Neural Networks 23 (2010) 244–256

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Echo state networks with filter neurons and a delay&sum readout

Georg Holzmann [b,a,*], Helmut Hauser [a]

[a] Institute for Theoretical Computer Science, Technische Universität Graz, A-8010 Graz, Austria

[b] Neural Information Processing Group, Technische Universität Berlin, D-10587 Berlin, Germany

ARTICLE INFO

Article history:
Received 5 January 2009
Accepted 8 July 2009

Keywords:
Echo state networks
Reservoir computing
Nonlinear dynamical system modeling
Delay learning
Time series prediction

ABSTRACT

Echo state networks (ESNs) are a novel approach to recurrent neural network training with the advantage of a very simple and linear learning algorithm. It has been demonstrated that ESNs outperform other methods on a number of benchmark tasks. Although the approach is appealing, there are still some inherent limitations in the original formulation.

Here we suggest two enhancements of this network model. First, the previously proposed idea of filters in neurons is extended to arbitrary infinite impulse response (IIR) filter neurons. This enables such networks to learn multiple attractors and signals at different timescales, which is especially important for modeling real-world time series. Second, a delay&sum readout is introduced, which adds trainable delays in the synaptic connections of output neurons and therefore vastly improves the memory capacity of echo state networks.

It is shown in commonly used benchmark tasks and real-world examples, that this new structure is able to significantly outperform standard ESNs and other state-of-the-art models for nonlinear dynamical system modeling.

1. Introduction

Recurrent neural networks (RNNs) can in theory approximate arbitrary nonlinear dynamical system with arbitrary precision (*universal approximation property* (Siegelmann & Sontag, 1991)) and are also able to (re)produce temporal patterns. However, learning the weights in RNNs is tedious. A number of specialized learning algorithms exist in literature (Jaeger, 2002b), which are usually converging slowly and often lead to suboptimal solutions.

A new and surprisingly easy to use network structure for RNNs was invented independently by Jaeger (2001), who called these RNNs echo state networks (ESN), and by Maass, Natschlaeger, and Markram (2002), who developed a similar approach for spiking neural networks and called this structure liquid state machine (LSM). Both methods are usually subsummed in literature under the more general term reservoir computing (Verstraeten, Schrauwen, D'Haene, & Stroobandt, 2007). The common idea is that input signals are fed into a fixed nonlinear dynamical system, called dynamic reservoir or liquid, which is composed of randomly connected neurons. Only the output connections, the readout, are then trained by simple linear regression. Fig. 1 shows a schematic overview of the reservoir computing concept.

Most implementations of the liquid state machine use a spiking neuron model called leaky integrate and fire (LIF) neuron (Maass & Bishop, 2001), whereas echo state networks are composed out of analog neurons, for instance linear, sigmoid or leaky-integrator units. The function of the reservoir can be compared to that of the kernel in support vector machines (Cristianini & Shawe-Taylor, 2000): input signals drive the nonlinear reservoir and produce a high-dimensional dynamical "echo response", which is used as a non-orthogonal basis to reconstruct the desired outputs by a linear combination. This strategy has the advantage that the recurrent network is fixed and simple offline or online algorithms for linear regression can compute the output weights. Therefore training is simple, fast and cannot get stuck in a local minima.

Although the reservoir computing approach is very appealing, there are still some inherent limitations in the original formulation and solutions for two main problems are the contribution of this work. First, the problem of learning multiple attractors or signals with different timescales is addressed. Since reservoir neurons are connected to each other, their states are coupled and it is for instance not possible to train one ESN to be a generator for a sum of multiple sines at different frequencies. While an additive superposition of sines is easy to handle for linear systems, nonlinear systems in general do not generate noninterfering additive couplings and a number of unwanted dynamic side effects can happen (Jaeger, Lukoševičius, Popovici, & Siewert, 2007). However, real-world time series are often

* Corresponding author at: Institute for Theoretical Computer Science, Technische Universität Graz, A-8010 Graz, Austria.
E-mail addresses: grh@mur.at (G. Holzmann), helmut.hauser@igi.tugraz.at (H. Hauser).
URL: http://grh.mur.at (G. Holzmann).

0893-6080/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2009.07.004

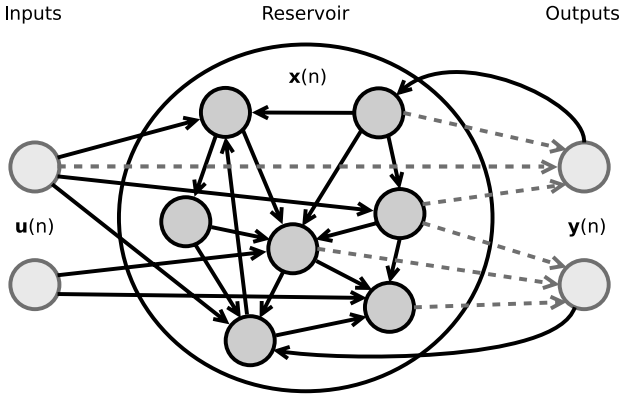

Fig. 1. Schematic overview of the reservoir computing approach with two inputs, two outputs and feedback from the outputs back to the reservoir. Solid black arrows indicate fixed, random connections and dotted arrows are trainable readout connections.

composed out of a superposition of multiple attractors, like for instance audio signals, where many sources are mixed together. One solution was proposed by Wustlich and Siewert (2007). They introduced filter neurons as an extension of ESNs with leaky-integrator units (Jaeger, 2007). If reservoir neurons have additional built-in filters, "specialized" neurons tuned to different frequency bands can emerge and more diverse echoes, which are able to model multiple attractors, will evolve. We build on this idea and propose an arbitrary order infinite impulse response (IIR) filter neuron, which allows to reuse well established structures from digital filter theory. The parameters of our suggested filters are different for each neuron and are fixed in advance, like the weights of connections in the reservoir. Therefore no additional training is necessary and a simple linear readout, like e.g. in the standard ESN formulations, can be used.

A similar approach was developed in parallel in Wyffels, Schrauwen, Verstraeten, and Stroobandt (2008). Compared to our networks, they used bigger reservoirs with pools of neurons with different filters, which were also tuned to the frequency spectrum of the desired response signal.

The second problem we address is the limited memory capacity of ESNs. The short-term memory, how many data points from the past are actually relevant for the computation of the current outputs, was analyzed in Jaeger (2002a), where it was shown that the memory capacity is restricted by the reservoir size. Especially in signal processing applications, which often use very high sampling rates, this would imply that one has to deal with very big reservoirs, which is not trivial to implement even on current computers. We introduce an alternative approach to handle long-term dependencies by adding trainable delays to the synaptic connections of readout neurons. This structure will be called delay&sum readout and was inspired by the delay&sum beamformer as used in signal processing with antenna or microphone array systems (see for instance Huang, Benesty, and Chen (2006)). Compared to standard echo state networks, where only weights of the readout are adjusted in training, a learning algorithm is introduced which modifies delays and weights of the delay&sum readout. The introduction of delays makes it possible to shift reservoir signals in time and is a computationally cheap method to vastly improve the memory capacity. Simulations in Section 4 show that also in chaotic time series prediction (Mackey–Glass System), where ESNs vastly outperformed all other techniques for nonlinear system modeling (Jaeger & Hass, 2004), a delay&sum readout is still able to improve the results. To the knowledge of the authors there exists no other generic method with comparable performance for sparse nonlinear system identification of higher-order systems.

Viewed from a biological perspective, also axons insert delays in connections between neurons. Although axonal transmission delays do not vary continually in the brain, a wide range of delay values have been observed (Paugam-Moisy, Martinez, & Bengio, 2008). Neuroscience experiments in Swadlow (1985) give also evidence to the variability of transmission delay values from 0.1 to 44 ms, see also Bringuier, Chavane, Glaeser, and Frégnac (1999) for a more recent investigation.

The contents of this paper is organized in five sections. Section 2 will extend the simple echo state network to ESNs with general IIR filter neurons in the reservoir. Next the delay&sum readout is introduced in Section 3 and two possible delay learning algorithms are presented. Four experiments, which demonstrate the usefulness of filter neurons and a delay&sum readout, are analyzed in Section 4. The first is a multiple superimposed oscillations (MSO) task, where filter neurons are mandatory, the second a sparse nonlinear system identification with long-term dependencies, where the delay&sum readout is able to vastly improve the memory capacity of ESNs. Afterwards a common benchmark task, the Mackey–Glass system, is discussed and then a real-world nonlinear audio prediction example is presented, where it is shown that the proposed structure is able to outperform state-of-the-art alternative methods. Finally Section 5 gives a short conclusion.

2. Filter neurons

Echo state networks (Jaeger, 2001) use sparsely and randomly connected neurons (the reservoir) and only the output layer (the readout) is adjusted during training. The output signals can be fed back as additional inputs, which makes it possible to train ESNs as oscillators or other generators. A schematic overview of ESNs with output feedback is presented in Fig. 1.

In this section we will briefly revise the standard formulations of ESNs, then the reservoir will be extended with additional build-in filters. This enables ESNs to model signals at different timescales and/or to learn multiple attractors in a single network.

2.1. From standard networks to low-pass neurons

Consider an ESN with L inputs, M outputs and a reservoir with N internal network units. At time $n = 1, 2, \ldots, n_{\max}$ the input vector is $\mathbf{u}(n)$ and the output $\mathbf{y}(n)$. The activations of internal units (neurons in the reservoir) are collected in an $N \times 1$ vector $\mathbf{x}(n) = (x_1(n), \ldots, x_N(n))^\intercal$ and internal connection weights in an $N \times N$ matrix $\mathbf{W}$, weights of input connections in an $L \times N$ matrix $\mathbf{W}^{\text{in}}$, feedback weights in an $M \times N$ matrix $\mathbf{W}^{\text{fb}}$ and output weights in an $(N + L) \times M$ matrix $\mathbf{W}^{\text{out}}$.

