LETTER Communicated by Peter Tiňo

Design Strategies for Weight Matrices of Echo State Networks

Tobias Strauss
tobias.strauss@uni-rostock.de
Welf Wustlich
welf.wustlich@uni-rostock.de
Roger Labahn
roger.labahn@uni-rostock.de
Department of Mathematics, University of Rostock, Rostock 18057, Germany

This article develops approaches to generate dynamical reservoirs of echo state networks with desired properties reducing the amount of randomness. It is possible to create weight matrices with a predefined singular value spectrum. The procedure guarantees stability (echo state property). We prove the minimization of the impact of noise on the training process. The resulting reservoir types are strongly related to reservoirs already known in the literature. Our experiments show that well-chosen input weights can improve performance.

1 Introduction

A considerable problem in theory and practice of artificial recurrent neural networks (RNN) is the training of recurrent connections. Various gradient descent algorithms (e.g., BPTT or RTRL; see Jaeger, 2002) have been developed, but they suffer from slow convergence and local minima. An approach to avoid these problems is the echo state approach (Jaeger, 2001a). Echo state networks (ESN) consist of three layers: an input layer; a hidden layer with recurrent connections, called a dynamical reservoir (DR); and the output layer. In contrast to ordinary networks, a fundamental concept of ESNs is to train only the output connections; all other connections remain fixed. This simple but effective training approach ensures good performance in many tasks (Lukoševičius & Jaeger, 2009).

The state of the art in creating dynamical reservoirs is choosing the connections randomly and scaling the spectral radius of the related weight matrix to (often slightly) lower than 1 afterward. This is done to satisfy the necessary condition for stability: the echo state property. A criticism of this method is that it does not ensure stability since the sufficient condition is much more restrictive and often unregarded. But it is practically more relevant that random matrices damp echos with different directions unequally.

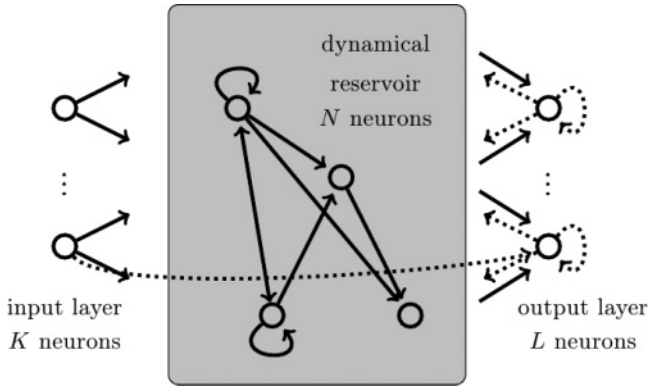

Figure 1: Layers of an echo state network. Dashed connections are ignored in our theoretical investigations. Feedback (output-to-hidden) connections are used only in section 5.3.

This means that certain parts of the signal might vanish quickly compared to other parts. We aim to construct reservoirs preserving a variety of echos.

An approach to creating reservoirs with certain dynamics is given in Ozturk, Xu, and Príncipe (2007). The authors investigate average state entropy and motivate a maximization of this value to obtain better-performing reservoirs. In contrast, we analyze the linear algebraic properties of a given weight matrix. We intend to adjust all eigenvalues and singular values. Furthermore, this letter is targeted at deeper insight into the internal dynamics of these reservoirs, at least in the linear case.

In the next section, we briefly introduce the background of ESNs. In section 3, we present methods to create weight matrices with desired properties. Those weight matrices should be equipped with predefined singular values and eigenvalues, easily verifiable stability, and a long short-term memory. In section 4, we analyze the impact of disturbances on ESNs and show that our new matrices are more robust against noise than randomly created ones. We have tested the new reservoirs on standard experiments with convincing success and describe the observations in section 5. There, we also investigate different ways of connecting the inputs to the reservoir.

2 Echo State Networks

Echo state networks (as well as most other neural networks) consist of neurons and links. Each neuron has a time-dependent activation. In Figure 1, we distinguish three kinds of neurons:

- Input units providing external activations $\boldsymbol{u} \in \mathbb{R}^K$
- Hidden units with internal activations $\boldsymbol{x} \in \mathbb{R}^N$
- Output units that generate the system's output signal $\boldsymbol{y} \in \mathbb{R}^L$

The connections between neurons are weighted. We collect these weights in weight matrices, which are usually denoted by $\boldsymbol{W}$ and a superscript referring to the location of the respective connections. The activations are calculated by

$$\boldsymbol{x}(n+1) = f(\boldsymbol{W}\boldsymbol{x}(n) + \boldsymbol{W}^{in}\boldsymbol{u}(n+1) + \boldsymbol{W}^{back}\boldsymbol{y}(n)), \tag{2.1}$$

$$\boldsymbol{y}(n+1) = f^{out}(\boldsymbol{W}^{out}(\boldsymbol{u}(n+1), \boldsymbol{x}(n+1), \boldsymbol{y}(n))). \tag{2.2}$$

