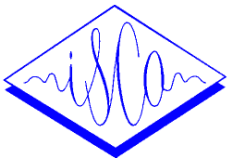

Fourth ISCA ITRW on
Speech Synthesis (SSW-4))
Perthshire, Scotland
August 29 - September 1, 2001

SEGMENTAL DURATION CONTROL WITH ASYMMETRIC CAUSAL RETRO-CAUSAL NEURAL NETWORKS

Çağlayan Erdem, Hans Georg Zimmermann
Siemens AG , Corporate Technology, D- 81730 Munich, Germany
email: {Caglayan.Erdem,Georg.Zimmermann}@mchp.siemens.de

Abstract

The generation of pleasant prosody parameters is very important for speech synthesis. A Prosody generation unit can be seen as a dynamical system. In this paper sophisticated time-delay recurrent neural network (NN) topologies are presented which can be used for the modeling of dynamical systems. Within the prosody prediction task left and right context information is known to influence the prediction of prosody control parameters. This can be modeled by causal-retro-causal information flows [1]. Since information being available during training is partially unavailable during application, there is a structural switching from training to application. This structural change of the information flow is handled by two asymmetric architectures.

These proposed new architectures allow the integration of further a priori knowledge. By this we are able to improve the performance of our duration control unit within our text-to-speech (TTS) system *Papageno*.

1. Introduction

A natural sounding synthetic voice is very important to achieve customers acceptance. A high quality prosody module has a big impact on the naturalness of a synthetic voice. Our acoustic prosody module consists of a duration control and a f0-contour unit. Both modules build up the acoustic prosody control unit as depicted in fig. 1.

In *Papageno* the segmental duration control is handled first. Those segmental durations generated by the duration control unit are afterwards used as inputs to the f0-generation unit. Continuous positional information of syllables is derived from them, which are important for the f0-generation task.

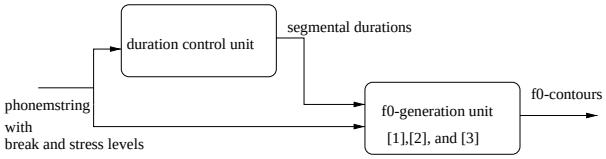

Figure 1: acoustic prosody unit

A segmental duration module controls the rhythm of a synthetic voice and the known effect of final lengthening. A duration control unit with better performance should have a positive effect on the performance of the f0-control unit.

As our TTS system is a data-driven concatenative system the duration control module also utilizes data-driven methods. State-of-the-art techniques for data-driven duration control are *classification and regression trees (CART)* [4] and NN [5] , [6], and [7]. Within the CART modeling a statistic for the segmental duration is built up for each node which is then used during synthesis. In a first approach our duration control module used a CART like method. The parameters for the nodes are derived from triphon clusters obtained by a tree-based clustering algorithm provided by standard clustering within HTK [8]. This approach will not be explained in detail. In the following it is called *stat*.

In a first run this statistic method was enhanced by a more sophisticated statistical approach considering larger contextual information on syllable and word level. But there is a dilemma between robustness and significance of the statistics as *Papageno* utilizes a restricted speech database for its fundus (about 1k sentences). Therefore the NN approach is employed to solve this dilemma due to his generalization property.

Similar to the f0-contour prediction task [1] the duration control unit uses left (past) and right (future) contextual information to establish the prediction.

This paper consists of four parts. In section 2 basics (finite unfolding, error correction) and a new partial retro-causal expansion for the integration of the structural switching are discussed. In section 3 asymmetric architectures are discussed which overcome the structural switching of the information flow. Section 4 depicts the application of the asymmetric modeling and gives results.

2. NN Architecture

In memory dependent dynamical systems the state s_t of the system at a certain time step depends on previous states s_{t-1} and on the input to the system u_t. This results in a recursion as depicted in fig. 2.

Such a system can be described by eq. (1) and eq. (2) (see [1])

$$s_t = f(s_{t-1}, u_t) \quad (1)$$

$$y_t = g(s_t) \quad (2)$$

with s_t: state of the system at time step t, u_t: external input to the system at time step t, and f: function describing the state transition. The output y_t of the system is given by eq. (2) and depends on s_t. This dependence is given by g.