The activations of neurons in the dynamic reservoir are updated according to

$$\mathbf{x}(n+1) = f(\rho \mathbf{W}\mathbf{x}(n) + \mathbf{W}^{\text{in}}\mathbf{u}(n+1) + \mathbf{W}^{\text{fb}}\mathbf{y}(n) + \upsilon(n+1)) \quad (1)$$

where $\upsilon(n+1)$ is a small optional noise term and f is the nonlinear activation function of the reservoir neurons, for example tanh, linear or a step function. If the largest absolute eigenvalue $|\lambda|_{\max}(\mathbf{W}) = 1$ and the spectral radius ρ is $0 \leq \rho < 1$, the network states $\mathbf{x}(n)$ become asymptotically independent of initial conditions and depend only on input history, which is called the "echo state property" (Jaeger, 2001).

The readout neurons have connections to the reservoir units $\mathbf{x}(n)$ and inputs $\mathbf{u}(n)$, which can be combined into a vector $\mathbf{s}(n)$

$$\mathbf{s}(n) = (x_1(n), \ldots, x_N(n), u_1(n), \ldots, u_L(n))^\intercal. \quad (2)$$

Finally the new outputs $\mathbf{y}(n+1)$ can be calculated with

$$\mathbf{y}(n+1) = g(\mathbf{W}^{\text{out}}\mathbf{s}(n+1)) \quad (3)$$

where g is again a linear or nonlinear activation function.

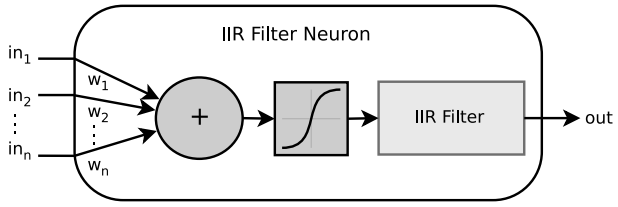

Fig. 2. Structure of a general IIR filter neuron, which is composed of an analog neuron with an additional IIR filter after the nonlinearity.

A training algorithm modifies the output weights $\mathbf{W}^{\text{out}}$, so that an optimal mapping from the internal states and inputs $\mathbf{s}(n)$ to a desired output target is achieved. Finding the right mapping is a linear regression task and if a mean squared error is used, the algorithm will find a global minimum of this error function. For more subtleties we refer the reader to Jaeger (2001) or Jaeger and Hass (2004).

The general idea of echo state networks can also be applied to networks with different neuron types. For example leaky-integrator units (Jaeger, 2007) incorporate information of the network states from the previous time steps when calculating the current state $\mathbf{x}(n)$, therefore the dynamics in the reservoir are slowed down. Such neurons are useful for learning slow dynamical systems. With leaky-integrator units the reservoir states are updated according to

$$\mathbf{x}(n+1) = (1-a\gamma)\mathbf{x}(n) + \gamma f(\rho\mathbf{W}\mathbf{x}(n) + \mathbf{W}^{\text{in}}\mathbf{u}(n+1) + \mathbf{W}^{\text{fb}}\mathbf{y}(n) + \upsilon(n+1))$$

where a and γ are additional leaking rate parameters. This model depends with the factor $(1-a\gamma)$ also on the previous state $\mathbf{x}(n)$ and has the echo state property if $|\lambda|_{\max}(\mathbf{W}) = 1, a > 0, \gamma > 0$, $0 \leq \rho < 1$ and $a\gamma \leq 1$ (Jaeger, 2007).

It was shown in Jaeger (2007) that the influence of one parameter γ, a, or ρ can be distributed over the others without a loss of essence. By setting $a = 1$, as done in Wustlich and Siewert (2007), this results in the state update equation

$$\mathbf{x}(n+1) = (1-\gamma)\mathbf{x}(n) + \gamma f(\rho\mathbf{W}\mathbf{x}(n) + \mathbf{W}^{\text{in}}\mathbf{u}(n+1) + \mathbf{W}^{\text{fb}}\mathbf{y}(n) + \upsilon(n+1)) \quad (4)$$

where the network has the echo state property if $0 \leq \rho < 1$ and $0 < \gamma \leq 1$. This model corresponds to a low-pass filter on the reservoir states, where the cutoff frequency can be determined by the parameter γ.

2.2. General IIR filter neurons

In Wustlich and Siewert (2007) two of the previously presented low-pass units from Eq. (4) were combined to get a band-pass neuron. Here we go one step further and propose an arbitrary order infinite impulse response (IIR) filter neuron, thus commonly known structures from filter theory can be recycled and one acquires a more fine-grained control over the filter's frequency and phase response. Fig. 2 shows the proposed IIR filter neuron, where the filter is calculated after the nonlinearity. The parameters of these filters are different for each neuron and are fixed in advance, like the weights of connections in the reservoir. Therefore no additional training is necessary and a simple linear readout, like e.g. in the standard ESN formulations, can be used.

A similar approach was also developed in parallel in Wyffels (2008). Compared to our networks, they used bigger reservoirs with pools of neurons with different filters, which were also tuned to the frequency spectrum of the desired response signal. We use a generic approach with no necessary special tuning and which results in rather small networks.

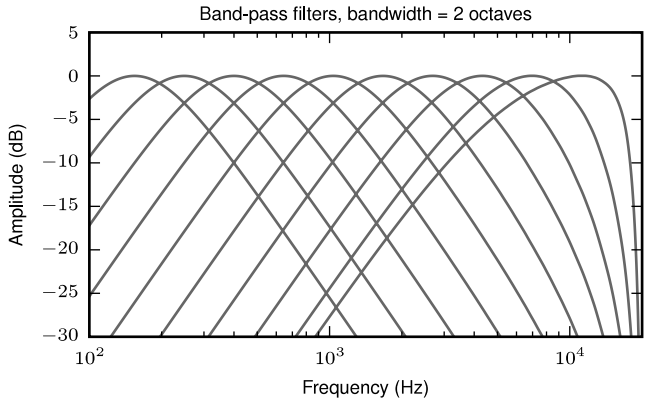

Fig. 3. Frequency response of band-pass filters, spaced logarithmically from 170 Hz to 19 000 Hz at a sampling rate of 44 100 Hz (note that the frequency axis is also logarithmic). In these examples every 10th neuron out of a reservoir with 100 neurons is printed and each filter has a bandwidth of 2 octaves.

A general IIR filter (Smith, 2007a) can be described with

$$a_0 y^*(n) = \sum_{j=0}^{K} b_j x^*(n-j) - \sum_{j=1}^{K} a_j y^*(n-j) \quad (5)$$

where vector $\mathbf{b} = (b_0, b_1, \ldots, b_K)$ contains weights of the feedforward path, vector $\mathbf{a} = (a_0, a_1, \ldots, a_K)$ of the feedback path, K determines the order of the filter and $y^*(n)$ and $x^*(n)$ are the filter output and input at time step n.

One independent IIR filter is now added to each neuron of the reservoir with size N. First the state is updated as in Eq. (1) to produce the signal $\tilde{\mathbf{x}}(n+1)$, which acts as an input for the filter (see Fig. 2), then one IIR filter with independent parameters is added to each neuron i, with $i = 1, \ldots, N$, to produce the new reservoir signal $x_i(n+1)$

$$\tilde{\mathbf{x}}(n+1) = f(\rho\mathbf{W}\mathbf{x}(n) + \mathbf{W}^{\text{in}}\mathbf{u}(n+1) + \mathbf{W}^{\text{fb}}\mathbf{y}(n) + \upsilon(n+1)) \quad (6a)$$

$$x_i(n+1) = \frac{1}{a_{i,0}}\left(\sum_{j=0}^{K} b_{i,j}\tilde{x}_i(n-j) - \sum_{j=1}^{K} a_{i,j}x_i(n-j)\right) \quad (6b)$$

where $b_{i,j}$ and $a_{i,j}$ are feedforward and feedback filter weights j for each reservoir neuron i and K determines the order of the filter as in Eq. (5). To be sure that the network has the echo state property, the maximum gain of each filter should be one, the spectral radius ρ within $0 \leq \rho < 1$ and the maximum absolute eigenvalue $|\lambda|_{\max}(\mathbf{W}) = 1$.

Since one can use the rich set of tools from filter design, it is possible to improve the performance for special tasks. We used second-order band-pass filters (biquad filters, $K = 2$) with a variable bandwidth, so that regions in the reservoir tuned to different frequency bands can evolve. The center frequencies are different for each neuron and are spaced logarithmically over a frequency range of interest as illustrated in Fig. 3. Filter parameters $\mathbf{a}$ and $\mathbf{b}$ are derived from analog prototypes and have been digitized using the bilinear transform as shown in Appendix A. However, any other band-pass filter design with a center frequency and filter bandwidth parameter could be used.

3. Delay&sum readout

In this section we address the limited memory capacity of ESNs, which is basically restricted by the size of the reservoir (Jaeger, 2002a). When modeling systems with long-term dependencies, one has to deal with very big networks, which is often not possible to implement even with modern computers. We propose an

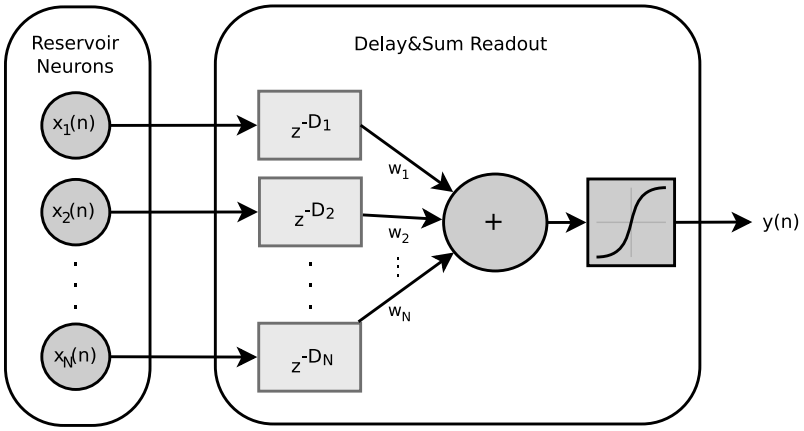

Fig. 4. Illustration of the delay&sum readout for one output neuron. The delay of a signal by D samples is denoted as z^{-D}. Each synaptic connection from a reservoir neuron to the output neuron consists of a trainable weight and a trainable delay.

alternative approach, which adds trainable delays in the synaptic connections of output neurons, called delay&sum (D&S) readout.

