Recurrent Neural Networks in Audio Processing

Georg Holzmann

May 2007

In this article echo state networks, a special form of recurrent neural networks, are discussed in the area of nonlinear audio signal processing. Echo state networks are a novel approach in recurrent neural networks with a very easy (linear) training algorithm. Signal processing examples in nonlinear system identification (valve distortion, clipping), inverse modeling (quality enhancement) and audio prediction are briefly presented and discussed.

Contents

1 Introduction **2**
- 1.1 Neural Networks . . . 2
- 1.2 Echo State Networks . . . 3
- 1.3 RNNs in Signal Processing . . . 3

2 Echo State Network Theory **4**
- 2.1 Notation and Activation Update . . . 5
- 2.2 Network Creation . . . 5
- 2.3 Offline Training Algorithm . . . 6
- 2.4 Online Training Algorithm . . . 6
- 2.5 Additional Nonlinear Transformations . . . 7
- 2.6 Short Term Memory Effects . . . 7

3 Audio Signal Processing Examples **8**
- 3.1 Nonlinear System Identification . . . 8
  - 3.1.1 Valve Distortion . . . 8
  - 3.1.2 Hard Clipping . . . 9
- 3.2 Inverse Modeling . . . 10
- 3.3 Audio Prediction . . . 12

4 Conclusion **14**

1 Introduction

Echo state networks, as described by Jaeger in [5], and liquid state machines, as described by Maass in [10], are a novel approach to recurrent nerwork training and were successfully used mainly in control theory problems. However, not much is known about using echo state networks in the field of signal processing, which is be the topic of this article.

First I will briefly introduce basic concepts of artificial neural networks, recurrent neural networks, echo state networks and will give some reasons why and for what one could use these networks for audio signal processing (section 1). Then the theory of echo state networks is discussed in more details (section 2) and finally 3 examples in audio signal processing are presented (section 3).

1.1 Neural Networks

An Artificial Neural Network (ANN) is an algorithm, which tries to model a very simplified version of our brain.
It consists of interconnected neurons, or also called perceptrons, where one neuron weights and adds up all its input connections and applies an activation function on this sum (a simple nonlinear function). The weights of the neurons are adjusted during training, so it's possible to learn any nonlinear input-output mapping (see Figure 1). Therefore artificial neural nets can be seen as a generalization of linear adaptive filters in nonlinear domain.

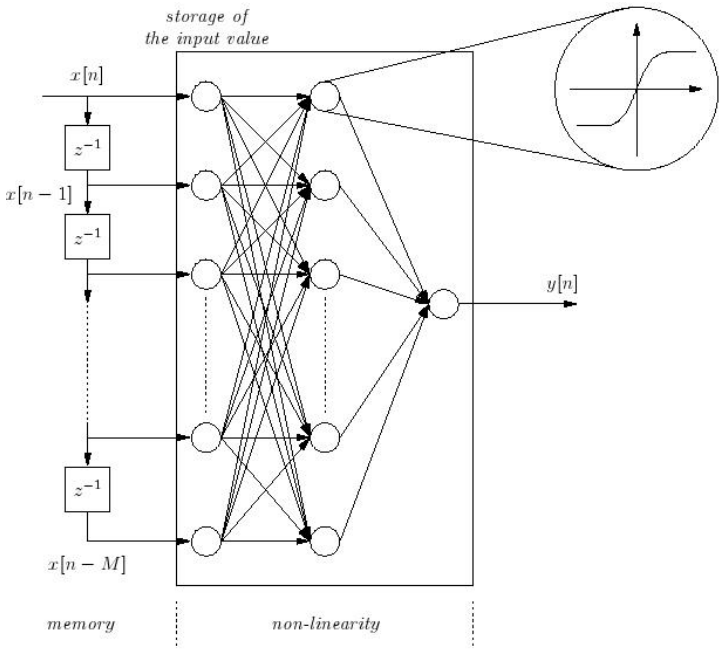

Figure 1: An example of a "Multilayer Perceptron" or "Feedforward Neural Net": All neurons are connected in feedforward mode, which means that there are no recurrent connections back to the previous layer. Zoomed is an activation function of one neuron, a tanh mapping in this case. (figure from [1])

Compared to Feedforward Neural Nets (Figure 1), Recurrent Neural Networks (RNNs) have

at least one cyclic path of synaptic connections. Mathematically Recurrent Neural Networks implement dynamical systems and can, in theory, approximate arbitrary nonlinear dynamical system with arbitrary precision (universal approximation property). RNNs are also able to (re)produce temporal patterns.
However, the problem with RNNs is the training algorithm, which is usually very complicated and can lead to suboptimal solutions.

1.2 Echo State Networks

Echo State Networks (ESNs) are a special kind of Recurrent Neural Networks, with a very easy training algorithm.
Usually a large, sparsely connected RNN is used as a "Dynamic Reservoir", which can be excited by inputs and/or feedback of the outputs. The big advantage is, that the connection weights of the Reservoir are not changed by training and only weights from the Reservoir to the output units are adapted, so training becomes a linear regression task (see Figure 2).