To describe a given dynamical process by eq. (1) and eq. (2),

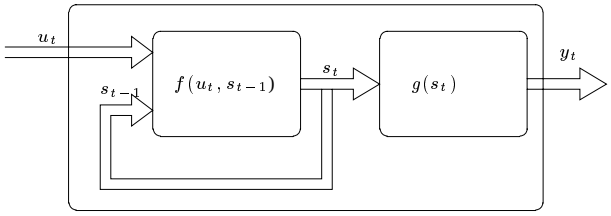

Figure 2: recursive system

the functions f and g need to be determined. If f and g cannot be found analytically the following optimization problem can be formulated:

$$\frac{1}{T}\sum_{t=1}^{T}(y_t - y_t^d)^2 \rightarrow \min_{f,g} \tag{3}$$

with T: number of observed data points, and y_t^d: observed data point at time step t.

A NN realization of eq. (1), eq. (2) and eq. (3) is formulated in eq. (4), eq. (5) and eq. (6):

$$\begin{aligned} s_t &= tanh(As_{t-1} + Bu_t) && (4)\\ y_t &= Cs_t && (5) \end{aligned}$$

$$\frac{1}{T}\sum_{t=1}^{T}(y_t - y_t^d)^2 \longrightarrow \min_{A,B,C} \tag{6}$$

with A,B, and C denoting weight matrices. This parameter optimization task is solved be adjusting A,B, and C such that eq. (6) is minimal.

If A,B, and C are shared weights then fig. 2 represents a finite apprximation of eq. (4) and eq. (5). The solution technique displayed here is called *finite unfolding intime* [9], [10]. State clusters s_{t+i} use $tanh$ as activation functions.

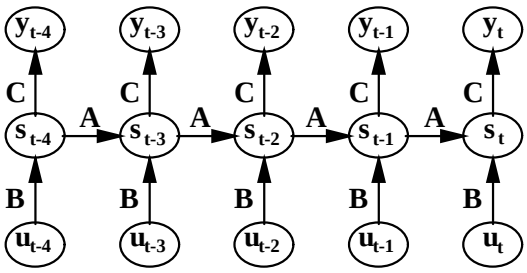

Figure 3: Finite Unfolding NN.

Finite unfolding in time generally causes long chains of neurons from input to output. In such long chains the drying out of the error signal occurs because of the asymptotic behavior of $tanh$. Therefore, *Vario Eta* [11] should be used as learning rule. Parameters A,B, and C are utilized for each time step. Incorporating more time step within the architecture does not effect the amount of parameter.

2.1. ECNN

For an explicit modeling of missing inputs (unconsidered, unknown, unavailable, not-measured) we extend the transition equation of the dynamical system eq. (1) by the model error $y_{t-1} - y_{t-1}^d$.

$$\begin{aligned} s_t &= f(s_{t-1}, u_t, y_{t-1} - y_{t-1}^d) && (7)\\ y_t &= g(s_t) && (8) \end{aligned}$$

a NN realization of these dependencies in eq. (7) and eq. (8) might be:

$$\begin{aligned} s_t &= \tanh(As_{t-1} + Bu_t + D\tanh(Cs_{t-1} - y_{t-1}^d)) && (9)\\ y_t &= Cs_t && (10) \end{aligned}$$

$$\frac{1}{T}\sum_{t=1}^{T}(y_t - y_t^d)^2 \longrightarrow \min_{A,B,C,D} \tag{11}$$

This NN is depicted in figure 4 again using shared weight matrices A,B,C,and D and *finite unfolding in time* as solution technique.

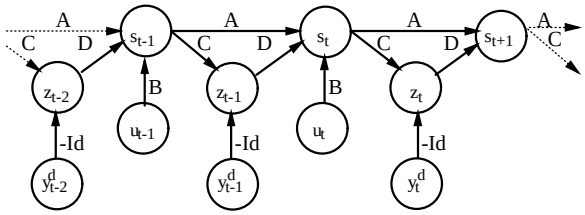

Figure 4: Error Correction NN.

In figure 4 an autonomous part of the dynamical system is denoted by the matrix A carrying state information of the dynamics between neighboring state clusters. This path allows the mapping of long-term forecasts.

