A COMPARISON BETWEEN RECURRENT NEURAL ARCHITECTURES FOR REAL-TIME NONLINEAR PREDICTION OF SPEECH SIGNALS

Juan Antonio Pérez-Ortiz, Jorge Calera-Rubio and Mikel L. Forcada
Departament de Llenguatges i Sistemes Informàtics
Universitat d'Alacant
E-03071 Alacant, Spain
Fax: +34 965 90 9326
E-mail: `japerez,calera,mlf@dlsi.ua.es`

Abstract. This paper presents a comparative study on the performance of recurrent neural networks trained in real-time to predict the next sample in a speech signal. The comparison is basically done versus linear predictors, and a pipelined recurrent neural network which has been proposed for this task. Results confirm those of previous works where limitations to deal with numeric time series were detected for recurrent neural architectures, specially when using the real-time recurrent learning algorithm. The decoupled extended Kalman filter training algorithm, on the other hand, overcomes partially some of these limitations.

INTRODUCTION

Real-time speech prediction is an important module in current digital communication systems such as mobile telephone systems. By assuming that the value of the signal at time t may be predicted from the value of the signal at previous times, a lowering of the bit rate may be achieved (depending on predictor's accuracy) by efficiently coding the difference between the actual signal at time t and the predicted signal. No preprocessing is done over speech in this paper: the prediction is performed directly over the sampled waveform.

Although the mechanisms involved in speech generation are inherently *nonlinear*, most current standards [3] consider adaptive *linear* architectures for predictor implementation, because of their acceptable balance between complexity and performance. On the other hand, nonlinear models should in principle take into account these nonlinearities and outperform traditional approaches.

Recurrent neural networks (RNN) [15] seem a promising nonlinear adaptive alternative for speech prediction: recurrent connections allow RNNs to have, in principle, a potentially unlimited memory about the past,[1] whereas adaptive learning makes them appropiate for nonstationary signals like speech.

Haykin and Li [9] designed a pipelined recurrent neural network (PRNN) for speech prediction and considered a modified version of the real-time recurrent learning (RTRL) [16] for training it. The PRNN was used in a cascade of nonlinear and linear predictors. Baltersee and Chambers [1] showed that the PRNN does not perform satisfactorily when RTRL is used and proposed the use of the decoupled extended Kalman filter (DEKF) [13] training algorithm instead, obtaining for the cascaded form of predictors advantages of approximately 2dB over linear filters alone. It has to be pointed out, however, that the results presented in both papers belong to a best-case situation: only one experiment with *ad hoc* parameter values was shown.

The PRNN is composed of several recurrent error propagation networks (REPN, like those used by Robinson and Fallside [14]) sharing weights and connected in such a way that the output of a network feeds the input of the next one. The PRNN may be considered partly as an architecture designed on purpose (in fact, we do not have evidence of other works using it). There is, however, a lack of studies on the performance of classical *general-purpose* RNNs when applied to speech prediction. This paper compares Baltersee and Chambers' results with new ones obtained for some classical RNNs, using both RTRL and DEKF.

METHOD

Following Haykin and Li [9], and Baltersee and Chambers [1], our model of predictor is composed of a nonlinear predictor (a RNN), which is supposed to linearize the input signal, followed by a linear predictor (a filter), which is supposed to take advantage of this linearization. Each module is separately trained. It is expected that "this combination of a nonlinear filter with a linear filter should be able to extract both the nonlinear and the linear information contained in the input signal to produce the prediction" [9].

Figure 1 shows a diagram for the complete cascaded architecture. The first module is trained to predict the sample $u[t]$ from the p previous samples[2] and the information stored in the network's state. The predicted signal $\hat{u}[t]$ is introduced into the linear module, which is trained to predict the sample at time $t+1$. The later is considered as the overall output of the system. As shown in the diagram (following again the aforementioned papers), the nonlinear module has input order p, and q is the corresponding order of the

[1]Actually, the memory of standard RNNs is very limited and reduces to a few samples due to the problem of vanishing gradients [2].

[2]The explicit introduction of recent samples into the network gives it an additional advantage over single-input networks which have access to recent history of the signal only through their state. The experiments will determine the importance of such an addition.