The functions $f : \mathbb{R} \to \mathbb{R}$ and $f^{out} : \mathbb{R} \to \mathbb{R}$ are applied component-wise. Typically f is a sigmoid function (like tanh). We assume $f^{out} = \mathrm{id}$ throughout this letter. The hidden layer is called a dynamical reservoir (DR). Once initialized, the weight matrix $\boldsymbol{W}$ of the DR remains constant like most of the other weights (e.g., weights of the input $\boldsymbol{W}^{in}$ or feedback connections $\boldsymbol{W}^{back}$). Only the output weights $\boldsymbol{W}^{out}$ are trained, usually by a fast linear regression algorithm. The states and the targets are collected row-wisely in the matrices $\boldsymbol{S}$ and $\boldsymbol{T}$, respectively. Then (if $f^{out} = \mathrm{id}$), we have to solve

$$\boldsymbol{S}(\boldsymbol{W}^{out})^T = \boldsymbol{T}. \tag{2.3}$$

Definition 1 *(Jaeger, 2001a). Let $\boldsymbol{u}(n) \in U$ and $\boldsymbol{x}(n) \in A$ with compact spaces U and A. Assume that the network has no output feedback connections. Then the network has echo states if the network state $\boldsymbol{x}(n)$ is uniquely determined by any left-infinite input sequence $\overline{\boldsymbol{u}}^{-\infty}$. More precisely, this means that for every input sequence $(\ldots, \boldsymbol{u}(-1), \boldsymbol{u}(0)) \in U^{-\mathbb{N}}$, for all state sequences $(\ldots, \boldsymbol{x}(-1), \boldsymbol{x}(0))$ and $(\ldots, \boldsymbol{x}'(-1), \boldsymbol{x}'(0)) \in A^{-\mathbb{N}}$, where for any $i \in \mathbb{Z}$,*

$$\boldsymbol{x}(i) = f(\boldsymbol{W}^{in}\boldsymbol{u}(i) + \boldsymbol{W}\boldsymbol{x}(i-1)) \tag{2.4}$$

and

$$\boldsymbol{x}'(i) = f(\boldsymbol{W}^{in}\boldsymbol{u}(i) + \boldsymbol{W}\boldsymbol{x}'(i-1)), \tag{2.5}$$

it holds that $\boldsymbol{x}(n) = \boldsymbol{x}'(n)$ for any n.

Theorem 1 *(Jaeger, 2001a). Assume $f = \tanh$ and a network without output feedback:*

a. *Let the weight matrix $\boldsymbol{W}$ satisfy $\sigma_{max} = \Lambda < 1$, where σ_{max} is its largest singular value. Then $\|\boldsymbol{x}(n+1) - \boldsymbol{x}'(n+1)\|_2 < \Lambda\|\boldsymbol{x}(n) - \boldsymbol{x}'(n)\|_2$ for all inputs $\boldsymbol{u}(n+1)$, for all states $\boldsymbol{x}(n), \boldsymbol{x}'(n) \in [-1, 1]^N$. This implies echo states for all inputs $\boldsymbol{u}(n+1)$ and for all states $\boldsymbol{x}(n), \boldsymbol{x}'(n) \in [-1, 1]^N$.*
b. *Let the weight matrix $\boldsymbol{W}$ have a spectral radius $\rho(\boldsymbol{W}) > 1$, where $\rho(\boldsymbol{W})$ is an eigenvalue of $\boldsymbol{W}$ with the largest absolute value. Then the network has*

an asymptotically unstable null state. This implies that it has no echo states for any input set U containing $\mathbf{0}$ and admissible state set $A = [-1, 1]^N$.

See Jaeger (2001a) for the proof. Obviously, theorem 1 provides a necessary and a sufficient condition for the echo state property.

To avoid a necessary but trivial case-by-case analysis, we assume both one-dimensional input and output (i.e., $L = K = 1$). Therefore, some matrices become vectors, which we denote by lowercase bold letters (i.e., $\boldsymbol{w}^{in}$ instead of $\boldsymbol{W}^{in}$ or $\boldsymbol{w}^{out}$ instead of $\boldsymbol{W}^{out}$). The higher-dimensional case can be treated in almost the same way. We also assume that $y(n+1)$ does not depend on $y(n)$ or $u(n+1)$. Thus, instead of the more general form, equation 2.2, we use

$$y(n+1) = f(\boldsymbol{w}^{out}\boldsymbol{x}(n+1)). \tag{2.6}$$

We usually refer to the linear case, $f = \mathrm{id}$, to allow some analysis. Experiments show that improvements derived from the linear case usually also work quite well in the nonlinear case (e.g. $f = \tanh$).