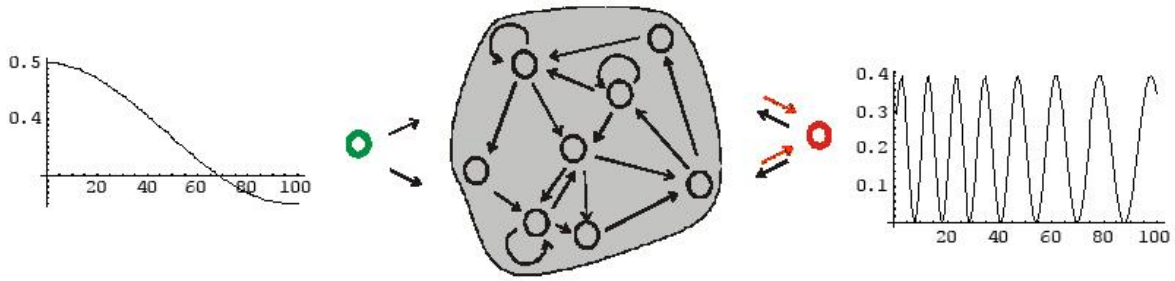

Figure 2: In this example an ESN is trained as a tuneable sinewave generator: Black arrows are the fixed input and feedback connections, gray the recurrent interconnected Dynamic Reservoir and only the red arrows are trainable output connections. (figure from [7])

Difficulties with existing algorithms have so far precluded supervised training techniques for RNNs from widespread use. Unlike with Feedforward Neural Nets, several types of training algorithms are known for RNNs, no clear winner (e.g. Backpropagation Through Time, Real-Time Recurrent Learning, Extended Kalman Filtering Approaches, ... see [7]) and usually these algorithms have a slow convergence and/or lead to suboptimal solutions.
Therefore Echo State Networks are a new, promising approach in supervised training of RNNs (see Figure 3).

1.3 RNNs in Signal Processing

But why should these complicated Recurrent Neural Network be used in Signal Processing ?
Classical methods of statistical signal processing are founded on three basic assumptions: linearity, stationarity and Gaussianity. These assumptions are made for the sake of mathematical tractability, but most real-life physical signals are generated by dynamic processes that are nonlinear, nonstationary and non-Gaussian.
If one wishes to simulate, predict, filter, classify or control nonlinear dynamical systems, one needs an executable system model and often it is infeasible to obtain analytical models, so one has to use blackbox modeling techniques (see [3]).

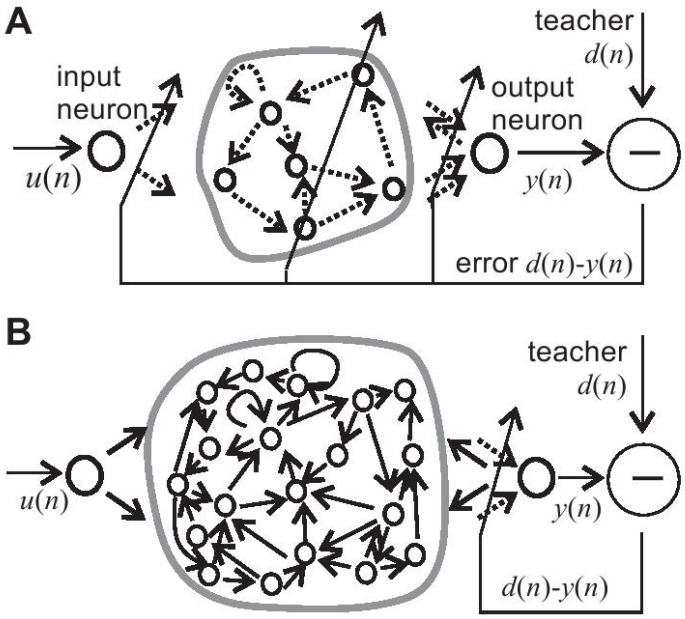

Figure 3: **A:** Schema of previous approaches to RNN learning, all weights are adapted; **B:** ESN approach, where usually much more neurons are used in the Dynamic Reservoir and only the output weights are adapted; (figure from [4])

Classical approaches for nonlinear signal processing consists of designing specific algorithms for specific problems (e.g. median and bilinear filters, Volterra filters, polynomial filters, functional links, ...). Therefore ESNs and in general RNNs extend and generalize the simple adaptive linear filter in nonlinear domain and can be used to model any nonlinear dynamical system.

Some possible applications of ESNs in Audio Signal Processing are:

- nonlinear adaptive filtering techniques
- audio signal prediction and restauration, quality enhancement
- system identification and simulation of nonlinear amplifiers, tubes, nonlinear transducer linearization, modelling of audio effects etc.
- learning based sound synthesis
- audio coding
- ...

2 Echo State Network Theory

For simplifying notation we only address the task of modeling single-input, single-output systems without feedback connections, as they are quite common in signal processing (see Figure 4).