The delay&sum readout has to learn in addition to a weight also a delay from each neuron signal of the reservoir to each output signal. Two possible learning algorithms are presented. An overview of the readout is shown in Fig. 4.

Consider an echo state network with a delay&sum readout, N internal units, L inputs and M outputs, where the state update $\mathbf{x}(n+1)$ is computed for standard ESNs as in Eq. (1) or for reservoirs with IIR filter neurons as in Eq. (6). Reservoir and input signals are combined into a vector $\mathbf{s}(n)$ as defined in Eq. (2). The mth output y_m of a delay&sum readout, where $m = 1, \ldots, M$, can be calculated with

$$y_m(n) = g\left(\sum_{i=1}^{N+L} W^{\text{out}}_{i,m} s_i(n - D_{i,m})\right), \quad m = 1, \ldots, M. \tag{7}$$

Here $\mathbf{D}$ is a $(N + L) \times M$ matrix (same size as $\mathbf{W}^{\text{out}}$) with integer delays from each neuron and input signal to each output and g is the activation function of the output neuron. Note that for zero delays $\mathbf{D} = \mathbf{0}$ this is the same as Eq. (3), only written as a sum instead of the scalar product.

A learning algorithm should find optimal values for delays $\mathbf{D}$ and weights $\mathbf{W}^{\text{out}}$. In the following subsections two possible offline solutions will be presented. First a simple learning algorithm is developed (*D&S Learning Algorithm A*), which just calculates the correlation between reservoir and target signals for delay estimation, and afterwards a more advanced version, based on an expectation-maximization (EM) method, is introduced (*D&S Learning Algorithm B*).

3.1. D&S Learning Algorithm A: Simple method

The simple learning algorithm calculates a cross correlation between all reservoir/input signals $\mathbf{s}(n)$ and all output signals $\mathbf{y}(n)$ and takes always the highest peak as the trained delay.[1] Afterwards the reservoir and input signals $\mathbf{s}(n)$ are delayed by the estimated values from delay matrix $\mathbf{D}$ and then weights $\mathbf{W}^{\text{out}}$ can be computed offline by linear regression with the correctly delayed signals.

The method is summarized considering an ESN with N internal units, L inputs and M outputs:

1. Calculate delays between all reservoir/input signals $s_j(n)$ with $j = 1, \ldots, N + L$ and all target output signals $t_m(n)$ with $m = 1, \ldots, M$ and collect them into a $(N+L) \times M$ delay matrix $\mathbf{D}$. For each reservoir/input to output connection:
   - Estimate cross correlation $r_{s_j t_m}(\tau)$ between signals (Knapp & Carter, 1976) and invert output activation function g of the ESN
     $r_{s_j t_m}(\tau) = E\left[s_j(n) g^{-1}(t_m(n + \tau))\right]$
     where $E[.]$ denotes the expectation operator.

     Because of the finite observation time $n_{\max}$ the cross correlation function $r_{s_j t_m}(\tau)$ can only be estimated[2] by
     $$\hat{r}_{s_j t_m}(\tau) = \frac{1}{n_{\max}} \sum_{n=0}^{n_{\max}-1} s_j(n) g^{-1}(t_m(n + \tau)),$$
     $$\tau = 0, 1, \ldots, n_{\max} - 1. \tag{8}$$
   - Take the maximum of the correlation function as an estimate of the time delay and collect it in delay matrix $\mathbf{D}$ at position (j, m)
     $$D_{j,m} = \arg\max_{\tau} |\hat{r}_{s_j t_m}(\tau)| \tag{9}$$
     where $|.|$ denotes the absolute value.

   Repeat the whole procedure for each target output $t_m(n)$ where $m = 1, \ldots, M$.
2. Delay all reservoir and input signals according to the estimated values in delay matrix $\mathbf{D}$.
3. Dismiss data from an initial washout period $n_{\min}$ where $n < n_{\min}$ to get rid of initial transients. Collect the remaining correctly delayed input and network states $(s_1(n - D_{1,m}), \ldots, s_{N+L}(n - D_{N+L,m}))$ for each output target signal $t_m(n)$ row-wise into a $(n_{\max} - n_{\min}) \times (N + L)$ matrix $\mathbf{S}_m$, where $n_{\max}$ is the number of training examples, and the current target signal $g^{-1}(t_m(n))$ into a $(n_{\max} - n_{\min}) \times 1$ matrix $\mathbf{T}_m$.
4. To calculate optimal weights, for each output m compute pseudo-inverse $\mathbf{S}_m^{\dagger}$ and set
   $$\mathbf{W}^{\text{out}}_m = (\mathbf{S}_m^{\dagger} \mathbf{T}_m)^{\intercal}$$
   where $\mathbf{W}^{\text{out}}_m$ denotes the weights from all reservoir and input signals to output signal $t_m(n)$ (mth column of $\mathbf{W}^{\text{out}}$).

[1] Estimating the time delay between two signals is usually known as the time difference of arrival (TDOA) problem (Huang et al., 2006). There exist more advanced methods to estimate the time difference, like the generalized cross correlation (Azaria & Hertz, 1986; Knapp & Carter, 1976), but in most of the simulations they did not improve the performance of the learning algorithm.

[2] Note that the cross correlation estimation should be implemented in frequency domain to save computation time (Smith, 2007b).

3.2. D&S Learning Algorithm B: EM-based method

The simple learning algorithm presented above has some limitations. Consider for instance the following linear system

$$y(n) = x(n) + x(n-1) + \xi\,(x(n-200) + x(n-201))\,.$$

If the scaling factor ξ is small, the simple method will only find correct delays for $x(n)$ and $x(n-1)$, but not for $x(n-200)$ and $x(n-201)$.[3] Therefore a more advanced version, based on an expectation-maximization (EM) method as applied in a similar problem with antenna array systems (Pedersen, Fleury, & Mogensen, 1997), is proposed in this subsection. In contrast to the simple learning algorithm, which computes the correlation always between one reservoir/input signal compared to the whole output signal, this algorithm subtracts the influence of all other reservoir/input signals and estimates the delay and weight compared to this residuum. Thus it leads to an iterative procedure where single delays and weights are calculated sequentially.

The algorithm starts by estimating the delay between the first reservoir signal $x_1(n)$ and the target output $\kappa_1(n) = y(n)$ and then calculates the single weight of this synaptic connection. Afterwards the delayed and weighted signal $x_1(n)$ will be subtracted from the original output $y(n)$ to produce the new target signal $\kappa_2(n)$ for the second reservoir signal $x_2(n)$. In theory this procedure will be repeated for all reservoir and input signals until the algorithm converges, i.e. delays and weights do not change anymore. However, it turns out that in the case of ESNs the weights converge very slowly, while on the other hand the delays converge fast and do not change already after a few iterations. Therefore we proposed a combined approach in learning Algorithm B. First the EM algorithm is used for a few iterations to estimate delays of the readout and afterwards the weights will be recalculated offline with the correctly delayed reservoir and input signals.

The EM-based learning algorithm is illustrated in Fig. 5 and can be summarized considering an ESN with N internal units, L inputs and M outputs:

1. The EM-algorithm estimates delays and weights between all reservoir/input signals $s_j(n)$ with $j = 1, \ldots, N+L$ and all target output signals $t_m(n)$ with $m = 1, \ldots, M$ and collects them into a $(N+L) \times M$ delay matrix $\mathbf{D}$ and weight matrix $\mathbf{W}^{\text{out}}$. One iteration for updating $D_{j,m}$ and $W^{\text{out}}_{j,m}$ for one target output $t_m(n)$ reads:
   - *E-Step*:
     Subtract influence of all other reservoir and input signals[4] to get residuum $\kappa_{j,m}(n)$
     $$\kappa_{j,m}(n) = g^{-1}(t_m(n)) - \sum_{i=1}^{N+L} W^{\text{out}}_{i,m} s_i(n - D_{i,m}) + W^{\text{out}}_{j,m} s_j(n - D_{j,m}) \quad (10)$$
     where g^{-1} is the inversion of the ESN output activation function.
   - *M-Step*:
     Calculate cross correlation $\hat{r}_{s_j\kappa_{j,m}}(\tau)$ between signals $s_j(n)$ and $\kappa_{j,m}(n)$ as in Eq. (8) and estimate the time delay
     $$D_{j,m} = \arg\max_{\tau} |\hat{r}_{s_j\kappa_{j,m}}(\tau)|, \quad (11)$$
     afterwards compute weight $W^{\text{out}}_{j,m}$ with correctly delayed reservoir/input signal $\hat{s}_j(n) = s_j(n - D_{j,m})$
     $$W^{\text{out}}_{j,m} = (\hat{\mathbf{s}}_j^{\mathsf{T}}\hat{\mathbf{s}}_j)^{-1}\hat{\mathbf{s}}_j^{\mathsf{T}}\kappa_{j,m}. \quad (12)$$

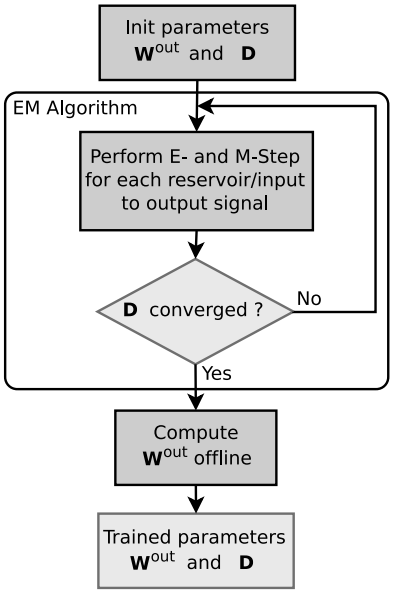

Fig. 5. EM-based learning algorithm. First weights and delays are estimated with the EM method, afterwards delays are fixed and weights are recalculated offline. For a description of the individual steps see Section 3.2.