Matrix C transforms the state s_t to its expectation y_t. While matrix B introduces external information u_t to the system, D propagates the model error (the expectation y_t being compensated by the observation y_t^d) to cluster s_t. The path

$$s_{t-1} \rightarrow C \rightarrow z_{t-1} \rightarrow D \rightarrow s_t$$

allows to map local structures as shocks or short term effects. The z-clusters represent the output clusters of the NN architecture. In cluster z_{t-1} the difference z_{t-1}= Cs_{t-1} - y_{t-1}^d (forecast error) between the expectation of the NN and the observation y_{t-1}^d is computed. Note that y_{t-1}^d is propagated by $-Id$ to z_{t-1}.

This difference has its optimum in zero, since this denotes no forecast error. Having no forecast errors results in a perfect description of the dynamics. So the target vector is set to zero during training.

If there is no mismatch between expectation and observation then no further information is propagated by matrix D to state s_t and we almost obtain a simple finite unfolding NN.

An existing mismatch delivers further input information to state s_t. This information is used during training for the adaption of parameters. By this error correction principle we obtain in z_t an

internal vector driving the transition of the system state together with external input u_{t-1} and previous states.

These internal vectors generate the error flow. It is computed at each output cluster time step of the unfolding.

If the internal autoregressive part coded in A (see fig. 2) and all external driving forces of a dynamics are known, it would be possible to give a perfect description of the dynamical system. But if it is not possible to identify the dynamics due to missing or unknown externals or noise, the last model error is an indicator of the models misspecification. Since the model error is used as a measure of unexpected shocks, the learning of false dependencies is lowered and models generalization ability is improved [12].

Incorporating information flow from the right to the left captures retro-causal dependencies. If this is handled symmetrically over all time steps this modeling results in fix point recurrencies as depicted in [1]. [1] presents state-of-the-art causal-retro-causal modeling. Those mentioned fix point recurrencies complicate computation of the resulting CRCECNN. Therefore this paper proposes a partial symmetric expansion in the following subsection which results in a partial CRCECNN (P-CRCECNN).

2.2. P-CRCECNN

The causal structure of ECNN is extended to a combined causal-retro-causal modeling. Thus by the modeling we are able to include future information of text to speak to speech synthesis.

The output equation of ECNN is extended by retro-causal modeling:

$$s_t = f(s_{t-1}, u_t, y_{t-1} - y^d_{t-1}) \quad (12)$$

$$r_t = g(r_{t+1}, u_t, y_t - y^d_t) \quad (13)$$

$$y_t = h(s_t, r_{t+1}) \quad (14)$$

One NN realization of these transition equations might be:

$$s_t = \tanh(As_{t-1} + Bu_t + D\tanh(Cs_{t-1} - y^d_{t-1})) \quad (15)$$

$$r_t = \tanh(A^R r_{t+1} + B^R u_t + D^R \tanh(C^R r_{t+1} - y^d_t)) \quad (16)$$

The model expectation of the z-cluster is determined by

$$y_t = \begin{cases} Cs_t & : \quad t < 0 \\ Cs_t + C^R r_{t+1} & : \quad t = 0 \\ C^R r_{t+1} & : \quad t > 0 \end{cases} \quad (17)$$

This NN utilizing shared weights and finite unfolding is depicted in figure 5.

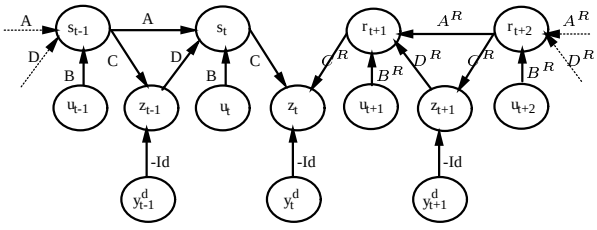

Figure 5: P-CRCECNN

Both information flows are coupled within the present time step. This architecture does not contain fix-point recurrent loops, which might cause instabilities during training.

3. Structural Switching

The structural switching will be explained using fig. 5. During training all segment durations modeled as observations y^d_{t+i} are known. But within the application there are no observation available for $i \geq 0$, because they are not predicted yet. For $i < 0$ predictions of the NN are re-utilized as observations.