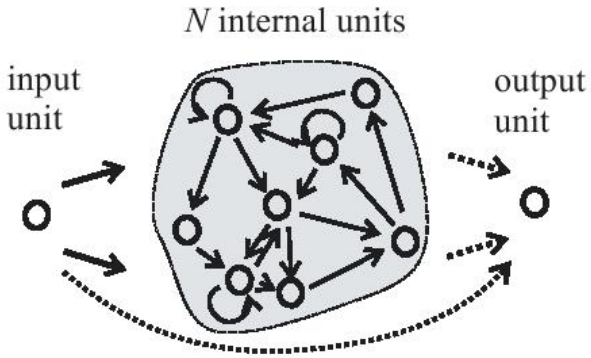

Figure 4: a single-input, single-output ESN; only the output connections are trained; (figure from [8])

2.1 Notation and Activation Update

In our observations we consider an ESN with a Dynamic Reservoir (DR) with N interal network units. At time $n \geq 1$ the input is $u(n)$ and the output $y(n)$.
The activations of the internal units (neurons of the DR) are collected in a $Nx1$ vector $x(n) = (x_1(n), ..., x_N(n))$, internal connection weights in a NxN matrix W, weigths of input connections in a $Nx1$ vector w^{in} and output weigths in a $(N+1)x1$ vector w^{out} (input and network to output connections - see Figure [8]).

The activation of the neurons in the Dynamic Reservoir are updated according to

$$x(n+1) = f(Wx(n) + w^{in}u(n+1) + v(n+1))$$

where $v(n+1)$ is a small noise term and f is the activation function in the DR (e.g. tanh).

With this internal state vector we can calculate the network output with

$$y(n+1) = f^{out}(w^{out}(u(n+1), x(n+1)))$$

again f^{out} is the output unit activation function (also tanh or linear, ...).

2.2 Network Creation

Under certain conditions the network state becomes asymptotically independent of initial conditions and depends only on input history, which is called the "Echo State Property". This means that all desired output signals can be build out of it's own "echos" in the Dynamic Reservoir.

The following mathematical proposition about the echo state property is given in [7]:
Assume an untrained network (w^{in}, W, w^{out}) with transfer functions tanh. Let W have a spectral radius $|\lambda_{max}| > 1$, where λ_{max} is the largest absolute value of an eigenvector of W. Then the network has no echo states with respect to any input/output interval UxD containing the zero input/output (0,0).

However, in practice it was consistently found (see also [7]), that when the previous proposition is NOT satisfied, we do have an echo state network. Therefore one can use the following procedure to initialize an echo state network:

- W is randomly generated from a uniform distribution over $[-1, 1]$
- W is then normalized to the spectral radius $\alpha < 1$ by scaling W with $\alpha / |\lambda_{max}|$
- the spectral radius α is a crucial parameter for the eventual success of an ESN, small α are for fast signals, large α for slow signals and a longer short term memory (see Section 2.6)

2.3 Offline Training Algorithm

In training we compute the output weights, such that the training error e_{train} is minimized in mean square sense:

$$e_{train}(n) = (f^{out})^{-1} y_{teach}(n) - w^{out}(u_{teach}(n), x(n))$$

where the effect of the output nonlinearity is undone by $(f^{out})^{-1}$. Therefore the computation of w^{out} is a linear regession task which can be calculated offline or online.

The offline version leads to a calculation of a simple pseudo-inverse and can be done in the following procedure:

- init W with spectral radius $\alpha < 1$ and run the ESN with the teaching input signal
- dismiss data from initial transients and collect remaining input and network states $(u_{teach}(n), x_{teach}(n))$ row-wise into a matrix M
- collect the training signals $(f^{out})^{-1} y_{teach}(n)$ into a vector r
- compute the pseudo-inverse M^{-1} and put $w^{out} = (M^{-1} r)^T$
- the ESN is now trained and can be used

2.4 Online Training Algorithm

Standard recursive algorithms for MSE minimization known from adaptive linear signal processing (like LMS and RLS) can be applied to online ESN estimation. According to [5] it turns out that LMS converges too slowly, but the "recursive least square" (RLS) algorithm, which is widely used in adaptive signal processing when fast convergence is of prime importance, works fine.

Given an open-ended, non-stationary training sequence, the training algorithm should determine an augmented vector $w^{out}(n)$ at each timestep. Therefore the RLS algorithm minimizes the following square error:

$$\sum_{k=1}^{n} \lambda^{n-k} ((f^{out})^{-1} y_{teach}(k) - (f^{out})^{-1} y_{[n]}(k))^2$$

where $\lambda < 1$ is the forgetting factor and $y_{[n]}(k)$ is the model output that would be obtained at time k, when a network with the current $w^{out}(n)$ would be used at all times $k = 1, ..., n$.

Two parameters characterise the tracking performance of an RLS algorithm.
The misadjustment M gives the ratio between the excess MSE incurred by the adaptation process and the optimal MSE that would be obtained with offline-training.
The time constant τ determines the exponent of the MSE convergence.

These parameters are related to the forgetting factor λ and the length of the tap-vector N (= length of w^{out}) with the following equations:

$$M = N\frac{1-\lambda}{1+\lambda}, \tau \approx \frac{1}{1-\lambda}$$

2.5 Additional Nonlinear Transformations

The modeling power of an ESN grows with network size. However, for hard nonlinear tasks a cheaper way to increase the power is to use additional nonlinear transformations of the network state $x(n)$ (see [8]).

This can be done with extending the internal state vector $x(n)$ with e.g. additional squared states, which leads to $x_{square}(n) = (x_1(n), ..., x_N(n), x_1^2(n), ..., x_N^2(n))$, a new input weight vector $u_{square}(n) = (u(n), u^2(n))$ and an output weight vector w^{out}_{square} which is two times longer as w^{out}.