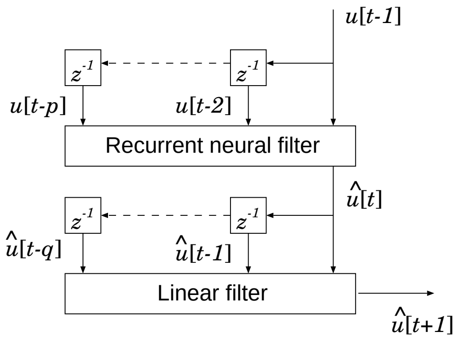

Figure 1: Architecture of the predictor combining nonlinear and linear filters.

linear predictor. In addition to the results for this hybrid model, we also show the results for the nonlinear and linear predictors alone.

Nonlinear Predictors

The following discrete-time RNNs acting as nonlinear predictors are compared: *pipelined recurrent neural network* (PRNN) [9], Elman's *simple recurrent net* (SRN) [5], and Robinson and Fallside's *recurrent error propagation network* (REPN) [14]. We also experimented with other recurrent architectures, such as the fully recurrent network [16] or the NARX network [10], obtaining results similar to those of the SRN and REPN, which, consequently, will not be shown. The number of state (recurrent) neurons of each architecture is denoted by N.

The two gradient-based real-time (that is, *online*) training algorithms used for these architectures are the *real-time recurrent learning* (RTRL) [16] and the *decoupled extended Kalman filter* (DEKF) [13].

Linear Predictors

The linear predictor is a finite-impulse response (FIR) filter whose weights are adapted either by the least-mean-square (LMS) [11, 12] or by the recursive least-squares (RLS) [11, 12] algorithms.

As a way of determining baseline performance, we evaluate as well the prediction quality of a simplest parameter-free predictor computing $\hat{u}[t+1] = u[t]$.

RESULTS

We study the quality of the predictors via the same three signals (length 10000) used by Baltersee and Chambers [1].[3] Performance is measured by

[3]The signals `s1`, `s2` and `s3` are available at Baltersee's homepage at `http://www.ert.rwth-aachen.de/Personen/baltersee.html`.

Table 1: Prediction gains for a 12-tap FIR filter.

Training	Signal 1	Signal 2	Signal 3
LMS	8.99	7.98	5.82
RLS	13.32	11.60	9.66

Table 2: Prediction gains with a PRNN. Values taken from [1].

Training	Signal 1	Signal 2	Signal 3
PRNN (RTRL) + LMS	10.25	9.49	7.30
PRNN (RTRL) + RLS	13.01	11.80	9.24
PRNN (DEKF) + RLS	14.73	13.59	10.90

means of the *prediction gain* (PG), which is defined as

$$G = 10 \log_{10} \left(\frac{S_u^2}{S_e^2} \right) \tag{1}$$

where S_u^2 is the estimated variance of the speech signal $u[t]$ and S_e^2 is the estimated variance of the error signal $e[t] = u[t] - \hat{u}[t]$.

The amplitudes of the three signals lie in the range $[0, 1)$, therefore we use the logistic sigmoid function $f(x) = (1 + e^{-x})^{-1}$ for the activations of the neurons in the output layers of the RNNs (and for every activation function in general).

Following the aforementioned works, we perform an initial epochwise training with 300 samples of the input signal for the neural architectures.[4] The number of training epochs is set to 200 for RTRL and to 5 for DEKF. These values gave good results in preliminary experiments: much higher values for DEKF or much lower ones for RTRL reduce the PG in a few dB.

Unless stated, all the PGs presented are the average for 7 different weight initializations; the variance of these gains is in all cases below 0.3 and it is not shown anywhere. Initial weights are taken randomly from a uniform distribution in [-0.2,0.2].

The PGs obtained with a linear filter alone with $q = 12$ taps, using LMS and RLS algorithms, are shown in Table 1. In this case, the forgetting factor for RLS is 0.998, the diagonal elements of the RLS inverse correlation matrix are initialized to 100, and LMS uses an adaptation constant of 0.2.

The results with the PRNN are taken from the experiments by Baltersee and Chambers [1] and are shown in Table 2. We refer interested readers to their paper for details about values of the training parameters. It has to be stressed that there is no indication in that paper about average results, since only the results for one experiment with *ad hoc* parameters (chosen differently for each signal) are shown. Anyway, even the best-case results obtained here for other recurrent architectures are worse than those obtained with the PRNN.