Because of this mismatch between training and application the retrocausal information flow has to be treated in a specific way. In the following to different ways of asymmetric P-CRCECNN are explained which overcome this mismatch.

In fig. 6 the idea of removing connections after training is depicted. The dotted connections C^R and D^R are trained. So the architecture is the same as depicted in fig. 5 during training. But within the application connections C^R and D^R are removed. The resulting architecture is then a finite unfolding in time without the error correction principle for the retro-causal information flow during application.

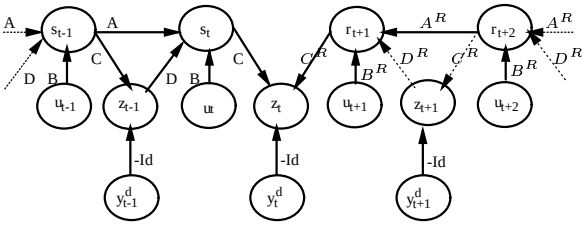

Figure 6: P-CRCECNN_removed

This NN realization is also determined by equations (12)... (14) for training.

Thus, during application, equation r_t is given by:

$$r_{t+1} = \tanh(A^R r_{t+2} + B^R u_{t+1}) \quad (18)$$

and y_t by:

$$y_t = \begin{cases} Cs_t & : \quad t < 0 \\ Cs_t + C^R r_{t+1} & : \quad t = 0 \end{cases} \quad (19)$$

The next architecture is established by using finite unfolding in time for the retro-causal path during training and application as shown in fig. 7.

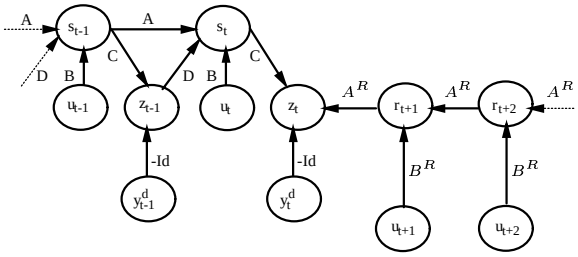

Figure 7: P-CRCECNN_finUnfold

This architecture utilizes eq.:

$$s_t = \tanh(As_{t-1} + Bu_t + D\tanh(Cs_{t-1} - y^d_{t-1}) \quad (20)$$

$$r_t = \tanh(A^R r_{t+1} + B^R u_t) \quad (21)$$

and eq. (19) for computing y_t

4. Application and Results

In this section the application of asymmetric P-CRCRECNNs within the segmental duration control unit of our acoustic prosody module is presented. These data-driven methods are applied to recordings of three hours of a german news speaker reading news from *Frankfurter Allgemeine Zeitung*. The patterns for training (80%) and testing (20%) are separated. A validation set of (20%) is selected randomly from the training set. This database is the same as used within the f0-generation task.

The f0-generation task utilized patterns organized on syllable level [1].

But within this task patterns are organized on phoneme level.

Following information is presented to the NN in a context of seven phonemes to the left and right.

- phonetic information

  with one-out-of-n coding the phoneme index is presented here. A phoneme-set of 45 phonemes is used. Additionally the four phoneme classes (vowel, fricative, nasal liquid, and plosive) are presented here.

- positional information

  Discrete information denotes whether the according syllable is an initial, medial or final one within the phrase and the word. A relative syllable position within a sentence and phrase is also given here.

- stress information

  Flags denoting the stress type of the according syllable are coded here. Word level stress is presented by four flags. Sentence level stress consists of two stress marks.

- linguistic categories

  the used category set consists of 14 categories, which are one-out-of-n coded.

These inputs are presented at each time step of the unfolding clusters denoted by u_{t+i}.

The according output vectors are modeled as observations and are presented at each time step in the clusters denoted by y^d_{t+i}.

Target values for the NN are normalized to ensure an optimized signal-flow during training of the NN due to $tanh$-activation-function. A first normalizing of segmental duration is obtained by the mean and standard deviation value from the triphon clusters (see section 1). A second normalizing was necessary to ensure an optimized signal flow during training of the NN. The mean and standard deviation were derived from the first normalized segmental durations.