3 Construction

As we mentioned in section 1, the dynamical reservoir is typically initialized randomly and stays untrained. Jaeger (2002) suggested the following procedure:

1. Randomly generate an internal weight matrix $\boldsymbol{W}_0$.
2. Normalize $\boldsymbol{W}_0$ to a matrix $\boldsymbol{W}_1$ with unit spectral radius by putting $\boldsymbol{W}_1 = (1/\rho(\boldsymbol{W}_0))\,\boldsymbol{W}_0$, where $\rho(\boldsymbol{W}_0)$ is the spectral radius of $\boldsymbol{W}_0$.
3. Scale $\boldsymbol{W}_1$ to $\boldsymbol{W} = \alpha\boldsymbol{W}_1$, where $\alpha < 1$, such that finally $\boldsymbol{W}$ has a spectral radius of α.

This procedure does not state anything about the singular values. Therefore, the echo state property is not ensured. Naturally the question arises about how to construct stable reservoirs. Furthermore, only the absolute value of the maximal eigenvalue is known, which leads to the question about a design for reservoirs with user-defined properties. One approach to answer these questions was developed in Ozturk et al. (2007). The authors suggested two different methods, but their second one is out of our focus since it is a bias learning method. The first one concerns steps 1 and 2 of Jaeger's reservoir construction. Instead of choosing a random matrix $\boldsymbol{W}_0$, Ozturk et al. generate a matrix with predefined eigenvalue spectrum

using the rational canonical form:

$$\boldsymbol{W}_0 := \begin{pmatrix} -a_1 & -a_2 & \cdots & -a_{N-1} & -a_N \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & 0 \\ 0 & \cdots & 0 & 1 & 0 \end{pmatrix}. \tag{3.1}$$

Due to the particular structure of $\boldsymbol{W}_0$, the characteristic polynomial of this matrix has the coefficients contained in the first row of $-\boldsymbol{W}_0$:

$$p(\lambda) = \det(\lambda \boldsymbol{I} - \boldsymbol{W}_0) = \lambda^N + a_1\lambda^{N-1} + a_2\lambda^{N-2} + \cdots + a_N\lambda^0. \tag{3.2}$$

Thus, all eigenvalues (and especially the spectral radius) are determined by the first row of $\boldsymbol{W}_0$. Consequently, a normalization to spectral radius 1 is unnecessary if the a_i are chosen properly. The first row of $\boldsymbol{W}_0$ is calculated by a maximization of the average state entropy through a uniform eigenvalue distribution in the complex plane. A neural network equipped with such a reservoir weight matrix is called ASE-ESN.

Our approach is similar. Instead of Jaeger's steps 1 and 2, we specify classes of matrices with spectral radius 1 from which $\boldsymbol{W}_0$ is taken.

3.1 Sparse and Orthogonal Reservoir Matrices. The echo state property guarantees that the outputs of the neural net are basically determined by the sequence of inputs. Due to theorem 1, this property is fulfilled if the singular values are smaller than 1 and the activation function f is Lipschitz continuous with Lipschitz constant 1. To calculate the singular values of any matrix $\boldsymbol{M} \in \mathbb{R}^{n\times n}$, we can use its singular value decomposition (SVD). There exist unitary matrices $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{C}^{n\times n}$ such that $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$, where $\boldsymbol{\Sigma}$ is a diagonal matrix containing the singular values on the main diagonal and $*$ denotes the Hermitian transpose. If we start with any diagonal matrix $\boldsymbol{D}$ and multiply it by any orthogonal matrices from the left the right, the singular values remain unchanged. This means that $\boldsymbol{D}$ and $\boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$ have the same singular values because $\boldsymbol{D}^T\boldsymbol{D}$ and $\boldsymbol{V}\boldsymbol{D}^T\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T = \boldsymbol{V}\boldsymbol{D}^T\boldsymbol{D}\boldsymbol{V}^T$ are similar and hence have the same eigenvalues.

The outline is now as follows. We start with an arbitrary diagonal matrix with maximal singular value 1 and obtain $\boldsymbol{W}_0$ by consecutive multiplication by orthogonal matrices. Then $\boldsymbol{W}_0$ also has a maximal singular value of 1 and the echo state property is satisfied if $0 \le \lambda < 1$ where $\boldsymbol{W} = \lambda \boldsymbol{W}_0$, as in step 3.

Furthermore, we propose choosing all singular values equal to 1. In this case, the absolute values of all eigenvalues also equal 1. Since the identity matrix $\boldsymbol{I}$ satisfies all desired properties, we may start with $\boldsymbol{D} = \boldsymbol{I}$. We first multiply $\boldsymbol{I}$ by some random permutation matrix $\boldsymbol{P}$ to shift nonzero entries away from the main diagonal. We do this in order to increase the dynamics in the neural network and force the neurons to interact with each other instead of just talking to themselves. Note that $\boldsymbol{P}$ is orthogonal; the desired spectral properties remain true.