[4]This attenuates partially the real-time nature of the prediction, but may be acceptable if the complexity is low. Experiments showed that this kind of initialization has a strong influence on the prediction gain obtained.

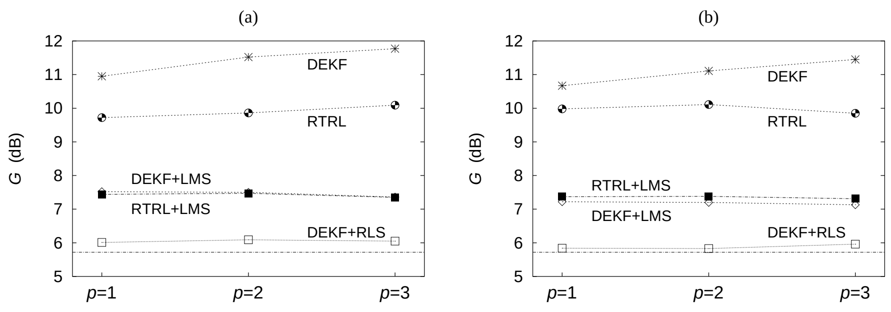

Figure 2: Prediction gains for signal 1: (a) SRN, (b) REPN. Constant line represents baseline reference.

Tables and graphs indicate the real-time training algorithms used for the RNNs and for the linear filter (if any) separated by a plus sign. The order of the linear filters is in all cases $q = 12$.

The results when using the predictors based on SRN and REPN with $N = 5$ are illustrated in Figs. 2 to 4 for different values of the input order p. The parameters in these cases are 0.3 for the RTRL learning rate (no additional momentum term was used), 0.2 for the LMS adaptation constant, 1 for the forgetting factors in RLS and DEKF (no forgetting: values under 1 make them suffer from instability), and 1000 for the initial diagonal elements of the correlation matrices of RLS and DEKF. The rest of the parameters of DEKF were set as proposed in [8, p. 771].

The value $N = 5$ and those of the input order $p = 1, 2, 3$ are chosen so that the number of learnable parameters is comparable to that used by Baltersee and Chambers [1], who consider PRNNs with around 35 adjustable weights.[5] Anyway, SRNs and REPNs with different number of state neurons give results (not shown) which do not vary significantly from those presented for $N = 5$; for example, with $N = 1$, the results using RTRL are approximately the same, whereas those using DEKF are 1dB below; with $N = 10$, RTRL gives again similar PGs, whereas DEKF improves them very little (between 0 and 0.5dB, depending on the particular signal and architecture).

Finally, the PGs for a simple filter taking $\hat{u}[t + 1] = u[t]$ are shown as constant lines in Figs. 2-4. This stands for the simplest way of predicting next sample and is considered here as a baseline reference.

DISCUSSION

Among the three recurrent architectures studied, only the DEKF-trained PRNN followed by a RLS-trained linear filter clearly surpasses (between 1dB and 2dB higher) the PG of a 12-tap linear filter trained with RLS. The rest of

[5]The number of weights, including biases, of a single-output p-th order SRN is $(p+N+2)N$; in the case of a single-output p-th order REPN, it is $(p + N + 2)N + p + 1$.

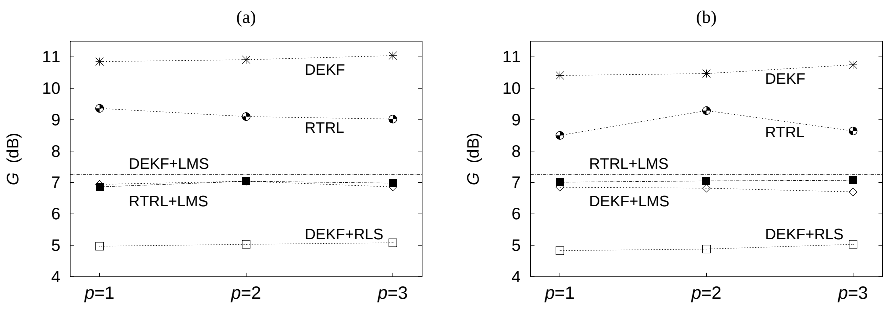

Figure 3: Prediction gains for signal 2: (a) SRN, (b) REPN. Constant line represents baseline reference.

neural configurations (cascaded or not) do worse than a simple RLS-trained FIR filter with a smaller number of parameters.