The learning procedure stays the same (now with $x_{square}(n)$, $u_{square}(n)$ and w^{out}_{square}) and the output can be calculated with

$$y(n+1) = f^{out}(w^{out}_{square}(u_{square}(n+1), x_{square}(n+1)))$$

2.6 Short Term Memory Effects

Short term memory means, how many of the previous input/output arguments $(u(n-k), y(n-k-1))$ are actually relevant for the current output value.

To get a measure of the short term memory capability of an ESN we consider a simple delay task. There we can calculate the squared correlation coefficient $r^2(u(n-k), y_k(n))$ between the correct delayed signal $u(n-k)$ and the network output $y_k(n)$ trained on the delay k. A values of 1 means perfect correlation and 0 complete loss of correlation.

So we can define the memory capacity MC of a network with

$$MC = \sum_{k=1}^{\infty} r^2(u(n-k), y_k(n))$$

It was proofen in [6] that for a ESN whose Dynamic Reservoir has N nodes, the maximal possible memory capacity is bounded by N:

$$MC <= N$$

When we plot the correlation coefficient against the delay k, we obtain the forgetting curves (see Figure 5), where we can see the memory capacity of various ESN setups.

However, as practical suggestions when one needs ESNs with long short term memory effects, one can resort to a combination of the following approaches:

- use large Dynamic Reservoirs
- use small input weights or linear update for the DR (might conflict with nonlinear modeling tasks)
- use DRs with almost unitary weight matrices (see Section 3.3 for more details)
- use a spectral radius α close to 1 (would not work if one wants to have fast oscillating dynamics)

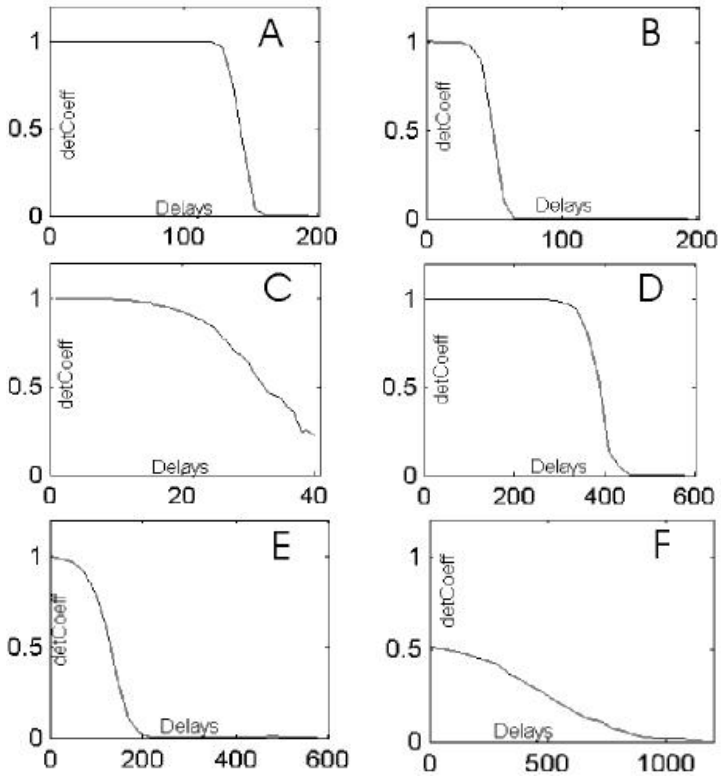

Figure 5: Forgetting curves for a 400 unit ESN: **A:** randomly created linear DR; **B:** randomly created sigmoid DR; **C:** like A, but with noisy state update of the DR; **D:** linear, almost unitary weight matrix; **E:** as D, but with noisy state update; **F:** as D, but with spectral radius $\alpha = 0.999$; (figure from [6])

3 Audio Signal Processing Examples

The following examples were implemented in Octave [1] and are also useable in Matlab.

3.1 Nonlinear System Identification

Like in linear adaptive signal processing we try to identify an input-output system, but nonlinear in the following two examples. In the first example we simulate the behaviour a valve (tube) distortion and in the second we learn a hard nonlinear system (hard clipping).

3.1.1 Valve Distortion

As audio input a flute and a male voice was used and then it was sent into the "Valve Saturation" digital audio plugin (LADSPA audio plugin - see [2]) to get the teacher signal. A single-input single-output ESN was trained with those signals.

The ESN was setup with 50 neurons in the DR and used additional squared state updates $x_{square}(n), u_{square}(n)$ as described in 2.5. The DR had a connectivity of 20%, spectral radius

[1] GNU Octave is an open source high-level language, primarily intended for numerical computations and mostly compatible with Matlab, see http://www.gnu.org/software/octave/

was set to $\alpha = 0.8$, as activation function tanh was used in the DR, and the output neurons had a linear activation function. Input weights w_{in} were sampled from a uniform distribution over [-4, 4], to get the DR activation functions closer to saturation.

In figure 6 and 7 one can see that the system idenfification works quite well and results in a low mean square error (MSE), also the audible results are quite good.