So far, every neuron has exactly one input and one output connection within the reservoir. To generate more connections, we multiply $\boldsymbol{PI} = \boldsymbol{P}$ by other orthogonal matrices $\boldsymbol{U}$ and $\boldsymbol{V}$. It is computationally important to use sparse weight matrices $\boldsymbol{W}$. Dense matrices consume many floating-point operations that slow down the algorithm during the working phase. Unfortunately, random $\boldsymbol{U}$ and $\boldsymbol{V}$ typically generate dense matrices $\boldsymbol{W}_0 = \boldsymbol{UPV}^T$. To avoid too many nonzero entries, we start with $\boldsymbol{W}_1 = \boldsymbol{P}$ and multiply $\boldsymbol{W}_{i-1}$ step-by-step by sparse orthogonal matrices $\boldsymbol{Q}_i$ to generate only a few nonzero entries, that is, $\boldsymbol{W}_i = \boldsymbol{Q}_i\boldsymbol{W}_{i-1}$ or $\boldsymbol{W}_i = \boldsymbol{W}_{i-1}\boldsymbol{Q}_i$. A set of sparse matrices is, for example, the set of two-dimensional (Givens) rotation matrices $\boldsymbol{Q}(h, k, \varphi)$ defined by

$$\boldsymbol{Q}(h,k,\varphi) := \begin{pmatrix} \ddots & & & & & & \\ & 1 & & & & & \\ & & \cos(\varphi) & & -\sin(\varphi) & & \leftarrow h\text{th row} \\ & & & 1 & & & \\ & & & & \ddots & & \\ & & & & & 1 & \\ & & \sin(\varphi) & & \cos(\varphi) & & \leftarrow k\text{th row} \\ & & & & & 1 & \\ & & & & & & \ddots \end{pmatrix}. \tag{3.3}$$

We stop this procedure if $\boldsymbol{W}_i$ reaches a predefined density and obtain

$$\begin{aligned} \boldsymbol{W}_{r+l+1} = {} & \boldsymbol{Q}\left(h_{i_l}, k_{i_l}, \varphi_{i_l}\right) \cdot \cdots \cdot \boldsymbol{Q}\left(h_{i_1}, k_{i_1}, \varphi_{i_1}\right) \\ & \times \boldsymbol{PQ}\left(h_{j_1}, k_{j_1}, \varphi_{j_1}\right) \cdot \cdots \cdot \boldsymbol{Q}\left(h_{j_r}, k_{j_r}, \varphi_{j_r}\right). \end{aligned} \tag{3.4}$$

Now we set $\boldsymbol{W}_0 = \boldsymbol{W}_{r+l+1}$ and continue with step 3 of Jaeger's procedure.

Thus, we obtain sparse and orthogonal reservoir matrices which we refer to as SORM in the following (algorithm 1). All singular values and the absolute values of the eigenvalues of the SORM are known since they are determined by the construction. At least in the linear case, the damping of our RNN is constant and the stability (echo state property) is easily verified.

Algorithm 1: SORM

1. Permute the rows of I to avoid small cycles. Then all singular values as well as the absolute values of the eigenvalues of the resulting matrix are 1.
2. Choose a rotation matrix $\boldsymbol{Q}(h, k, \varphi)$ of type 3.3 and multiply it from the left or from the right by the current weight matrix.
3. Repeat step 2 until the desired target density is reached.

3.2 CyclicSORMs. We assume only one input neuron, that is, the input weight matrix $\boldsymbol{w}^{in}$ is a vector. In section 4, we will show that the matrix

$$\boldsymbol{M} := (\boldsymbol{w}^{in} | \boldsymbol{W}\boldsymbol{w}^{in} | \dots | \boldsymbol{W}^{N-1}\boldsymbol{w}^{in}) \tag{3.5}$$

has a direct influence on the impact of errors in the following sense. If the condition number of $\boldsymbol{M}$ is very high or the rank of $\boldsymbol{M}$ is low, then the neural net is susceptible to noise (in the linear case). This means that the columns of $\boldsymbol{M}$ should be linearly independent to guarantee a full rank and orthogonal to guarantee a small condition number. A second motivation for a high rank is given by Jaeger (2001b). He proved that short-term memory is maximum if $\boldsymbol{WM}$ is regular, which implies that $\boldsymbol{M}$ is regular. This means $\boldsymbol{M}$ is directly related to the network's ability to remember past inputs.

Note that $\boldsymbol{M} = (\boldsymbol{w}^{in} | \boldsymbol{W}\boldsymbol{w}^{in} | \dots | \boldsymbol{W}^{N-1}\boldsymbol{w}^{in}) = (\boldsymbol{w}^{in} | \lambda \boldsymbol{W}_0 \boldsymbol{w}^{in} | \dots | \lambda^{N-1} \boldsymbol{W}_0^{N-1} \boldsymbol{w}^{in})$, and thus, $\boldsymbol{M}$ can never be orthogonal for $\lambda \neq 1$ and orthogonal $\boldsymbol{W}_0$. Although the columns are mutually orthogonal, they do not have unit length. Therefore, we aim to construct a set of matrices $\boldsymbol{W}_0$ whose elements generate at least an orthogonal matrix $\boldsymbol{M}_0 := (\boldsymbol{w}^{in} | \boldsymbol{W}_0 \boldsymbol{w}^{in} | \dots | \boldsymbol{W}_0^{N-1} \boldsymbol{w}^{in})$. The rank of $\boldsymbol{M}$ equals rank $(\boldsymbol{M}_0)$ and the condition number of $\boldsymbol{M}_0$, $\mathrm{cond}(\boldsymbol{M}_0)$, is $\frac{1}{\lambda^{N-1}} \mathrm{cond}(\boldsymbol{M})$. Since $\boldsymbol{w}^{in}$ is part of $\boldsymbol{M}_0$, our investigations include also these input weights.