For evaluation the trained NN were used to predict segmental durations of sentences which were in the test set. Three audio-files are generated with those predictions which are then used for evaluation.

In fig. 8 the relative squared errors for the first 50 segmental durations of an audio-file are plottet. As can be seen the *stat*-approach (*ai804-stat.roh*) generates the larger error signal. The *P-CRCECNN_removed* signal (*ai804-removed.roh*) is the lowest except at triphone number 46. The *P-CRCECNN_finUnfold* errors are denoted by (*ai804-finUnfold.roh*).

It is possible to see the different approaches delivering stronger error signals at the same triphones. We observed most of these triphones to have a higher standard deviation (about 30% or higher than the mean standard deviation). Note that the mean and standard deviation values generated within the *stat*-approach were utilized for a further normalization of the segmental durations, which were used as observations y^d_t for the NN.

In table 1 the mean square errors (MSE) are given for the two asymmetric NN realisations and the *stat* method. Each of the NN is better than the *stat*.

NN	MSE in %
stat	5.67
P-CRCECNN_removed	3.49
P-CRCECNN_finUnfold	4.35

Table 1: MSE

Table 2 depicts the correlation of the predictions and the original segmental durations. Again the statistic approach is outperformed.

NN	correlation
stat	0.7885
P-CRCECNN_removed	0.8723
P-CRCECNN_finUnfold	0.8232

Table 2: correlation factors

In a further test it was observed that 85.6% of phrase-breaks were realized with a clear final lengthening. The statistic approach did not realize phrase lengthening.

As perception is a highly complex process not necessarily modeled appropriately by isolated physically distances, informal listening test were performed.

Files generated by resynthesis utilizing the three different methods were presented to seven non-expert listeners. They had to judge which of the presented files were most pleasant and least pleasant to them. They also had to give a ranking to the other two files. This ranking was then scaled on a value set from 4 to 1, with 4 denoting the most pleasant file.

The mean values of the ranking are shown in table 3.

method	marks
stat	1.3
P-CRCECNN_removed	3.5
P-CRCECNN_finUnfold	2.5

Table 3: listening tests

The asymmetric P-CRCECNN with connections removed after training was evaluated to be most pleasant.

This architecture uses the error correction principle within the retro-causal path for modeling local prosodic structures. This