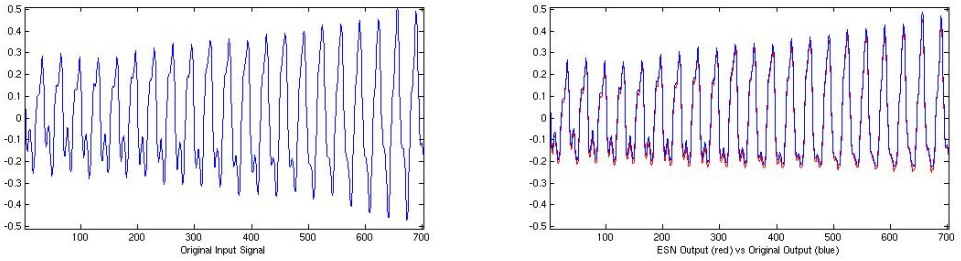

Figure 6: 700 samples simulation of a valve distortion with a flute sound; left the original input signal and right the teacher output signal (blue) and the output of the ESN (red); $MSE_{test} = 3.2602\text{e-}04$;

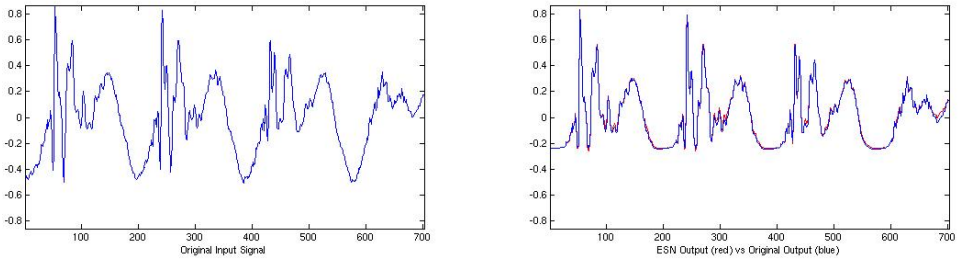

Figure 7: 700 sample simulation of a valve distortion with a male voice sound; left the original input signal and right the teacher output signal (blue) and the output of the ESN (red); $MSE_{test} = 1.6024\text{e-}04$;

Figure 8 shows the spectogram of an input and output sound, where one can see how the valve simulation emphasizes higher partials.

3.1.2 Hard Clipping

Hard clipping means, that all samples bigger then ± 0.2 were clipped to ± 0.2, which results in lots of additional partials.

Again a male voice and a flute sample was used as input signal for the system and the same single-input single-output ESN as in the previous example was trained.

Figure 9 shows the time domain result with the flute sound, which gave good results and a low MSE.

Also with the male voice sample the system indentification worked quite well and a MSE_{test} of 1e-05 to 3e-05 was achieved.

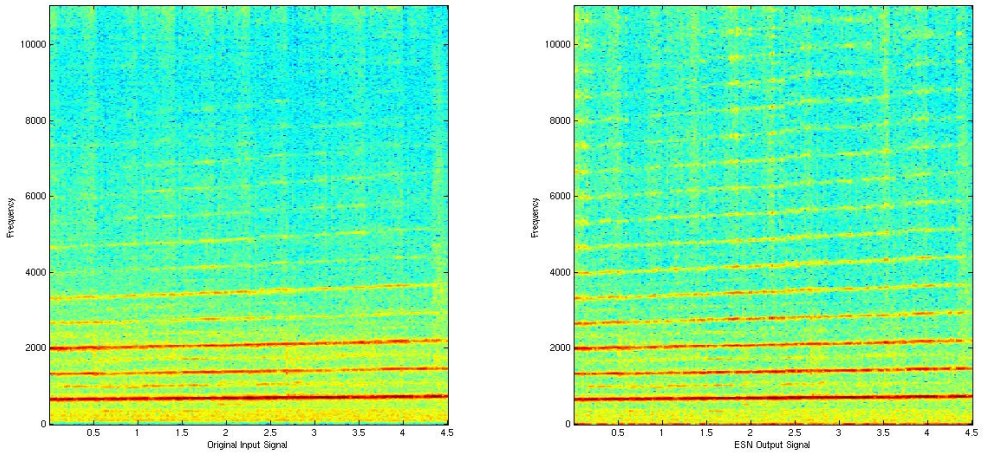

Figure 8: 4.5 sec simulation with flute sound; left the original input signal, right ESN output signal with stronger partials; $MSE_{test} = 6.0871\text{e-}04$;

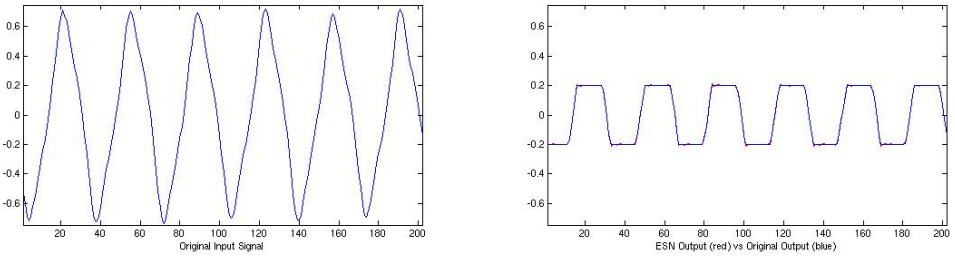

Figure 9: 700 samples hard clipping of a flute sound; left the original input signal and right the teacher output signal (blue) and the output of the ESN (red); $MSE_{test} = 1.1016\text{e-}05$;

3.2 Inverse Modeling

Often it is more interesting to learn the inverse of a system, which might be unstable in linear case.

In this example a string and a male voice sample (samplingrate 8 and 16 kHz) was downsampled and upsampled afterwards, to get really rid of all the high frequencies. The ESN network tried to reconstruct all the high frequencies, with using the resampled version as input and the original version as teacher signal.