The construction of section 3.1 yields $\boldsymbol{W}_0 = \boldsymbol{U}\boldsymbol{P}\boldsymbol{V}^T$ (whereas $\boldsymbol{U} = \boldsymbol{Q}(h_{i_l}, k_{i_l}, \varphi_{i_l}) \dots \boldsymbol{Q}(h_{i_1}, k_{i_1}, \varphi_{i_1})$ and $\boldsymbol{V}^T = \boldsymbol{Q}(h_{j_1}, k_{j_1}, \varphi_{j_1}) \cdots \boldsymbol{Q}(h_{j_r}, k_{j_r}, \varphi_{j_r})$). We denote the ith column of the identity matrix by $\boldsymbol{e}_i$.

Lemma 1. *If $\boldsymbol{U} = \boldsymbol{V}$, $\boldsymbol{w}^{in} = \boldsymbol{V}\boldsymbol{e}_1$ and the permutation related to $\boldsymbol{P}$ is of cycle length N, then $\boldsymbol{M}_0$ is orthogonal.*

Proof. Assume $\boldsymbol{U} := \boldsymbol{V}$. The resulting matrix $\boldsymbol{W}_0 = \boldsymbol{V}\boldsymbol{P}\boldsymbol{V}^T$ is similar to the permutation matrix $\boldsymbol{P}$ (i.e., both matrices have the same eigenvalues). Now $\boldsymbol{W}_0^k \boldsymbol{w}^{in}$ is orthogonal to $\boldsymbol{W}_0^j \boldsymbol{w}^{in}$ for $k \neq j \in \{0, \dots, N-1\}$ if and only if $(\boldsymbol{W}_0^k \boldsymbol{w}^{in})^T \boldsymbol{W}_0^j \boldsymbol{w}^{in} = 0$. Since $\boldsymbol{w}^{in}$ is the first column of the orthogonal matrix $\boldsymbol{V}$,

$$\begin{aligned}(\boldsymbol{W}_0^k \boldsymbol{w}^{in})^T \boldsymbol{W}_0^j \boldsymbol{w}^{in} &= \boldsymbol{e}_1^T \boldsymbol{V}^T (\boldsymbol{V}\boldsymbol{P}^T\boldsymbol{V}^T)^k (\boldsymbol{V}\boldsymbol{P}\boldsymbol{V}^T)^j \boldsymbol{V}\boldsymbol{e}_1 \\ &= \boldsymbol{e}_1^T (\boldsymbol{P}^k)^T \boldsymbol{P}^j \boldsymbol{e}_1 = \boldsymbol{e}_1^T \boldsymbol{P}^{j-k} \boldsymbol{e}_1. \end{aligned} \tag{3.6}$$

Algorithm 2: CyclicSORM

1. Choose a permutation π of cycle length N. The related matrix is denoted by $\boldsymbol{P}$.
2. Choose a sparse and orthogonal transformation matrix $\boldsymbol{V}$ as in section 3.1.
3. Set $\boldsymbol{W}_0 := \boldsymbol{V}\boldsymbol{P}\boldsymbol{V}^T$.

Let us denote the permutation corresponding to $\boldsymbol{P}$ by π. Then $\boldsymbol{P}\boldsymbol{e}_1 = \boldsymbol{e}_{\pi(1)}$. Therefore, $\boldsymbol{e}_1^T \boldsymbol{P}^{j-k} \boldsymbol{e}_1 = 0$ is satisfied for any $k \neq j \in \{0, \ldots, N-1\}$ iff π is a cycle of length N (i.e., $\boldsymbol{P}^i \boldsymbol{e}_1 \neq \boldsymbol{e}_1$ for any $i \in \{1, \ldots, N-1\}$). Consequently, $\boldsymbol{M}_0$ is orthogonal for this special choice of $\boldsymbol{W}_0$ and $\boldsymbol{w}^{in}$ iff the cycle length of $\boldsymbol{P}$ is N.

The properties of $\boldsymbol{W}_0 = \boldsymbol{V}\boldsymbol{P}\boldsymbol{V}^T$ are well known. For example, the weight matrix $\boldsymbol{W}_0$ is similar to $\boldsymbol{P}$, that is, the characteristic polynomial, $p(x)$, equals that of $\boldsymbol{P}$, $p(x) = x^N - 1$. (To show this, use the Laplace expansion to calculate $\det(x\boldsymbol{I} - \boldsymbol{P})$.) This means that the eigenvalues are uniformly distributed on the complex unit circle.