   Perform E- and M-steps for each element of $\mathbf{s}(n)$ to complete one iteration.
2. Iterate step 1 (EM algorithm) until the delays are converged and repeat the whole procedure for each target output $t_m(n)$ where $m = 1, \ldots, M$.
3. Afterwards all delays are fixed and reservoir/input signals are delayed according to the estimated values in delay matrix $\mathbf{D}$.
4. Dismiss data from an initial washout period $n_{\min}$ where $n < n_{\min}$ to get rid of initial transients. Collect the remaining correctly delayed input and network states $(s_1(n - D_{1,m}), \ldots, s_{N+L}(n - D_{N+L,m}))$ for each output target signal $t_m(n)$ row-wise into a $(n_{\max} - n_{\min}) \times (N+L)$ matrix $\mathbf{S}_m$, where $n_{\max}$ is the number of training examples, and the current target signal $g^{-1}(t_m(n))$ into a $(n_{\max} - n_{\min}) \times 1$ matrix $\mathbf{T}_m$.
5. To calculate optimal weights, for each output m compute pseudo-inverse $\mathbf{S}_m^{\dagger}$ and put
   $$\mathbf{W}^{\text{out}}_m = (\mathbf{S}_m^{\dagger}\mathbf{T}_m)^{\mathsf{T}}$$
   where $\mathbf{W}^{\text{out}}_m$ denotes the weights from all reservoir and input signals to output signal $t_m(n)$ (mth column of $\mathbf{W}^{\text{out}}$).

4. Experiments

In the following section we present classical benchmark tasks and a real-world application, which will demonstrate where and how filter neurons and a delay&sum readout have advantages. The parameters for all four tasks are presented in Table 1. Other parameters, if necessary, will be described before presenting the simulation results.

4.1. Multiple superimposed oscillations

In the multiple superimposed oscillations (MSO) task an echo state network is trained to be a generator of a superposition of sine signals. This is an often studied task in ESN literature (Jaeger, 2007), because standard echo state networks are unable to learn functions composed of only two superimposed oscillators. The problem is that the dynamics of all reservoir neurons are coupled, while this task requires that different oscillations can be represented simultaneously by the internal state vector. It is easy for linear systems, for instance ESNs with only linear activation functions, to

[3] For example if $\xi < 0.1$, using an ESN with 100 neurons in the reservoir, the simple learning algorithm would not find the delays for $x(n-200)$ and $x(n-201)$.

[4] Note that the E-step can be implemented recursive to save computation time.

Table 1
ESN setups for the tasks of Section 4.

Parameters	MSO	System identification	Mackey–Glass	Audio prediction
Reservoir neurons	100	100	400	100
Reservoir connectivity	0.2	0.05	0.1	0.2
Spectral radius	0.8	0.8	0.95	0.8
Feedback weights between	[−0.1, 0.1]	no	[−0.56, 0.56]	[−0.5, 0.5]
Input weights between	No	[−0.1, 0.1]	[−0.14, 0.14]	No
Reservoir nonlinearity	tanh	tanh	tanh	tanh
Output nonlinearity	Linear	Linear	Linear	Linear
Filter neurons	Yes	No	No	Yes
Filter bandwidth	0.5 octaves	–	–	2 octaves

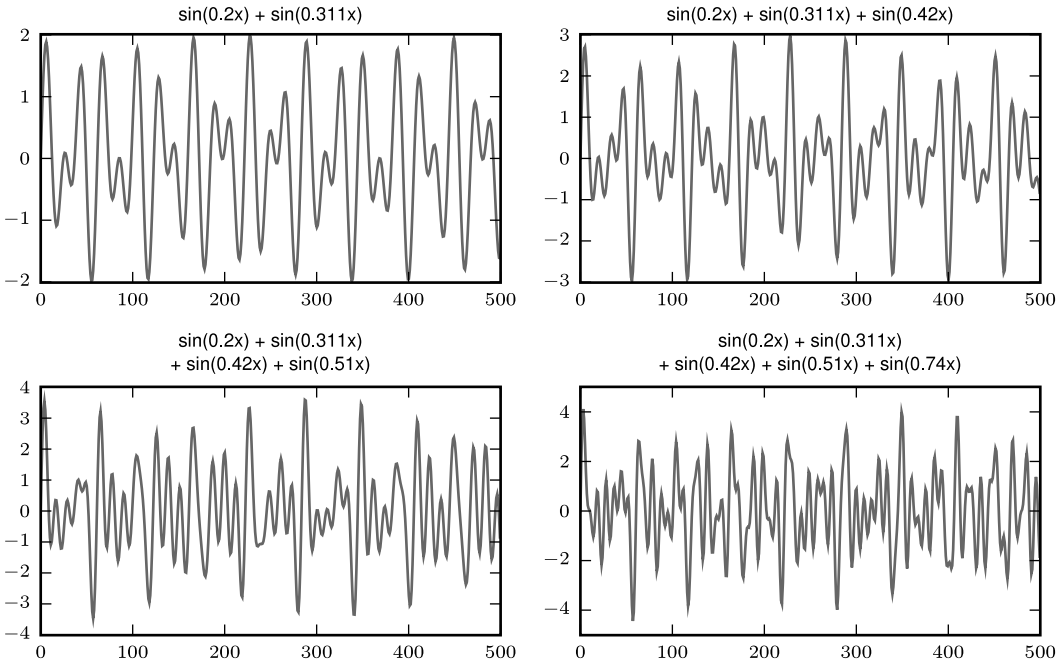

Fig. 6. 500 time steps of the target signals for the multiple superimposed oscillations (MSO) task from Schmidhuber et al. (2007) to learn functions composed of two to five sines.

generate multiple sines. The training error will be close to machine precision as long as there are at least two reservoir neurons for one sine component. However, in such a linear system the generated sine components are not stably phase-coupled and therefore do not result in asymptotically stable oscillations. In the presence of small perturbations, both the amplitude and relative phase of the sine components will go off on a random walk (Jaeger, 2007).

With our proposed filters in the reservoir, specialized neurons tuned to different frequencies exist and hence more diverse echoes emerge, which are able to generate a superposition of sinewaves. Other solutions for this problem were already presented for instance by Schmidhuber, Wierstra, Gagliolo, and Gomez (2007), who used an evolino-based long short-term memory (LSTM) recurrent neural network, or by Xue, Yang, and Haykin (2007), where an ESNs with multiple decoupled reservoirs was constructed. In the following we will compare the performance of an ESN with filter neurons and a delay&sum readout to the results of Schmidhuber et al. (2007) and Xue et al. (2007).

In Schmidhuber et al. (2007) a long short-term memory network is trained to learn functions composed of two to five sines:

$$\sin(0.2x) + \sin(0.311x) + \sin(0.42x) + \sin(0.51x) + \sin(0.74x).$$

All four target signals are shown in Fig. 6. The LSTM as well as the ESN network were first driven with 100 samples to washout initial transients, then network weights were calculated from steps 101 to 400 and the testing error was evaluated in time steps 401 to 700.

For all simulations the same echo state network with 100 IIR filter neurons in the reservoir was trained, a detailed setup is given in Table 1. Additionally the target signal was linearly rescaled into a range of [−1, 1], to be able to use the same setup for all four targets. To contain all sine frequencies necessary for the task, the center frequencies of neurons were logarithmically spaced between frequencies $0.19/(2\pi)$ and $0.75/(2\pi)$. Fig. 7 shows the $\text{NRMSE}_{\text{test}}$[5] for ESN and LSTM networks, averaged over 20 independent simulations with randomly created reservoirs. One can see that standard ESNs without filters are not able to learn multiple sines. When using filter neurons in the reservoir, the test error is much lower as with an LSTM network. A further performance boost can be achieved with an additional delay&sum readout as presented in Section 3. For instance with two sines, the delay&sum ESN is able to outperform the NRMSE of the LSTM network by a factor of 10^6.

The parameter which has a big influence on the performance is the bandwidth of the band-pass filters. A good value was found at a bandwidth of 0.5 octaves. Because of the short number of training and washout samples, the maximum delay in the delay&sum readout was restricted to 100 samples. Both learning algorithms, the *D&S Learning Algorithm A* from Section 3.1 and the EM-based *D&S Learning Algorithm B* from Section 3.2, performed similarly.

[5] The normalized root mean square error (NRMSE) is defined in Appendix B.

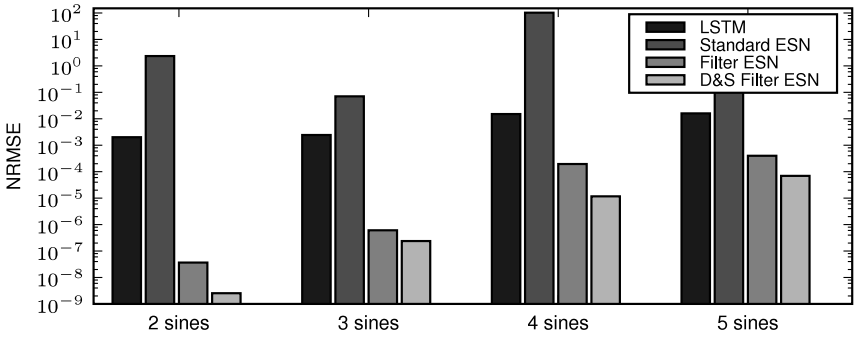

Fig. 7. Performance of the multiple superimposed oscillations (MSO) task. Weights are trained from time steps 101 to 400 and the $\text{NRMSE}_{\text{test}}$ is calculated from steps 401 to 700, averaged over 20 independent experiments. The LSTM network results from Schmidhuber et al. (2007) are compared to a standard ESN, to an ESN with filter neurons and finally to an ESN with filter neurons and a delay&sum readout. Note that the y-axis is logarithmic.