In general the same ESN as before was used (size 50, quadratic state updates, spectral radius α=0.8, ...), but best results were made with linear activation functions in the Dynamic Reservoir and tansig activation functions in the output layer.

A very important parameter here was the training size. With a small trainingsize one gets much high frequency components, which might sound very metallic and distorted, on the other side a too large trainingsize results in very poor high frequencies.
Therefore, as a compromise, 2500 samples (and the first 300 samples of those were discarded

in the training algorithm) were used for training in the following examples.

In figure 10 one can see the results for resampling factors 2, 3 and 4.

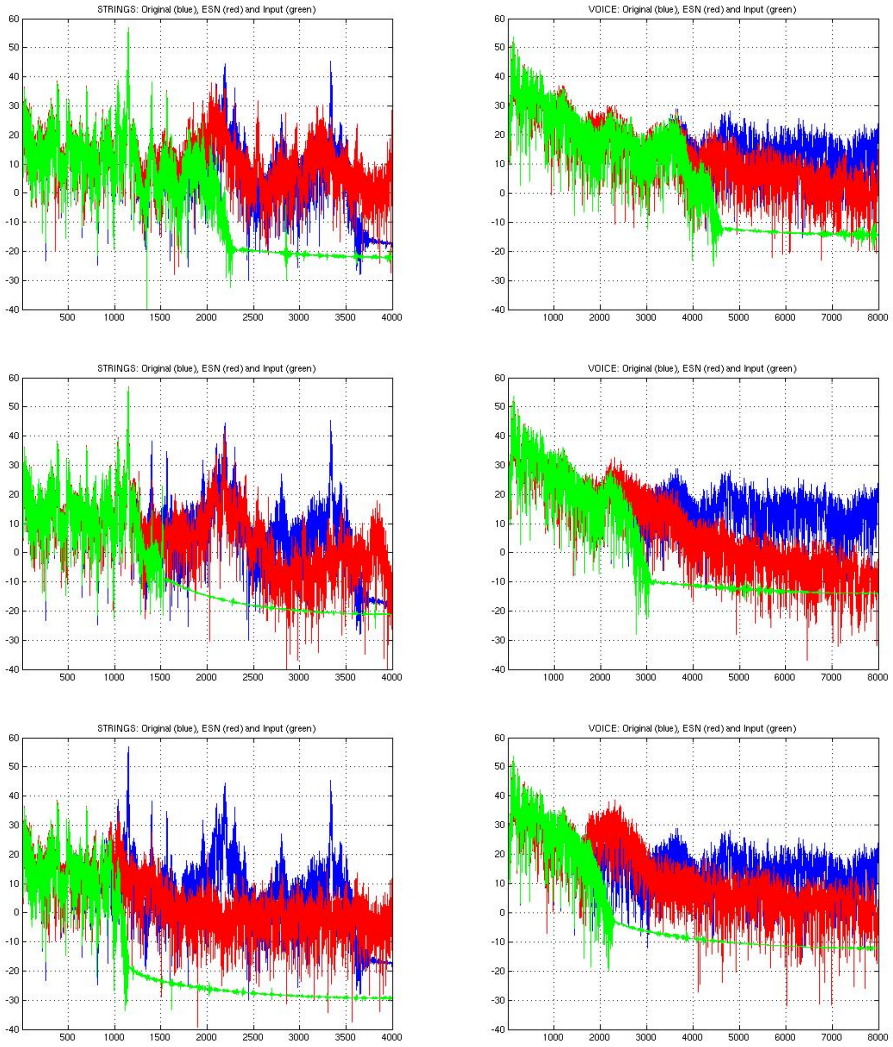

Figure 10: Downsampling and upsampling of string (left) and voice (right) sounds; blue is the spectrum of the original signal, green the down- and upsampled signal, red the output of the ESN, which tries to reconstruct the original signal; first row with a resampling factor of 2, second row with factor 3 and third row with factor 4;

For resampling factor 2 the reconstruction of the string sample works quite good (String MSE_{test}=0.0075), the voice sounds a little bit metallic, although the MSE is quite good (Voice MSE_{test}=0.0042).

The audible results for resapmling factor 3 are quite similar as with factor 2 and resulted in String MSE_{test}=0.0103 and Voice MSE_{test}=0.0048.

With resampling factor 4 it does not work at all for the string sample anymore, the voice is not much different than before (higher samplingrate than the string sample) and a MSE of

String MSE_{test}=0.0345 and Voice MSE_{test}=0.0160 was achieved.

3.3 Audio Prediction

The task of audio prediction is to compute future samples out of previous input samples, which is e.g. necessary in audio restoration, whenever a sequence of consecutive samples is missing or when impulsive noise appears. Usually a preprocessing stage localizes the noise or missing sample position and then a reconstruction stage is needed.

Common methods for missing data reconstruction are based on the assumption that signals can be modeled as a P order autoregressive process, where the model parameters are taps of a linear predictor filter. Then missing samples are predicted in foward and backward mode and crossfaded afterwards (see Figure 11). The main drawback of an autoregressive model based approach is the big model order of two or three times the length of missing data.