Afterward we again scale the spectral radius to the desired value. The procedure yields sparse and orthogonal matrices as in section 3.1. Additionally, the columns of the above matrices go through the orthogonal subspaces in a cyclical manner if $\boldsymbol{w}^{in} = \boldsymbol{V}\boldsymbol{e}_1$. Under these conditions, the reservoir saves the last N inputs $u(n)$ in disjoint orthogonal subspaces of $\mathbb{R}^N$. Only inputs older than $N-1$ time steps do not have their own subspace and superpose newer inputs. The matrices provide a maximum short-term memory since they produce fewer superpositions of inputs than random matrices do. Mainly for the experimental section, we abbreviate the notation and call neural networks equipped with reservoirs based on the above procedure *CyclicSORMs* (see algorithm 2).

For all theoretical investigations, we assume a linear activation function f. We show later that under this assumption, $\boldsymbol{w}^{in} = \boldsymbol{V}\boldsymbol{e}_1$ yields a well-conditioned correlation matrix of the network states. Using a linear combination of different columns of $\boldsymbol{V}$ does not make sense because no additional dynamic is introduced. This linear combination can also be done within the readout layer, leading to the same outcome. The only (unwanted) effect is a reduction of memory capacity.

However, for nonlinear activation functions ($f = \tanh$), there could be (there are, as we will see later) additional dynamics if $\boldsymbol{w}^{in}$ is such a linear combination of different columns of $\boldsymbol{V}$. Therefore, inspired by the analysis for $\boldsymbol{w}^{in}$, we will use a weighted sum of different columns of $\boldsymbol{V}$ as input weights for the experiments.

3.3 RingOfNeurons and ChainOfNeurons. Taking a closer look at the CyclicSORMs, we discover an even simpler system that has the same properties in the linear case. Again, let us assume linear activation functions $f = \mathrm{id}$ and $f^{out} = \mathrm{id}$ for this section. Then the current activations can be

calculated as

$$\boldsymbol{x}(n) = \boldsymbol{W}\boldsymbol{x}(n-1) + \boldsymbol{w}^{in}u(n), \tag{3.7}$$

$$\boldsymbol{y}(n) = \boldsymbol{w}^{out}\boldsymbol{x}(n). \tag{3.8}$$

For $\boldsymbol{W}$ and $\boldsymbol{w}^{in}$ generated by the construction of section 3.2, we obtain

$$\boldsymbol{x}(n) = (\lambda \boldsymbol{V}\boldsymbol{P}\boldsymbol{V}^T)\boldsymbol{x}(n-1) + \boldsymbol{V}\boldsymbol{e}_1 u(n) \tag{3.9}$$

and

$$\boldsymbol{V}^T\boldsymbol{x}(n) = \lambda \boldsymbol{P}\boldsymbol{V}^T\boldsymbol{x}(n-1) + \boldsymbol{e}_1 u(n). \tag{3.10}$$

Considering $\boldsymbol{x}(n)$ and $\widehat{\boldsymbol{x}}(n) = \boldsymbol{V}^T\boldsymbol{x}(n)$ as states of the same but rotated dynamical system, we can also work with the simpler version $\widehat{\boldsymbol{x}}$ generated by the network update

$$\widehat{\boldsymbol{x}}(n) = \lambda \boldsymbol{P}\widehat{\boldsymbol{x}}(n-1) + \boldsymbol{e}_1 u(n), \tag{3.11}$$

with the very simple reservoir matrix $\boldsymbol{P}$ together with input weights $\boldsymbol{e}_1$. The system produces the same outputs $y(n)$ as before if $y(n) = \boldsymbol{w}^{out}\boldsymbol{x}(n) = \boldsymbol{w}^{out}\boldsymbol{V}\widehat{\boldsymbol{x}}(n)$, that is, with appropriately modified output weights $\widehat{\boldsymbol{w}}^{out} = \boldsymbol{w}^{out}\boldsymbol{V}$.

According to the permutation matrix $\boldsymbol{P}$, the internal units of the simplified network are connected in a cyclic way. The units can be relabeled such that without loss of generality, the reservoir matrix is

$$\widehat{\boldsymbol{W}} := \lambda \begin{pmatrix} 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & 0 \\ 0 & \dots & 0 & 1 & 0 \end{pmatrix}. \tag{3.12}$$

The neural net with this matrix $\widehat{\boldsymbol{W}}$ is called *RingOfNeurons*.

Assume a RingOfNeurons together with $\widehat{\boldsymbol{w}}^{in} = \boldsymbol{e}_1$ and $\lambda \neq 0$. Then $x_1(n) = \sum_{i=0}^{\infty} \lambda^{iN} u(n-iN)$, while all other activations $x_j(n)$ are independent of $u(n-iN)$ for all $i \in \mathbb{N}$. This means that neither $u(n)$ nor former inputs can be reproduced without error except for the trivial case that $u(n) = \alpha u(n-N)$ for some α because the activation $x_1(n)$ is a superposition of the inputs $u(n-iN)$ ($i \in \mathbb{N}$). We would like to avoid this superposition by forbidding the connection from the last internal neuron to the first one

($\widehat{w}_{1,N} = 0$). We get

$$\widehat{W} := \lambda \begin{pmatrix} 0 & 0 & \dots & 0 & 0 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & 0 \\ 0 & \dots & 0 & 1 & 0 \end{pmatrix}. \tag{3.13}$$

The above matrix $\widehat{W}$ is nilpotent and has only zero eigenvalues. Such a dynamical reservoir acts as a FIFO memory. After N steps, the system forgets the input. We refer to this reservoir as a ChainOfNeurons.