A second approach to learn the MSO task was proposed by Xue et al. (2007). Since they used a different training setup, no direct comparison to the previously presented results for the LSTM network was possible. In Xue et al. (2007) the same two sines problem

$$\sin(0.2x) + \sin(0.311x)$$

was simulated with a "decoupled echo state network with lateral inhibition", which basically consists of four almost independent ESN reservoirs with 100 neurons each, where each one has a different spectral radius (0.6, 0.7, 0.8 and 0.9). They trained the network from 600 time steps, after a washout period of 100 steps and the test error was calculated from steps 701 to 1000. A mean square error[6] of $\text{MSE}_{\text{test}} = 3 \times 10^{-3}$ was achieved.

For the same washout, training and test data size, using the same ESN setup as before with only 100 neurons (in contrast to 400 neurons in Xue et al. (2007)), the simulations resulted in an MSE_{test} of 3.49×10^{-14}, averaged over 20 independent experiments.[7] When also using a reservoir with 400 neurons, the MSE_{test} was even 3.04×10^{-20} and therefore significantly lower as in Xue et al. (2007).

4.2. Sparse nonlinear system identification

One useful property of the delay&sum readout is that the system is independent from a time-shift of the input signals. For instance the nonlinear systems $y(n) = x(n)x(n-1)$ and $y(n) = x(n-100)x(n-101)$ really need the same complexity in the reservoir, because the reservoir runs autonomous and a delay&sum readout simply learns the delay. Even better, with the D&S readout it is also possible to learn systems which depend on different time regions, as for example $y(n) = x(n)x(n-50)x(n-100)$, without the need of very big reservoirs for just modeling the time-shifts. In this subsection we will study two examples, where long-term dependencies are important. They will show that the delay&sum readout is able to vastly extend the short-term memory of ESNs (Jaeger, 2002a) and that the proposed networks are well suited for a sparse nonlinear system identification with long-term dependencies.

The first example is a modified version of the commonly known 10th-order nonlinear autoregressive moving average (NARMA) system benchmark task, which was introduced in Atiya and Parlos (2000) for a comparison of recurrent neural network architectures. Afterwards it was analyzed in various papers about echo state networks, for instance in Jaeger (2003). In Atiya and Parlos (2000) the original system is given as

$$y(n) = 0.3y(n-1) + 0.05y(n-1)\left[\sum_{i=1}^{10} y(n-i)\right] + 1.5x(n-10)x(n-1) + 0.1.$$

We used a modified version, where every delay in the system equation is multiplied by a factor τ to control the long-term dependence while keeping the essential structure

$$y(n) = 0.3y(n-\tau) + 0.05y(n-\tau)\left[\sum_{i=1}^{10} y(n-i\tau)\right] + 1.5x(n-10\tau)x(n-\tau) + 0.1. \quad (13)$$

For instance by setting $\tau = 10$ this results in a sparse NARMA system of order 100.

The same ESN setup as in Jaeger (2003) was employed[8] and is given in Table 1. This system was first driven with uniformly distributed white noise input between [0, 0.5] for a washout time of 2000 time steps, then output weights and delays were calculated from the next 5000 steps. Afterwards the trained model was simulated with a new random input signal and after discarding initial values, the test error was computed as a normalized root mean square error[9] from 2000 time steps. It was also helpful to use an additional bias input, because Eq. (13) has a constant offset of 0.1. The performance of an ESN with and without a delay&sum readout, dependent on time lag τ, is shown in Fig. 8.

This task demonstrates the possibility of a delay&sum readout, to vastly extend the short-term memory of ESNs. Without delays in the readout, the error increases fast. Fig. 8 shows also that for small τ, the simple *D&S Learning Algorithm A* performs better than the EM-based *D&S Learning Algorithm B*. However, the error of the EM-based method stays approximately constant when increasing τ. Simulations with IIR filter neurons in the reservoir showed, that they are not useful for this task. Usually the ESN output drifted away from the target signal and networks became unstable.

The best achieved performance in Atiya and Parlos (2000) was an $\text{NMSE}_{\text{train}}$ of 0.241 for the original 10th-order system[10] ($\tau = 1$),

[6] The mean square error (MSE) is defined in Appendix B.

[7] The target signal was again scaled into a range of $[-1, 1]$ and afterwards rescaled to the original range for MSE_{test} calculation.

[8] Note that this ESN used additional squared state updates in the reservoir as introduced in Jaeger (2003).

[9] The normalized root mean square error (NRMSE) is defined in Appendix B.

[10] According to Jaeger (2003), the authors of Atiya and Parlos (2000) miscalculated the $\text{NMSE}_{\text{train}}$, because they used a formula for zero-mean signals. In Jaeger (2003) the value was recalculated to $\text{NMSE}_{\text{train}} \approx 0.241$, which also agrees with the plots given in Atiya and Parlos (2000). For a definition of the normalized mean square error (NMSE) see Appendix B.

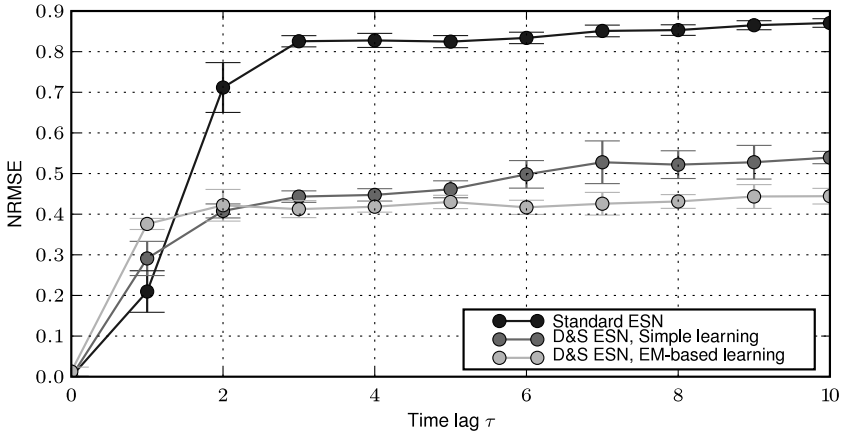

Fig. 8. NRMSE development for a nonlinear system identification task of Eq. (13) for different time lags τ. Each simulation was repeated ten times, mean and standard deviation for ESNs with a setup as in Table 1 are plotted. The figure shows the different error development for an ESN with and without a delay&sum readout, using the simple and EM-based learning method.

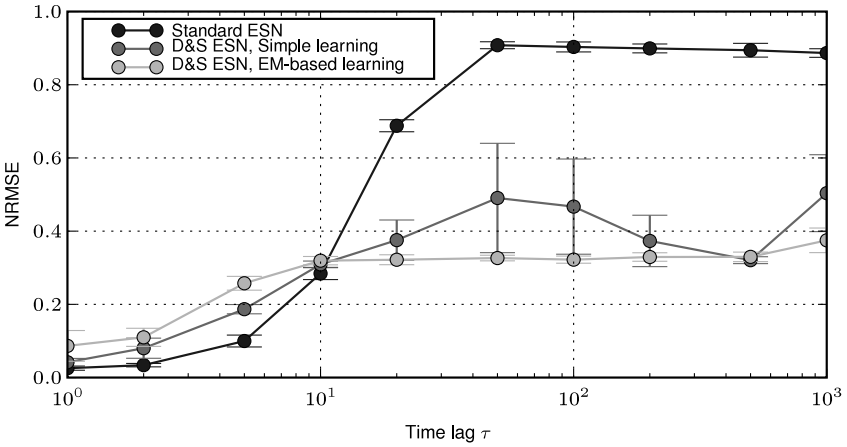

Fig. 9. NRMSE development for a nonlinear system identification task of Eq. (14) for different time lags τ. Each simulation was repeated ten times, mean and standard deviation for ESNs with a setup as in Table 1 are plotted. The figure shows the different error development for an ESN with and without a delay&sum readout, using the simple and EM-based learning method. Note that the x-axis is logarithmic.

which corresponds to an $\text{NRMSE}_{\text{train}} \approx 0.49$. Note that they calculated the error directly form training and not from testing data as done here. One can see that the test error of a 100 neuron delay&sum ESN for a 100th-order system ($\tau = 10$) is still better than the best training error for the 10th-order system in Atiya and Parlos (2000).[11] When using bigger reservoirs and more training data it is even possible to further increase the prediction performance.

As a second example, a nonlinear moving average (NMA) system is analyzed, which incorporates information from different time regions. The system is given as

$$y(n) = \left(\sum_{i=0}^{3} x(n-i)\right)\left(\sum_{i=0}^{2} x(n-\tau-i)\right) \times \left(\sum_{i=0}^{2} x(n-2\tau-i)\right) \tag{14}$$

where parameter τ is the time lag between the three groups in time.

Again the setup from Table 1 was used and the system was driven with uniformly distributed white noise input between [0, 0.5]. After a washout period of 2500 time steps the network was trained from the next $5000 + 2\tau$ steps.[12] The trained model was then simulated with a new random input signal and after discarding $100+2\tau$ initial values the test error was computed from 2000 time steps. Fig. 9 shows the error development dependent on time lag τ for different delay learning algorithms.

The interesting result is that as time lag τ becomes bigger than 10 (which means a maximum delay of 22 steps in the system), the $\text{NRMSE}_{\text{test}}$ of a D&S ESNs approximately remains at a value above 0.3, whereas the error of standard ESNs further increases. Fig. 9 show also, that the EM-based *D&S Learning Algorithm B* has a more constant performance and a much smaller standard deviation than the simple *D&S Learning Algorithm A*.

4.3. Mackey–Glass system

One classical benchmark task for time series modeling is the Mackey–Glass delay differential equation. It is given for instance in Jaeger (2001) as

$$\dot{y}(t) = \alpha \frac{y(t-\tau)}{1 + y(t-\tau)^{\beta}} - \gamma y(t) \tag{15}$$

and usually the parameters are set to $\alpha = 0.2$, $\beta = 10$ and

[11] An additional unpublished result with a recurrent neural network trained as in Feldkamp, Prokhorov, Eagen, and Yuan (1998) is mentioned in Jaeger (2003), which reached a slightly better precision as the ESN setup from Table 1 for the $\tau = 1$ task. Unfortunately the exact error value is not given in Jaeger (2003).

