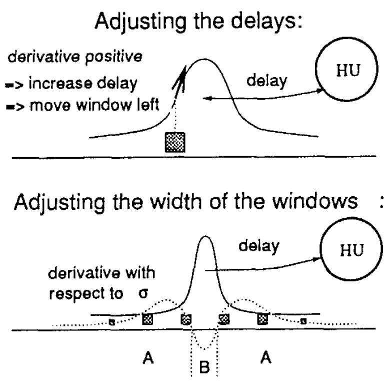

Figure 4: A graphical explanation of the learning rules for delays and widths: The derivative of the gaussian input-window with respect to time is used for adjusting the time-delays. The derivative with respect to $\sigma$ (dotted line) is used for adjusting the width of the window. A majority of activation in area A will cause the window to grow. A majority of activity in area B will cause the window to shrink.

reason for the implementation of a learning rule that adjusts the width of the input-windows of each connection.

With this gaussian input-window over time, it is possible to compute how the input of unit j would change if the delay of a connection or the width of the input-window were changed. The formalism is the same as for the derivation of the learning rules for the weights in a standard Back-Propagation network. The change of a delay is proportional to the derivative of the output error with respect to the delay. The change of the width is proportional to the derivative of the error with respect to the width of the input-window. As in all Back-Propagation networks, the error is propagated back to the hidden layer. The learning rules for weights $w_{ji}$, delays $d_{ji}$ and widths $\sigma_{ji}$ were derived from

$$\Delta w_{ji} = -\epsilon_1 \frac{\partial E}{\partial w_{ji}}$$

$$\Delta d_{ji} = -\epsilon_2 \frac{\partial E}{\partial d_{ji}}$$

$$\Delta \sigma_{ji} = -\epsilon_3 \frac{\partial E}{\partial \sigma_{ji}}$$

where $\epsilon_1$, $\epsilon_2$ and $\epsilon_3$ are the learning rates and E is the error. As in the derivation of the standard Back-Propagation learning rules, the chain rule is applied (z = w, d, $\sigma$):

$$\frac{\partial E}{\partial z_{ji}} = \frac{\partial E}{\partial x(t)_j} \frac{\partial x(t)_j}{\partial z_{ji}}$$

where $\frac{\partial E}{\partial x(t)_j}$ is the same in the learning rules for weights, delays and widths. The partial derivatives of the input with respect to the parameters of the connections are computed as follows:

$$\frac{\partial x(t)_j}{\partial w_{ji}} = \sum_{\tau=0}^{t} y_i(\tau)\theta(\tau, t, d_{ji}, \sigma_{ji})$$

$$\frac{\partial x(t)_j}{\partial d_{ji}} = \sum_{\tau=0}^{t} y_i(\tau) w_{ji} \frac{\partial}{\partial d_{ji}} \theta(\tau, t, d_{ji}, \sigma_{ji})$$

$$\frac{\partial x(t)_j}{\partial \sigma_{ji}} = \sum_{\tau=0}^{t} y_i(\tau) w_{ji} \frac{\partial}{\partial \sigma_{ji}} \theta(\tau, t, d_{ji}, \sigma_{ji})$$

## IIa. Adding New Connections

Learning algorithms for neural networks that add or delete hidden units have recently been proposed [8, 9]. In our Tempo 2 network we currently add connections to the already existing ones.

During learning, the Tempo 2 network starts with one connection between two units. Depending on the task this may be insufficient and it would be desirable to add new connections where more connections are *needed*. As we pointed out before, the number of connections determines the amount of training data which is needed for good generalization. This means that new connections should be only added if the progress of learning is slowed down. On the other hand, connections should be added *before* many epochs are used to fine-tune the parameters. If new resources are allocated in an already fine-tuned network this may ruin the fine-tuning that is already made.

New connections are added by splitting already existing connections and afterwards training them independently. The rule for splitting a connection is motivated by observations during training runs. It was observed that input-windows started moving backwards and forwards (that means the time-delays changed) after a certain level of performance was reached. This can be interpreted as inconsistent time-delays which might be caused by temporal variability of certain features in the speech tokens. During training we compute the standard deviations of all delay changes and compare them with a threshold:

$$if \sum_{alltokens} (\Delta d_{ji}(token) - \frac{\sum_{alltokens} |\Delta d_{ji}|}{\#tokens})^2 > threshold$$

then split connection ji.

The network is initialized with random weights and delays and constant widths $\sigma$ of the input-windows.

## III. APPLICATION

We applied the Tempo 2 network to phoneme classification (/b/, /d/ and /g/ classification). 783 phonemes were used for training and 759 tokens were user for testing. In the database, no preselection of tokens was performed. The tokens are extracted from entire word utterances by a single speaker.