Because of the same motivation as at the end of section 3.2, for practical experiments, we use linear combinations of the columns of the identity matrix $\boldsymbol{I}$ as input weights in section 5.

3.4 Related Publications. The idea of using orthogonal matrices was reported in Jaeger (2001b). However, the construction of the matrix was different and without control of the matrix density. Jaeger showed experimentally that the short-term memory of reservoirs equipped with orthogonal matrices is almost maximum.

A theoretical analysis of the short-term memory of related reservoir matrices in linear networks was done by White, Lee, and Sompolinsky (2004). They investigated the memory of random orthogonal matrices and distributed shift register networks. The latter are strongly related to the CyclicSORM, the RingOfNeurons, and the ChainOfNeurons. In fact, the ChainOfNeurons is a distributed shift register network. The difference from the other kinds of reservoir weights is that a distributed shift register depends on only the last N inputs. The authors give precise formulas to calculate the memory capacity of these networks.

Reservoirs similar to the RingOfNeurons and the ChainOfNeurons were also introduced in Čerňanský and Tiňo (2008) and Rodan and Tiňo (2011), where the major intent was to simplify network complexity. The authors concluded that for some tasks, the above reservoirs work as well as random reservoirs. Čerňanský and Tiňo (2008) investigate full input connections as well as $\boldsymbol{w}^{in} = \alpha \boldsymbol{e}_1$. Rodan and Tiňo (2011) analyze an even more simplified connection method where all absolute values of the input weights are constant and only the sign varies. Here, the authors prove that under certain conditions, the RingOfNeurons can achieve a memory capacity arbitrarily close to the theoretical bound N.

4 Estimation of Errors

The states of the dynamical reservoir are typically noisy. Sources of noise are, for example, perturbed input data, user-forced noise to avoid overfitting or to regularize the training equation (ridge regression). But noisy activations lead to perturbations of the training, to incorrect output weights, and, finally, to a larger output error. One would expect that the error strongly depends on the rank and the singular values of the matrix $\boldsymbol{M} := (\boldsymbol{w}^{in}|\boldsymbol{W}\boldsymbol{w}^{in}|\dots|\boldsymbol{W}^{N-1}\boldsymbol{w}^{in})$. This is exactly what we will show. Furthermore, we will prove that for linear activation functions, the ChainOfNeurons' matrix $\boldsymbol{M}$ has optimal singular values. Randomly created weight matrices usually yield ill-conditioned or even singular $\boldsymbol{M}$ and thus are more susceptible to noise.

We store all reservoir states row-by-row in the matrix $\widehat{\boldsymbol{S}}$. Analogously, the targets are saved in $\boldsymbol{t}$. Assuming the linear case, Jaeger (2001b) proved that memory capacity is bounded by N and that MC_k (a measure for the memory of the input k time steps before) is decreasing monotonically. This was shown experimentally to be valid also in the nonlinear case (see section 5.1 or Ozturk et al., 2007). Consequently, if one wants to reproduce the input $u(n-N)$ entered N time steps before, this implies two facts: since MC is bounded by N, $u(n-k)$ $(k=0,\dots,N-1)$ is regenerated with errors; and since MC_k is monotonically decreasing, the ESN cannot reproduce $u(n-N)$ without error. Therefore, we assume that the exact state $\boldsymbol{x}(n)$ depends on only the last N inputs to guarantee an optimal, noiseless linear dependence of $\boldsymbol{y}(n)$ on the previous N inputs. The exact matrix of the reservoir activations is denoted by $\boldsymbol{S}$, and $\boldsymbol{S}^+$ is its pseudoinverse. We define the perturbation $\boldsymbol{\Delta S}$ by $\widehat{\boldsymbol{S}} = \boldsymbol{S} + \boldsymbol{\Delta S}$. Analogously, let $\widehat{\boldsymbol{w}}^{out}$ be the solution of

$$\|\widehat{\boldsymbol{S}}(\widehat{\boldsymbol{w}}^{out})^T - \boldsymbol{t}\| \to \min\,, \tag{4.1}$$

whereas $\boldsymbol{w}^{out} = \boldsymbol{t}^T(\boldsymbol{S}^+)^T$ is the exact output weight vector and $\boldsymbol{\Delta w}^{out}$ denotes the difference $\widehat{\boldsymbol{w}}^{out} - \boldsymbol{w}^{out}$. Let $\boldsymbol{I}_p$ be the $p \times p$ identity matrix and assume for this section that $\|\cdot\|$ is the spectral norm. The proofs of the following theorems and lemmas are given in the appendix.