When using SRN or REPN alone, DEKF yields much better results than standard RTRL algorithm: DEKF attains PGs between 1dB and 3dB higher. The results with both architectures and DEKF consistently confirm those of previous works [4] where the improvement of nonlinear predictors over LMS-trained linear predictors are reported to be around 3dB. However, both training algorithms do not reach the PGs of a RLS-trained FIR filter.

Interestingly, cascading nonlinear predictors based on SRN or REPN and linear predictors produce worse results than using the nonlinear predictors alone. Results with these cascaded forms are very negative as compared with baseline (a predictor reproducing current sample); in fact, for signals 2 and 3 they are even worse. Hence, the following situation arises: it can be considered that we have two types of nonlinear predictors, P (a PRNN) and S (a SRN or a REPN, which present similar behaviour), and that, optionally, we feed a linear predictor L with their output. Let G_P, G_S denote the PGs of the nonlinear predictors alone, G_L the PG of the linear predictor, and G_{PL}, G_{SL} the PGs of the hybrid cascaded model combining one of the nonlinear predictors and the linear one. From the previous results we can write:[6]

$$G_{PL} > G_S \tag{2}$$
$$G_{PL} > G_P \tag{3}$$
$$G_{SL} < G_S \tag{4}$$
$$G_{PL} > G_L \tag{5}$$
$$G_{SL} < G_L \tag{6}$$

From (5) and (6), we conclude that P cancels accurately nonlinearities in the signal and exploits its linear relationships, whereas, conversely, S seems to amplify nonlinearities, lowering the performance of the linear filter. This seems an important aspect worthing a deeper study.

[6]Equation (3) should be true, since Baltersee and Chambers do not show results for a PRNN alone.

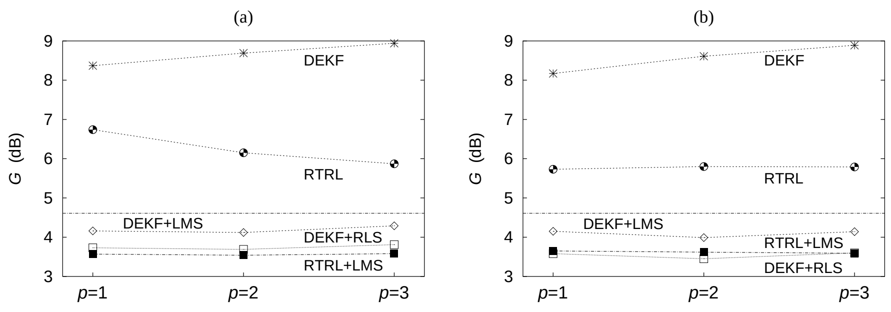

Figure 4: Prediction gains for signal 3: (a) SRN, (b) REPN. Constant line represents baseline reference.

From (3) and (4), we infer that no cascaded configuration is appropiate for SRN or REPN, whereas it is highly recommended for PRNN. Equations (2) and (4) state the superiority of PRNN in cascaded form over the other recurrent models.

When comparing REPN and SRN, the later gives slightly higher PGs. A possible explanation is that a correct use of state information is necessary when using SRN architecture, whereas REPN may be ignoring this information and concentrating exclusively on direct connections from input to output layer.[7] The positive dependence on the input order p is clear when using DEKF but is less obvious in the case of RTRL (in fact, in some cases, increasing p decreases the corresponding prediction gain).

Finally, we also embedded SRN and REPN in a *real* speech coding system following the G721 adaptive differential pulse code modulation (ADPCM) standard [3]. We replaced the infinite-impulse response (IIR) predictor specified by G721 (two poles and six zeros) with SRNs and REPNs, and kept the standard's adaptive quantizer. The new results confirm the previous ones: in this case, only DEKF attains PGs similar to those of the original IIR predictor and RTRL gives much lower gains.