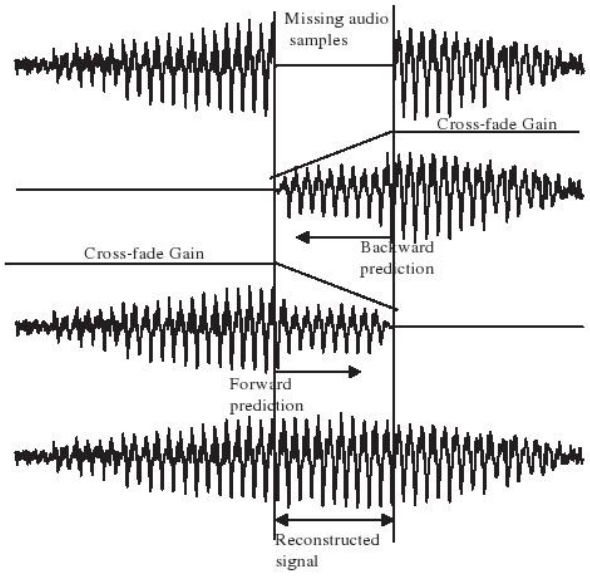

Figure 11: Schema of forward and backward prediction; (figure from [9])

For the signal prediction task an ESN without input and one output unit was used. This output unit features connections that project back the teacher signal into the Dynamic Reservoir, the weights of these backprojections are also not changed during training (see Figure 12).

To predict a signal we drive the ESN with the teacher audio signal (teacher forcing), which means that the teacher signal is feed back into the DR and thereby excites activation dynamics within the DR (the "echos" of the signal). Then we let the ESN predict the future signal from the echos of the teacher signal.

In these examples ESNs with 100 and 400 neurons in the DR (connectivity 20% and 5%) were used. To get a better short-term memory performance the DR was initiated with an "almost" unitary weight matrix W, which was done by replacing the singular value diagonal matrix from octaves/matlabs SVD function by the identity matrix, then the resulting matrix was scaled with the factor $C = 0.9$ (see [6] for more details on this method). A linear activation function was used in the DR (again to get better short-term memory), the output neurons had

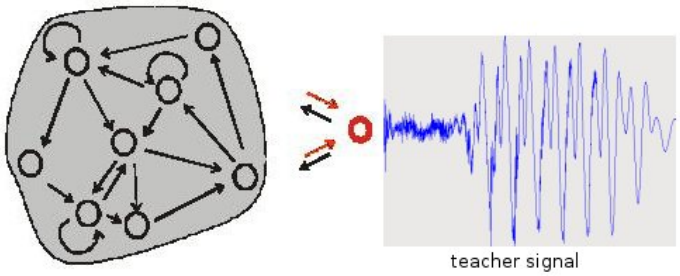

Figure 12: ESN setup for the prediction task: teacher signal is projected back into the DR, then the teacher signal stops and the ESN is producing the prediction out of the echos from the teacher signal;

a tanh activation function and feedback weights w_{fb} were sampled from a uniform distribution over [-0.5, 0.5].

Two different testsignals were used: first testsignal was a steady tone of two violins with some noise and the second testsignal was a band (drums, bass, keyboard, flute), which played music with a beat (to get more transients). The ESN was first trained for forward prediction, then for backward prediction and finally these two generated signals were crossfaded.

It turned out that prediction works quite well up to the short term memory limit (see also 2.6). In Figure 13 one can see a 100 and 400 neuron ESN working on the short term memory limit, which are 200 and 800 samples ($= 2N$) in that case (forward and backward prediction).

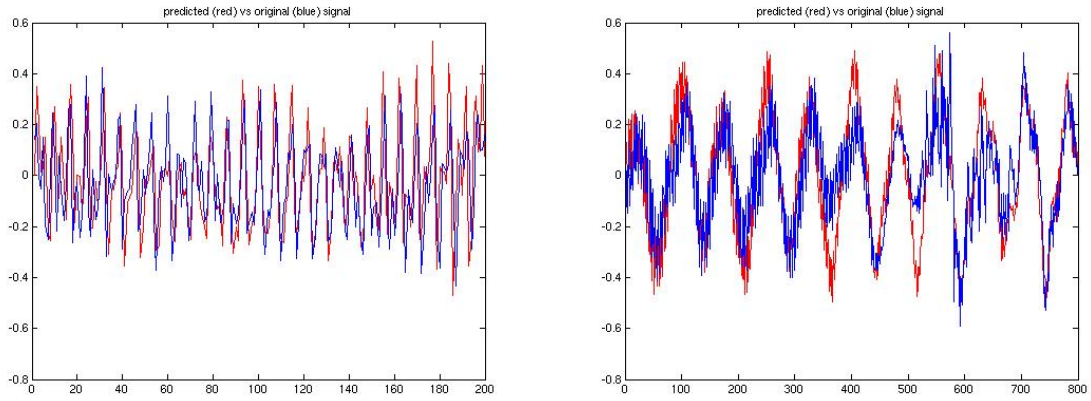

Figure 13: left an ESN with 100 neurons predicting 200 samples of a string sound (MSE_{test}=0.0232); right a 400 neuron ESN predicting 800 samples of a beat sound (MSE_{test}=0.0225);

It is also possible to go far beyond the short term memory limit, but in this region the quality of the sound changes and locks into a steady state which somehow remembers only the periodic information (see Figure 14). An other interesting thing is that the MSE is still the same, although the sound quality is different.