[12] Training size depends on τ to be able to compare higher values of τ.

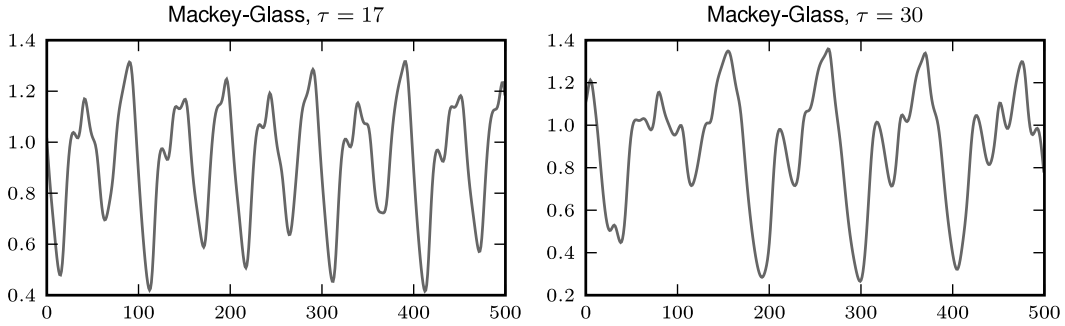

Fig. 10. 500 steps of a Mackey–Glass sequence for $\tau = 17$ (left) and $\tau = 30$ (right). Both systems have a chaotic attractor and are standard benchmark tasks in research on time series prediction.

$\gamma = 0.1$. The system has a chaotic attractor if $\tau > 16.8$ and in most of the benchmarks τ is set to $\tau = 17$ or $\tau = 30$. A plot for both settings is shown in Fig. 10. In discrete time, the Mackey–Glass delay differential equation can be approximated by

$$y(n+1) = y(n) + \delta\left(0.2\frac{y(n-\tau/\delta)}{1+y(n-\tau/\delta)^{10}} - 0.1y(n)\right) \tag{16}$$

with a stepsize δ. Here the stepsize is set to the same value $\delta = 1/10$ as in Jaeger (2001) and Jaeger and Hass (2004), to be able to compare the results.[13] The time series was afterwards rescaled into a range of $[-1, 1]$ by transforming it with $y_{ESN}(n) = tanh(y(n)-1)$ as done in Jaeger (2001), so that it can be used to train ESNs with tanh activation functions in the reservoir.

The ESN setup is summarized in Table 1, which is almost the same as in Jaeger (2001), except that they used leaky-integrator neurons. We chose standard units, because filter neurons were not useful in this task. They would divide the reservoir in autonomous sub-networks, which are able to learn multiple attractors. However, here only one single attractor should be trained and therefore the performance with filter neurons in the reservoir was worse than without.

Training sequences were generated from Eq. (16) for $\tau = 17$ and $\tau = 30$ as in Jaeger (2001). In both cases first the network was trained from 2000 samples, after an initial washout time of 1000 steps, then the same experiment was repeated for a training size of 20 000 samples. Furthermore a small noise term $\upsilon(n)$ in the state update Eq. (1) was necessary during training to get stable attractors.

The mean squared error of a specific prediction horizon is used to compare the results to other modeling techniques. A common value for the Mackey–Glass system is a 84 or 120 steps prediction (Jaeger, 2001). For instance to calculate an NRMSE_{84}, the trained echo state network is driven with a 1000 sample time series of different original Mackey–Glass signals (teacher forcing) and afterwards simulated for 84 time steps to get the predicted output $\hat{y}(n+84)$. This value can be compared, after being re-transformed to the original scale, with the original Mackey–Glass signal $y(n+84)$. To average over different initial states, the same echo state network was run 100 times with new teacher forcing signals. At each run i the predicted output $\hat{y}_i(n+84)$ and target signal $y_i(n+84)$ were collected and the NRMSE_{84} was calculated with

$$\text{NRMSE}_{84} = \sqrt{\frac{1}{100\sigma^2}\sum_{i=1}^{100}(y_i(n+84) - \hat{y}_i(n+84))^2} \tag{17}$$

where σ^2 is the variance of the original attractor signal.

[13] Actually the Mackey–Glass sequence was generated from the same code as in Jaeger and Hass (2004), which can be downloaded from http://www.faculty.iu-bremen.de/hjaeger/pubs/ESNforMackeyGlass.zip.

Table 2
NRMSE of the Mackey–Glass sequence with $\tau = 30$. Results are for a 84 and 120 step prediction and for a training length of 2000 (2 K) and 20 000 (20 K) steps, always with a washout time of 1000 (1 K) samples. The error was calculated from 100 independent runs.

Method	NRMSE_{84}	NRMSE_{120}
Standard ESN, 1 K + 2 K	0.136	0.217
D&S ESN, 1 K + 2 K	0.0311	0.0486
Standard ESN, 1 K + 20 K	0.0718	0.120
D&S ESN, 1 K + 20 K	0.02	0.0317

The results for a 84 and 120 step prediction with $\tau = 30$ are shown in Table 2. A uniform noise term $\upsilon(n)$ was added during training in the state update equation. For the 2000 step training sequence it was chosen between $[-10^{-5}, 10^{-5}]$ and for the 20 000 step training between $[-10^{-8}, 10^{-8}]$. Delays were calculated with the *D&S Learning Algorithm A*, because the EM-based *D&S Learning Algorithm B* did not improve the prediction performance. In Jaeger (2001) an NRMSE_{84} of 0.11 for $\tau = 30$ was achieved, which is comparable to the results reported here for standard ESNs with an NRMSE_{84} of 0.136. Fig. 11 shows the further development of the error for $\tau = 17$ and $\tau = 30$, up to a prediction horizon of 400 time steps. The advantage of a delay&sum readout is bigger in the $\tau = 30$ task, because the sequence seems to be much more difficult to learn.

For a comparison of the Mackey–Glass time series prediction to other techniques, one has to consider (Jaeger & Hass, 2004), where it was shown that for the $\tau = 17$ task ESNs vastly outperform all other known methods. A bigger ESN with 1000 reservoir neurons was trained in that paper and resulted in an NRMSE_{84} of 6.309×10^{-5}, compared to a previously best result of an NRMSE_{84} of 1.995×10^{-2}. Results from various other techniques are summarized in Gers, Eck, and Schmidhuber (2001).

A comparison is more interesting for the $\tau = 30$ task. According to Jaeger and Hass (2004), the best previous result was in Feldkamp et al. (1998), where a multilayer perceptron with output feedback was trained with a refined version of backpropagation through time (extended Kalman filter multi-stream training). They calculated a root mean square error[14] for a 120 step prediction and achieved an RMSE_{120} of 0.04. An ESN with a delay&sum readout achieves an NRMSE_{120} of 0.0317, as shown in Table 2, which corresponds to an RMSE_{120} of 0.00891. When using bigger ESNs with 700 neurons and a refined version[15] as in Jaeger and Hass (2004), the error can be further reduced to an RMSE_{120} of 0.0072 for 2000 and an RMSE_{120} of 0.00265 for 20 000 training steps, which improves the result in Feldkamp et al. (1998) by a factor of approximately 15.

[14] For a definition of the root mean square error (RMSE) see Appendix B.

[15] In the refined version ten independently created ESNs, 700 neurons each, were trained one after the other, but in simulation the outputs of all 10 networks were averaged. The ten networks got also the output feedback from the averaged value.

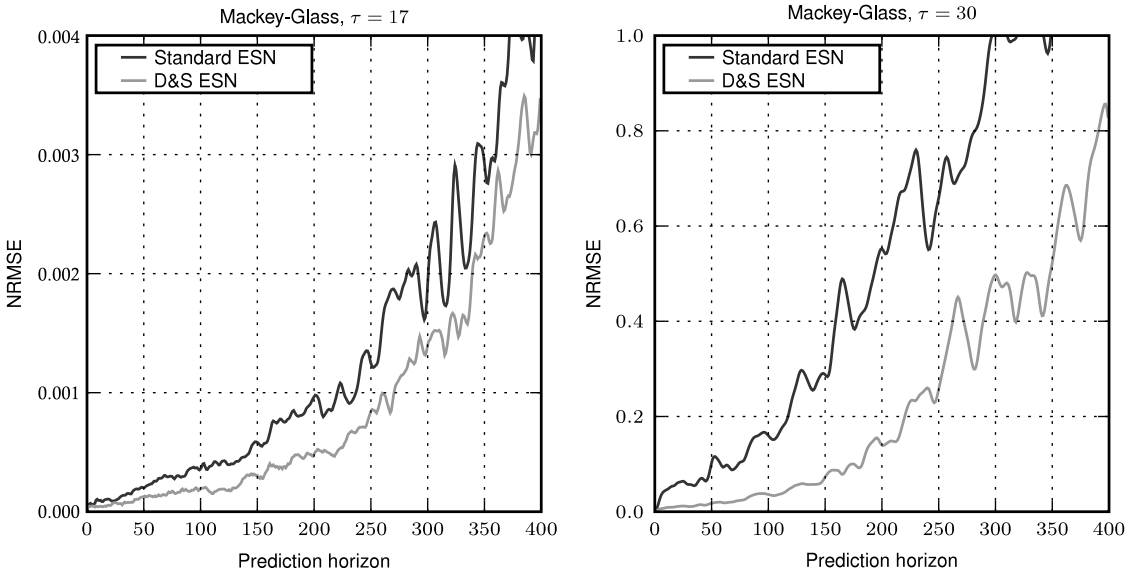

Fig. 11. NRMSE development of the Mackey–Glass sequence with $\tau = 17$ (left) and $\tau = 30$ (right), for a prediction horizon up to 400 time steps. Delays and weights of the readout were trained from 2000 samples. The performance boost of a delay&sum readout is bigger in the $\tau = 30$ task.