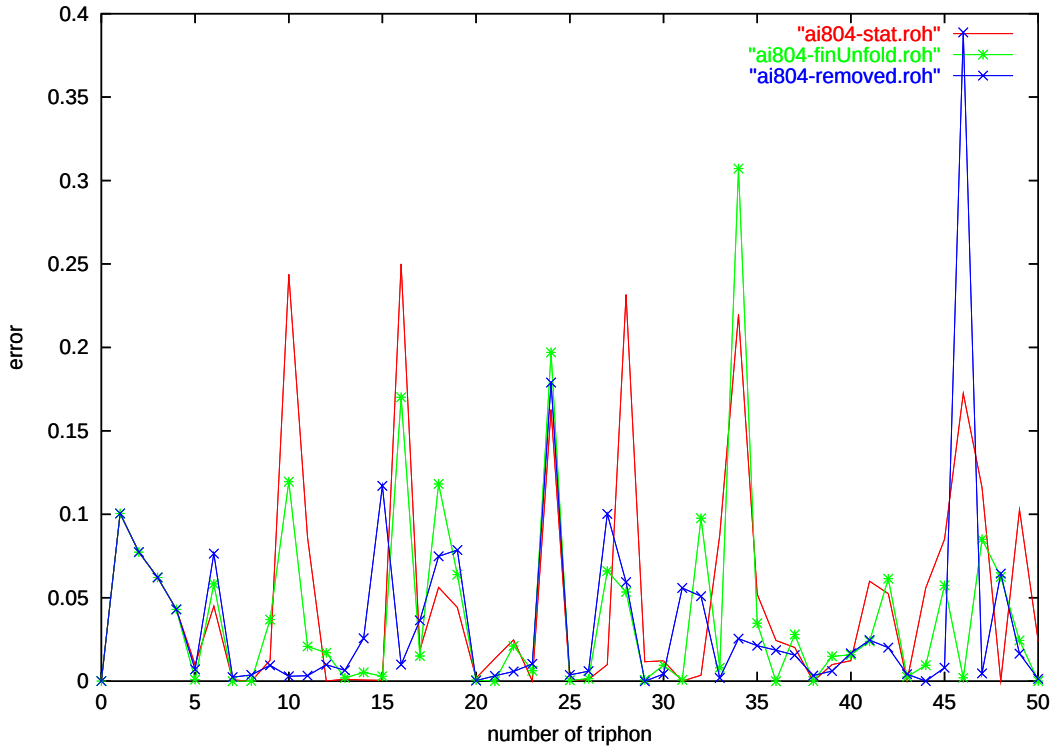

Figure 8: realitive squared errors

seems to help the long term forecast path improving its generalization ability, as this NN performs better than the P-CRCECNN_finUnfold, which does not utilize error correction. The long term forceast path within P-CRCECNN_finUnfold has also to capture short time events.

5. Conclusions

In this paper specialized error correction NN architectures are presented. First a partial CRCECNN is proposed to overcome closed loop computation problems. Afterwards a structural change of the information flow from training to application is integrated into modeling of the NN. Two asymmetric NN architectures are presented to overcome this structural switching.

They are applied within the duration control task of our TTS system *Papageno*. The best performing NN is selected by numeric and listening experiments.

Informal listening tests show that an asymmetric architecture improves performance. A sophisticated architecture with connections trained but removed afterwards during application delivers most pleasant prosodic parameters.

A prior utilized statistic approach within the duration control unit of our acoustic prosody module was outperformed by the presented NN approaches. The prior module is replaced by the NN unit.

6. References

[1] H.-G. Zimmermann, Achim F. Müller, Çağlayan Erdem, and Rüdiger Hoffmann : prosody generation by causal retro-causal error correction neural networks *, MSC 2000, Japan, ATR*

[2] Çağlayan Erdem, Martin Holzapfel, Rüdiger Hoffmann: Natural f0-contours with a new Neural-Network-hybrid-approach In *ICSLP 2000, Beijing*

[3] Ralf Haury, Martin Holzapfel: Optimisation of a Neural Network for Pitch Contour Generation, *ICASSP 1998, Seattle*

[4] M. Riley: Tree-based modelling of segmental durations. *in G. Bailly and C. Benoit, editors., Talking Machines: Theories, Models and Designs, pages 265-273, Elsevier, North-Holland, 1992*

[5] W.N. Campbell: Syllable-based segmental duration *in G. Bailly and C. Benoit, editors., Talking Machines: Theories, Models and Designs, pages 211-224, Elsevier, North-Holland, 1992*

[6] R.Cordoba, J. A. Vallejo, J.M. Montero, J. Gutierrez-Arriola, M.A. Lopez, J.M.Pardo: Automatic modelling of duration in a spanish text-to-speech system using neural networks *,Eurospeech '99*

[7] Marcel Riedi: A neural-network-based model of segmental duration for speech synthesis *,Eurospeech '95*

[8] Martin Holzapfel: HMM based database segmentation and unit selection for concatenative Speech Synthesis *, Proc.joint ASA/DAGA spring meeting hosted by DAGA 99, Berlin*

[9] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E.

Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstucture of Cognition*, volume I, chapter Foundations, pages 318–362. MIT Press/Bradford Books, Cambridge, MA, 1986.

[10] H.-G. Zimmermann, R. Neuneier, R. Grothmann: Neural Network Architectures for the Modeling og Dynamical Systems *In: A Field Guide to Dynamic Reccurrent Network, Eds.: John F. Kolen, Stefan C. Kremer, Publisher IEEEPress 2000, ISBN 0-780-35369-2*

[11] Ralph Neuneier and Hans Georg Zimmermann. How to train neural networks. In G. B. Orr and K.-R. Müller, editors, *Neural Networks: Tricks of the Trade*, pages 373–423. Springer Verlag, Berlin, 1998.

[12] H. G. Zimmermann, R. Neuneier, R. Grothmann: Modeling of Dynamical Systems by Error Correction Neural Networks, *in: A. Soofi and L. Cao [eds.], Modeling and Forecasting Financial Data, Techniques of Nonlinear Dynamics, Kluwer Academic Publishers, 2000 forthcoming.*