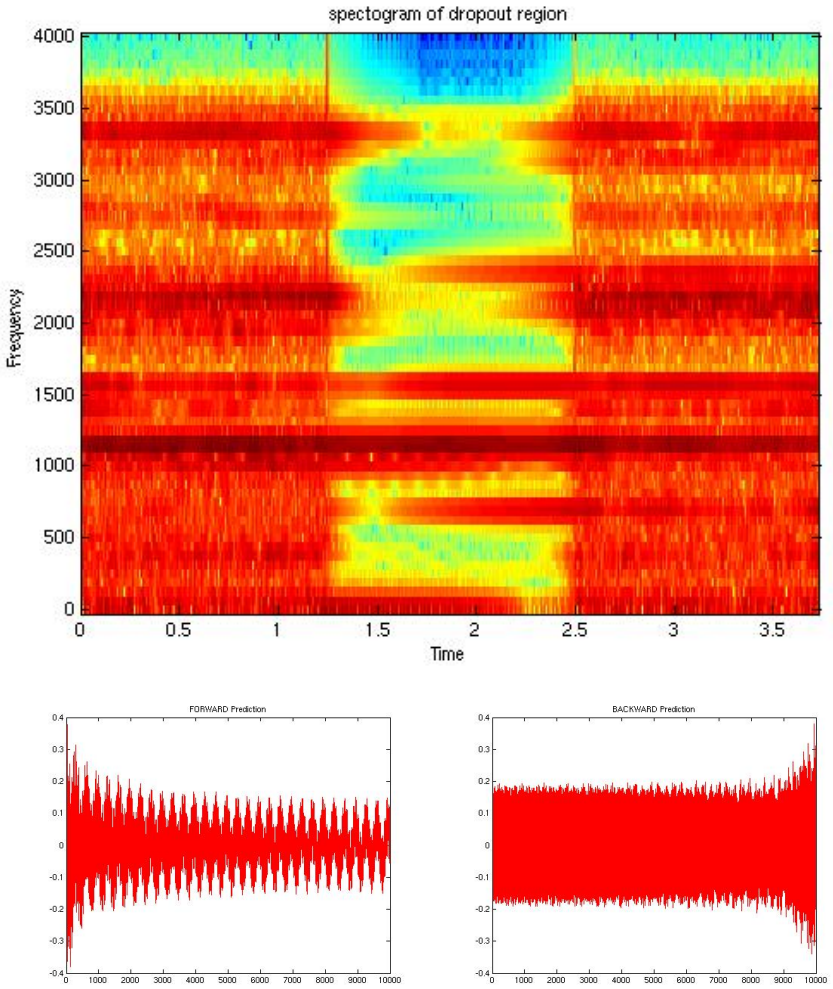

Figure 14: spectogram of predicting 10000 samples of a string sound with a 400 unit ESN (MSE_{test}=0.0266); one can see that the noisy information is gone; at the bottom time domain plots of forward and backward prediction;

4 Conclusion

In the last section I presented some possible applications of Echo State Networks in Audio Signal Processing.
In this connection the following observations were made:

- the nonlinear system identification and audio prediction task worked quite well
- however, a real comparison to other algorithms needs to be done
- in the nonlinear system identification task it was very important to use quadratic state updates

- in all these examples the algorithm was very robust, which means that it somehow worked out of the box without much finetuning of the parameters (exception: inverse modeling, there the success heavily depends on training size)

Some problems, that could be figured out in future developments:

- the MSE is not always a useful measure of the sound quality, our perception works different
- it is easy to generate additional high partials of a sound, but hard to undo this process - maybe other activation functions are needed or a processing in frequency domain
- e.g. it was not possible to learn the inverse system of the tube simulation or hard clipping - the MSE and the plots were quite good, but the sound was distorted
- for long predictions the ESN output changes into a state reproducing only the periodic parts of an input sound - this might be useful or not

References

[1] Christoph Krall Gernot Kubin. Nonlinear signal processing. 2005. Lecture Notes.

[2] Steve Harris. Valve saturation ladspa plugin. http://plugin.org.uk/ladspa-swh/docs/ladspa-swh.html#tth_sEc2.107; accessed 30-05-2007.

[3] Simon Haykin. Neural networks expand sp's horizons. 1996. IEEE Signal Processing Magazine.

[4] Harald Hass Herbert Jaeger. Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless telecommunication. 2004. International University Bremen.

[5] Herbert Jaeger. The "echo state" approach to analysing and training recurrent neural networks. 2001. Frauenhofer Institue for Autonomous Intelligent Systems, GMD Report 148.

[6] Herbert Jaeger. Short term memory in echo state networks. 2002. Frauenhofer Institue for Autonomous Intelligent Systems, GMD Report 152.

[7] Herbert Jaeger. A tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the "echo state network" approach. 2002. Frauenhofer Institue for Autonomous Intelligent Systems.

[8] Herbert Jaeger. Adaptive nonlinear system identification with echo state networks. 2003. Frauenhofer Institue for Autonomous Intelligent Systems.

[9] Aurelio Uncini. Audio signal processing by neural networks. 2003. University of Rome.

[10] Henry Markram Wolfgang Maass, Thomas Natschläger. Real-time computing without stable states: A new framework for neural computation based on perturbations. Institute for Theoretical Computer Science TU Graz, Ecole Polytechnique Federale de Lausanne.