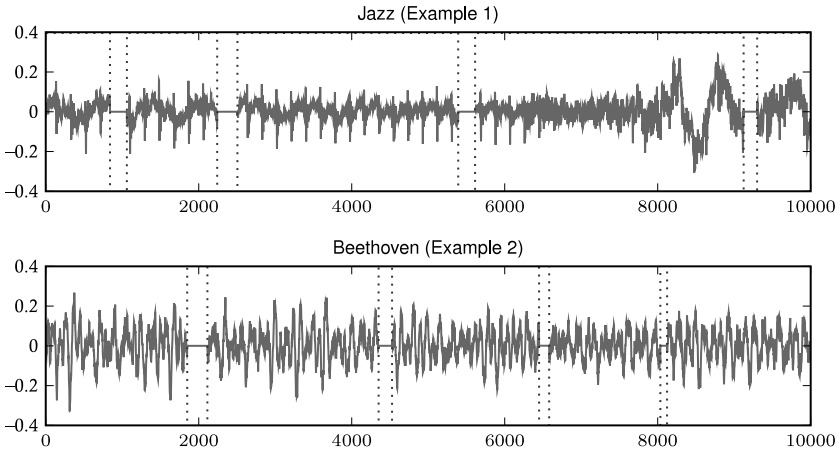

Fig. 12. Extract from the two audio examples of the nonlinear audio prediction task. The first one is a jazz quartet, the second an orchestra composition by Beethoven. They were taken from Ausserlechner (2006) and dropouts, generated from a distribution as presented in Table 3, are marked with dotted lines.

4.4. Nonlinear audio prediction

This task presents a real-world audio prediction application of echo state networks and compares the performance to alternative state-of-the-art models. In audio prediction one tries to forecast future samples out of a given history horizon. Such methods are necessary for instance in audio restoration, whenever a sequence of consecutive samples is missing or when impulsive noise appears.

The scenario of this task, which was defined in Ausserlechner (2006), is a wireless signal transmission, where short dropouts can occur due to transmission problems. Short gaps with a length of 2 to 6 ms are assumed, the exact number and lengths of dropouts are shown in Table 3. After each gap, the signal is predicted for further 200 samples and crossfaded with the original one to avoid discontinuities at the transition points (see Fig. 14). Two different audio examples were taken from Ausserlechner (2006), both with a duration of five seconds and a sampling rate of 44 100 Hz. The first one is a short recording of a jazz quartet and the second an orchestra composition by Beethoven. Short extracts of both, with random dropouts as defined in Table 3, are shown in Fig. 12.

The prediction performance of ESNs for both examples is compared to two commonly used, state-of-the-art alternative models:

Table 3
Dropout distribution for the nonlinear audio prediction task, taken from listening test three of Ausserlechner (2006). The location was randomly chosen within an audio file with a duration of five seconds and a sampling rate of 44 100 Hz, resulting in two percent of corrupted audio.

Length (ms)	Length (samples)	Occurrences
2	88	9
3	132	6
4	176	5
5	220	4
6	264	3

a pattern matching algorithm (PatMat) and a linear autoregressive (AR) model.

Pattern matching algorithms (Goodman, Lockhart, Wasern, & Wai-Choong, 1986; Niedzwiecki & Cisowski, 2001) use a signal template, usually just before the dropout, which is compared to areas in the past of the audio signal (the search window). Then the area within this search window with the highest similarity to the audio template is selected and copied into the dropout region. Many different versions of pattern matching algorithms exist and were evaluated in Ausserlechner (2006). Here only the one with the best prediction result, which is based on waveform differences

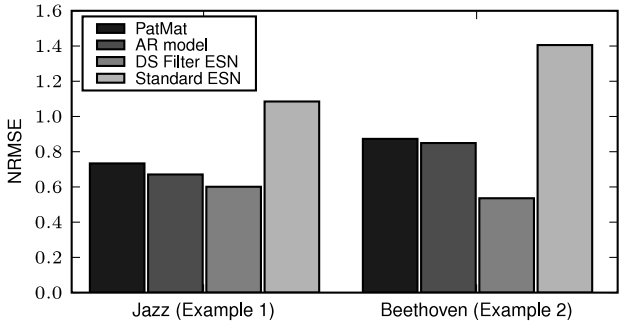

Fig. 13. NRMSE values calculated from the dropout regions of both audio examples of the nonlinear audio prediction task. Standard ESNs were not able to produce good predictions, but ESNs with filter neurons and a delay&sum readout showed in general a slightly better performance than other described state-of-the-art techniques.

and defined in Eq. (6) of Goodman et al. (1986), will be compared to our approach.

Additionally a linear autoregressive model (Godsill & Rayner, 1997), which is widely used for signal modeling, was considered. The order of this linear predictor should be fixed to a value high enough, so that complex signals can be represented. Here the order was set to 300. Furthermore no audio signal is truly stationary, therefore it is necessary to train the model from a relatively short block of samples and a block length of 3000 samples was found to be optimal. Finally filter coefficients of the AR model were estimated with a least squares approach based on those 3000 samples as described in Section 3.3.1 of Godsill and Rayner (1997).

For both audio examples a standard echo state network and an ESN with filter neurons and a delay&sum readout was trained. The exact setup is shown in Table 1. Because audio signals are not stationary, the training size is the most critical parameter in this task. Standard ESNs were first driven by 500 samples and afterwards output weights were calculated from the next 1000 steps. For D&S ESNs it was possible to use a bigger washout time of 3000 samples and the readout was trained from the next 3000 time steps.

A uniform noise term $\upsilon(n)$ was added during training in the state update equation, so that no unstable networks were produced in simulations. For the standard ESN the noise was drawn from $[-0.0001, 0.0001]$ and for the D&S ESN with filter neurons from $[-0.0002, 0.0002]$. Note that for standard ESNs the final error was lower, if the same amount of noise was added in training as well as in testing, otherwise the output signal energy decreased and the audible results were much worse.

For D&S ESNs, the maximum delay was restricted to 1000 samples, because the training size should be kept small. Here the EM-based *D&S Learning Algorithm B* from Section 3.2 performed a little bit better than *D&S Learning Algorithm A* and has been used for the simulations and the resulting figures. Band-pass filters in the reservoir were spaced logarithmically between 60 Hz and 11 000 Hz, to contain all frequencies important for the task. Furthermore a bandwidth of 2 octaves was found to be optimal.

The performance of all algorithms was benchmarked by calculating the NRMSE[16] between model output and original signals, only from data in the dropout regions. Results for both audio examples are presented in Fig. 13. One can see that standard ESNs are not able to produce good predictions, similar as in the multiple superimposed oscillations task from Section 4.1, even when using much bigger reservoirs or more training data. ESNs with filter neurons and a delay&sum readout showed in general a slightly better performance than all other methods, especially in the second audio example. Prediction examples from a delay&sum ESN, a standard ESN, a linear autoregressive model and from the pattern matching algorithm are shown in Fig. 14.

5. Conclusion

We proposed two enhancements of echo state networks: general IIR filter neurons in the reservoir and a delay&sum readout. Both methods overcome limitations of the original formulation and drastically augment the possibilities of ESNs. Simulations with benchmark datasets and audio signals demonstrated the usefulness of the developed techniques.

With the proposed filter neurons in the reservoir, it was possible to model multiple attractors with one single network. When using filters, "specialized" neurons are tuned to different frequencies and more diverse echoes can evolve in the reservoir because not all units are coupled. The parameters of these filters are different for each neuron and are fixed in advance, therefore no additional training is necessary and any readout, like for instance a linear or a delay&sum readout, can be used with our suggested structure. Simulations in Section 4 showed, that filter neurons are mandatory for the multiple superimposed oscillations (MSO) and the nonlinear audio prediction task. Our presented model was able to outperform state-of-the-art alternative algorithms.

The delay&sum readout adds trainable delays in the synaptic connections of readout neurons. It vastly increases the short-term memory capability of echo state networks and is an efficient approach to handle long-term dependencies. We introduced two learning algorithms, which modify delays and weights of the readout neurons. This readout was able to boost the performance in all presented simulations. Especially in the sparse nonlinear system identification with long-term dependencies and the chaotic time series prediction task (Mackey–Glass system), it was therefore possible to get an improvement compared to all other known approaches.

Acknowledgements

This work was written under the support of project # FP6-015879 (FACETS) of the European Union, project # P17229 of the Austrian Science Fund FWF and under the partial support of Integrated Graduate Program on Human-Centric Communication at the Technical University Berlin. The authors would also like to thank Welf Wustlich, Dmitriy Shutin and Wolfgang Maass for fruitful discussions and ideas.

Appendix A. Filter parameters

Band-pass filters are implemented as second-order IIR systems and are derived from analog prototypes, which have been digitized using the bilinear transform (Bristow-Johnson, 2008) to finally get the following parameters for coefficients **a** and **b** of Eq. (5):

$$\omega_0 = \frac{2\pi f_0}{f_s} \tag{18}$$

$$\alpha = \sin(\omega_0)\sinh\left(\frac{ln(2)}{2}\phi\frac{\omega_0}{\sin(\omega_0)}\right) \tag{19}$$

$$\begin{aligned} &b_0 = \alpha, \quad b_1 = 0, \quad b_2 = -\alpha, \\ &a_0 = 1+\alpha, \quad a_1 = -2\cos(\omega_0), \quad a_2 = 1-\alpha \end{aligned} \tag{20}$$

where f_0 is the center frequency of the band-pass filter, f_s the sampling rate and ϕ the bandwidth in octaves between -3 dB cutoff frequencies. Fig. 3 shows a plot of the here presented band-pass filters.

Appendix B. Error measures

Different error measures are used to be able to compare the results to values given in literature. The mean square error (MSE)

[16] The normalized root mean square error (NRMSE) is defined in Appendix B.