CONCLUDING REMARKS

When the output of a DEKF-trained PRNN is processed by a RLS-trained linear filter, the resulting PGs surpass those obtained with a RLS-trained linear filter alone. This paper studies whether this behaviour can be extended to the case of other classical RNNs. Results are shown for SRN and REPN architectures, but we also experimented with other recurrent networks. Although these RNNs give easily higher prediction gains than those of a LMS-trained linear filter, the best results, attainable via the DEKF training algorithm, are similar (but lower) to those of a RLS-trained linear filter.

[7]The output of a REPN is computed from the network's state and the current input, whereas a SRN computes it solely from the network's state.

This paper makes evident a notorious advantage of DEKF training algorithm over RTRL. Besides that, the signal predicted by SRN and REPN presents a stronger nonlinear character than the actual signal and makes a cascaded configuration with a subsequent linear predictor unfeasible. The performance of PRNN, however, is improved by the linear predictor.

Some works have detected serious limitations [6, 7] of RNNs when applied to nonlinear numeric prediction tasks. The findings presented in this paper suggest similar conclusions.

ACKNOWLEDGEMENTS

This work has been supported by the Generalitat Valenciana through grant FPI-99-14-268, and by the Spanish Comisión Interministerial de Ciencia y Tecnología through grants TIC97-0941 and TIC2000-1599-C02-02.

REFERENCES

[1] J. Baltersee and J. A. Chambers, “Non-linear adaptive prediction of speech signals using a pipelined recurrent network,” **IEEE Transactions on Signal Processing**, vol. 46, no. 8, 1998.

[2] Y. Bengio, P. Simard and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” **IEEE Transactions on Neural Networks**, vol. 5, no. 2, pp. 157–166, 1994.

[3] N. Benvenuto, G. Bertocci and W. R. Daumer, “The 32-kb/s ADPCM coding standard,” **AT&T Technical Journal**, vol. 2, pp. 270–280, 1987.

[4] M. Birgmeier, “Nonlinear prediction of speech signals using radial basis function networks,” in **Proceedings of the European Signal Processing Conference, EUSIPCO 96**, Trieste, Italy, 1996.

[5] J. L. Elman, “Finding structure in time,” **Cognitive Science**, vol. 14, pp. 179–211, 1990.

[6] F. A. Gers, D. Eck and J. Schmidhuber, “Applying LSTM to time series predictable through time-window approaches,” in **Proc. ICANN 2001, Int. Conf. on Artificial Neural Networks**, Vienna, Austria, 2001.

[7] M. Hallas and G. Dorffner, “A comparative study on feedforward and recurrent neural networks in time series prediction using gradient descent learning,” in Trappl, R. (ed.), **Cybernetics and Systems 98, Proceedings of 14th European Meeting on Cybernetics and Systems Research**, Vienna, 1998, pp. 644–647.

[8] S. Haykin, **Neural networks: a comprehensive foundation**, New Jersey: Prentice-Hall, 2nd edn., 1999.

[9] S. Haykin and L. Li, “Non-linear adaptive prediction of non-stationary signals,” **IEEE Transactions on Signal Processing**, vol. 43, no. 2, 1995.

[10] K. S. Narendra and K. Parthasarathy, “Identification and control of dynamical systems using neural networks,” **IEEE Transactions on Neural Networks**, vol. 1, pp. 4–27, 1990.

[11] A. V. Oppenheim and R. W. Schafer, **Discrete-time signal processing**, Prentice-Hall, 1989.

[12] J. Proakis and D. Manolakis, **Digital signal processing**, Prentice Hall, 3rd edn., 1996.

[13] G. V. Puskorius and L. A. Feldkamp, “Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks,” **IEEE Transactions on Neural Networks**, vol. 5, no. 2, pp. 279–297, 1994.

[14] A. J. Robinson and F. Fallside, “A recurrent error propagation speech recognition system,” **Computer Speech and Language**, vol. 5, pp. 259–274, 1991.

[15] A. C. Tsoi and A. Back, “Discrete time recurrent neural network architectures: a unifying review,” **Neurocomputing**, vol. 15, pp. 183–223, 1997.

[16] R. J. Williams and D. Zipser, “A learning algorithm for continually training recurrent neural networks,” **Neural Computation**, vol. 1, pp. 270–280, 1989.