The network is initialized with random weights, random delays, constant widths of the input windows and one connection between two units. During training, weights, delays, widths and the number of connections are changed. We found that the learning rates

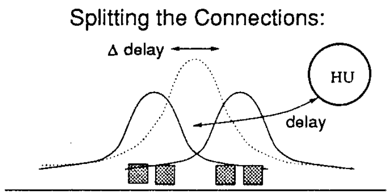

Figure 5: Splitting of a connection. The dotted line represents the 'old' window and the solid lines represent the two windows after splitting, respectively. A connection is split if its input-window starts moving backwards and forwards during training.

( $\epsilon_1$, $\epsilon_2$ and $\epsilon_3$ ) for the parameters weights, delays and widths can not be chosen independently of each other [6, 7]. If the weights are updated with a large step size then delays and widths have to be updated with a large step size, too. Otherwise the parameter set with the large step size is learned "egoistically".

In order to evaluate the usefulness of each adaptive parameter, the network was trained and tested with a variety of combinations of constant and adaptive parameters (see Table 1). The preliminary results on 'bdg' classification reported in Table 1 already approach the results obtained with a carefully handtuned TDNN on the same task. However, no handtuning of time-delays and window-sizes is needed. Additionally, the evolving network performs the task with approximately a fourth of the connections needed in a TDNN.

## IV. CONCLUSION

Our results suggest that it is possible to learn the architecture of neural networks for speech recognition systems. Although handtuned networks perform very well it is desirable to have an automatic optimization process because different speakers, databases or languages might require different network architectures.

The evolving networks are more compact than the handtuned counterparts. Currently the number of hidden units in our network is constant. We intend to incorporate an algorithm that also adds hidden units during training (like [8, 9]).

With the Tempo 2 algorithm we proposed a training algorithm for neural networks that trains the temporal parameters of the network (delays and widths of the input windows) as well as the weights. A comparison of the performances with one adaptive parameter set (either weights, delays or widths) shows that the main parameters are the weights. Delays and widths seem to be of lesser importance, but in combination with the weights the temporal parameters can improve performance, especially generalization. A Tempo 2 network with trained delays and widths and *random* weights can classify 70% of the phonemes correctly. This suggests that learning temporal parameters is effective. At higher levels of processing temporal parameters may be significant to learn temporal and rhythmic relationships as they occur in speech and other tasks.

| adaptive parameters | constant parameters | Training Set | Testing Set |
|---|---|---|---|
| weights | delays, widths | 93.2% | 89.3% |
| delays | weights, widths | 64.0% | 63.0% |
| widths | weights, delays | 63.5% | 61.8% |
| delays, widths | weights | 70.0% | 68.6% |
| weights, delays | widths | 98.3% | 97.8% |
| weights, delays, widths | - | 98.8% | 98.0% |

Table 1: /b/, /d/ and /g/ classification performance with 8 hidden units in one hidden layer.

## V. ACKNOWLEDGEMENT

The authors gratefully acknowledge the support of the McDonnel-Pew Foundation (Cognitive Neuroscience Program) and ATR Interpreting Telephony Research Laboratories.

## REFERENCES

[1] Tank, D. W. and Hopfield, J. J.. Neural Computation By Concentrating Information In Time. In *Proceedings National Academy of Sciences*, pages 1896–1900, April 1987.

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

[3] Lang, K. J., Hinton, G. E. and Waibel, A. H.. A Time-Delay Neural Network Architecture For Speech Recognition. *Neural Networks Journal*, 1990.

[4] Kamm, C. E.. Effects Of Neural Network Input Span On Phoneme Classification. In *Proceedings of the International Joint Conference on Neural Networks*, June 1990.

[5] Rumelhart, D. E., Hinton, G. E. and Williams, R. J.. Learning Internal Representations By Error Propagation. In J.L. McClelland and D.E. Rumelhart, editors, *Parallel Distributed Processing; Explorations in the Microstructure of Cognition*, chapter 8, pages 318–362. MIT Press, Cambridge, MA, 1986.

[6] Bodenhausen, U.. The Tempo Algorithm: Learning In A Neural Network With Adaptive Time-Delays. *Proceedings of the International Conference on Neural Networks*, January 1990.

[7] Bodenhausen, U.. Learning Internal Representations Of Pattern Sequences In A Neural Network With Adaptive Time-Delays. *Proceedings of the International Conference on Neural Networks*, June 1990.

[8] Fahlman, S. and Lebiere, C.. The Cascade-Correlation Learning Architecture. In *Advances in Neural Information Processing Systems*. Morgan Kaufmann, 1990.

[9] Hanson, S. J.. Meiosis Networks. In *Advances in Neural Information Processing Systems*. Morgan Kaufmann, 1990.