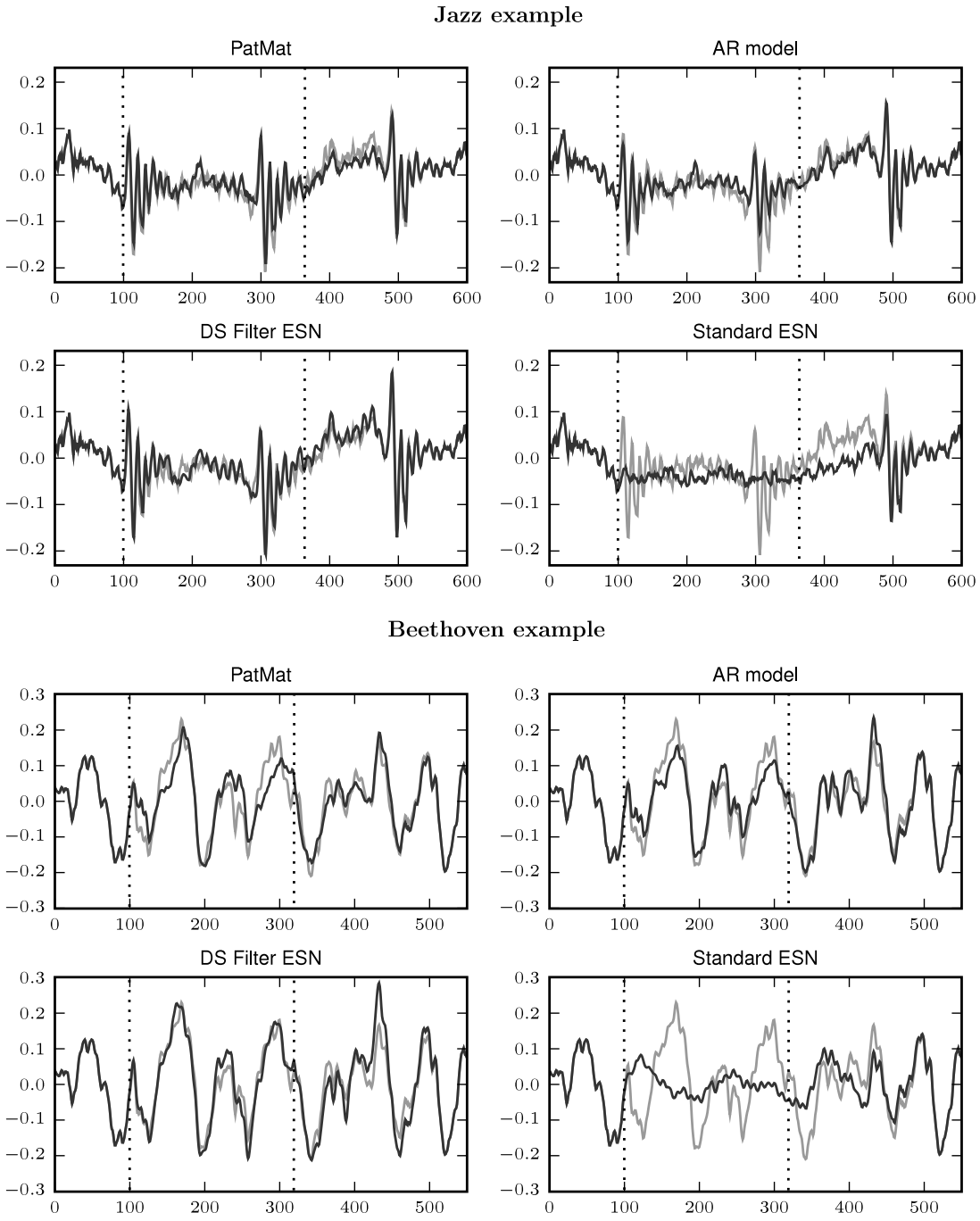

Fig. 14. Example dropouts from both audio signals of the nonlinear audio prediction task. The beginning and ending of missing data is marked with dotted lines. After the dropout the signal is predicted for further 200 samples and crossfaded with the original one to avoid discontinuities at the transition points. The original signal is plotted in light gray, the predicted signal in dark gray.

between a target signal $y_{\text{target}}(n)$ and a generated signal $y(n)$ for $n = 0, \ldots, N-1$ is calculated as

$$\text{MSE} = \frac{1}{N}\sum_{n=0}^{N-1}\left(y_{\text{target}}(n) - y(n)\right)^2. \tag{21}$$

Another possibility is to take the square root of the MSE, which results in the root mean square error (RMSE)

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{n=0}^{N-1}\left(y_{\text{target}}(n) - y(n)\right)^2}. \tag{22}$$

When using signals with different amplitudes, it is more meaningfull to normalize the error with the variance of the target signal σ^2_{target}, which is called the normalized mean square error (NMSE)

$$\text{NMSE} = \frac{1}{N\sigma^2_{\text{target}}}\sum_{n=0}^{N-1}\left(y_{\text{target}}(n) - y(n)\right)^2. \tag{23}$$

Finally, by taking the square root of the NMSE, one gets the normalized root mean square error (NRMSE)

$$\text{NRMSE} = \sqrt{\frac{1}{N\sigma^2_{\text{target}}}\sum_{n=0}^{N-1}\left(y_{\text{target}}(n) - y(n)\right)^2}. \tag{24}$$

References

Atiya, A. F., & Parlos, A. G. (2000). New results on recurrent network training: Unifying the algorithms and accelerating convergence. *IEEE-NN*, *11*(3), 697.

Ausserlechner, H. (2006). Untersuchungen von dropout-concealment-algorithmen. Institute for Electronic Music and Acoustics, Graz, Austria. Diplomarbeit.

Azaria, M., & Hertz, D. (1986). Time delay estimation by generalized cross correlation methods. *Acoustics, Speech, and Signal Processing; IEEE Transactions on Signal Processing*, *32*, 285–380.

Bringuier, V., Chavane, F., Glaeser, L., & Frégnac, Y. (1999). Horizontal propagation of visual activity in the synaptic integration field of area 17 neurons. *Science*, *283*, 695–699.

Bristow-Johnson, R. (2008). Cookbook formulae for audio eq biquad filter coefficients. Online resource, accessed 14.4.2008.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press.

Feldkamp, L., Prokhorov, D., Eagen, C., & Yuan, F. (1998). Enhanced multi-stream kalman filter training for recurrent networks. *Nonlinear Modeling: Advanced Black-Box Techniques*, 29–53.

Gers, F. A., Eck, D., & Schmidhuber, J. (2001). Applying LSTM to time series predictable through time-window approaches. *Lecture Notes in Computer Science*, *2130*, 669+.

Godsill, S., & Rayner, P. (1997). *Digital audio restoration*. Springer.

Goodman, D., Lockhart, G., Wasern, O., & Wai-Choong, W. (1986). Waveform substitution techniques for recovering missing speech segments in packet voice communications. *Acoustics, Speech, and Signal Processing, IEEE Transactions*, *34*, 1440–1448.

Huang, Y., Benesty, J., & Chen, J. (2006). *Acoustic MIMO signal processing*. Berlin, Heidelberg: Springer-Verlag.

Jaeger, H. (2001). The "echo state" approach to analysing and training recurrent neural networks. *Tech. rep. no. 148*. German National Research Center for Information Technology Bremen.

Jaeger, H. (2002a). Short-term memory in echo state networks. *Tech. rep. no. 152*. German National Research Center for Information Technology Bremen.

Jaeger, H. (2002b). A tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the "echo state network" approach. *Tech. rep. no. 159*. German National Research Center for Information Technology Bremen.

Jaeger, H. (2003). Adaptive nonlinear system identification with echo state networks. In *Advances in neural information processing systems 15* (pp. 593–600). MIT Press.

Jaeger, H., & Hass, H. (2004). Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, *304*, 78–80.

Jaeger, H., Lukoševičius, M., Popovici, D., & Siewert, U. (2007). 2007 special issue: Optimization and applications of echo state networks with leaky-integrator neurons. *Neural Networks*, *20*(3), 335–352.

Knapp, C., & Carter, G. (1976). The generalized correlation method for estimation of time delay. *Acoustics, Speech, and Signal Processing; IEEE Transactions on Signal Processing*.

Maass, W., & Bishop, C. (2001). *Pulsed neural networks*. MIT Press.

Maass, W., Natschlaeger, T., & Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, *14*(11), 2531–2560.

Niedzwiecki, M., & Cisowski, K. (2001). Smart copying — a new approach to reconstruction of audio signals. *IEEE Transactions on Signal Processing*, *49*, 2272–2282.

Paugam-Moisy, H., Martinez, R., & Bengio, S. (2008). Delay learning and polychronization for reservoir computing. *Neurocomputing*, *71*(7–9), 1143–1158.

Pedersen, K. I., Fleury, B. H., & Mogensen, P. E. (1997). High resolution of electromagnetic waves in time-varying radiochannels. In *Personal, indoor and mobile radio communications, 1997. 'Waves of the year 2000', Vol. 2* (pp. 650–654).

Schmidhuber, J., Wierstra, D., Gagliolo, M., & Gomez, F. (2007). Training recurrent networks by evolino. *Neural Computation*, *19*(3), 757–779.

Siegelmann, H. T., & Sontag, E. D. (1991). Turing computability with neural nets. *Applied Mathematics Letters*, *4*, 77–80.

Smith, J. O. (2007a). *Introduction to digital filters with audio applications*. W3K Publishing, Online book, accessed 14.4.2008.

Smith, J. O. (2007b). *Mathematics of the discrete fourier transform (dft)*. W3K Publishing, Online book, accessed 24.4.2008.

Swadlow, H. (1985). Physiological properties of individual cerebral axons studied in vivo for as long as one year. *Journal of Neurophysiology*, *54*, 1346–1362.

Verstraeten, D., Schrauwen, B., D'Haene, M., & Stroobandt, D. (2007). 2007 special issue: An experimental unification of reservoir computing methods. *Neural Networks*, *20*(3), 391–403.

Wustlich, W., & Siewert, U. (2007). Echo-state networks with band-pass neurons: Towards generic time-scale-independent reservoir structures. *PLANET Intelligent Systems GmbH*, posted on the reservoir computing mailing list, accessed 23.3.2008.

Wyffels, F., Schrauwen, B., Verstraeten, D., & Stroobandt, D. (2008). Band-pass reservoir computing (pp. 3203–3208).

Xue, Y., Yang, L., & Haykin, S. (2007). 2007 special issue: Decoupled echo state networks with lateral inhibition. *Neural Networks*, *20*(3), 365–376.