Theorem 2 *(first-order approximation of $\boldsymbol{\Delta w}^{out}$). For sufficiently small $\|\boldsymbol{\Delta S}\|$,*

$$\begin{aligned}\boldsymbol{\Delta w}^{out} \approx\, & \boldsymbol{t}^T[(\boldsymbol{S}+\boldsymbol{\Delta S})\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*(\boldsymbol{V}\boldsymbol{J}_l\boldsymbol{V}^*\boldsymbol{\Delta S}^T\boldsymbol{S})^+ + \boldsymbol{\Delta S}\boldsymbol{V}\boldsymbol{J}_l\boldsymbol{V}^*(\boldsymbol{S}^T\boldsymbol{\Delta S}\boldsymbol{V}\boldsymbol{J}_l\boldsymbol{V}^*)^+] + \\ & + \boldsymbol{r}^T\boldsymbol{\Delta S}(\boldsymbol{S}^T\boldsymbol{S})^+ - \boldsymbol{w}^{out}\boldsymbol{\Delta S}^T(\boldsymbol{S}^+)^T\end{aligned} \tag{4.2}$$

is a first-order approximation of the disturbance of $\boldsymbol{w}^{out}$*, where* $\boldsymbol{r} := \boldsymbol{t} - \boldsymbol{S}(\boldsymbol{w}^{out})^T$ *is the residuum and* $\boldsymbol{V}$ *is the orthogonal matrix from the SVD* $\boldsymbol{S} = \boldsymbol{U}\begin{pmatrix} \boldsymbol{\Sigma} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{pmatrix}\boldsymbol{V}^*$*. Let* $\boldsymbol{J}_l = \begin{pmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_{\bar{s}} \end{pmatrix}$ *and* $\boldsymbol{J}_u = \begin{pmatrix} \boldsymbol{I}_s & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{pmatrix}$ *be the diagonal matrices with* $\bar{s} = N - rank(\boldsymbol{S})$ *1 entries in the lower right and* $s = rank(\boldsymbol{S})$ *entries in the upper left corner, respectively.*

Lemma 2. *Let* $f = id$*. If* $rank(\boldsymbol{M}) = N$ *and at least N nonzero inputs are entered, then*

$$\boldsymbol{J}_l = \boldsymbol{0}. \tag{4.3}$$

In the following pages, we aim to estimate the ratio $\frac{\|\Delta \boldsymbol{w}^{out}\|}{\|\boldsymbol{w}^{out}\|}$ and minimize this approximation afterward to guarantee robust training. We estimate this ratio for small perturbations $\|\Delta \boldsymbol{S}\|$ using theorem 2:

$$\begin{aligned} &\frac{\|\Delta \boldsymbol{w}^{out}\|}{\|\boldsymbol{w}^{out}\|} \\ &\approx \frac{1}{\|\boldsymbol{w}^{out}\|}(\|\boldsymbol{r}^T \Delta \boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S})^+ - \boldsymbol{w}^{out}\Delta \boldsymbol{S}^T(\boldsymbol{S}^+)^T \\ &\quad + \boldsymbol{t}^T[(\boldsymbol{S}+\Delta \boldsymbol{S})\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*\left(\boldsymbol{V}\boldsymbol{J}_l\boldsymbol{V}^*\Delta \boldsymbol{S}^T\boldsymbol{S}\right)^+ + \Delta \boldsymbol{S}\boldsymbol{V}\boldsymbol{J}_l\boldsymbol{V}^*\left(\boldsymbol{S}^T\Delta \boldsymbol{S}\boldsymbol{V}\boldsymbol{J}_l\boldsymbol{V}^*\right)^+]\|) \end{aligned} \tag{4.4}$$

$$\begin{aligned} &\leq \frac{\|\boldsymbol{r}\|}{\|\boldsymbol{w}^{out}\boldsymbol{S}^T\|}\frac{\|\Delta \boldsymbol{S}\|}{\|\boldsymbol{S}\|}\|(\boldsymbol{S}^T\boldsymbol{S})^+\|\|\boldsymbol{S}\|^2 + \frac{\|\Delta \boldsymbol{S}\|}{\|\boldsymbol{S}\|}\|\boldsymbol{S}^+\|\|\boldsymbol{S}\| + \\ &\quad + \frac{\|\boldsymbol{t}^T[(\boldsymbol{S}+\Delta \boldsymbol{S})\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*\left(\boldsymbol{V}\boldsymbol{J}_l\boldsymbol{V}^*\Delta \boldsymbol{S}^T\boldsymbol{S}\right)^+ + \Delta \boldsymbol{S}\boldsymbol{V}\boldsymbol{J}_l\boldsymbol{V}^*\left(\boldsymbol{S}^T\Delta \boldsymbol{S}\boldsymbol{V}\boldsymbol{J}_l\boldsymbol{V}^*\right)^+]\|}{\|\boldsymbol{w}^{out}\|}. \end{aligned} \tag{4.5}$$

If $\text{rank}(\boldsymbol{M}) = N$ and $f = \text{id}$, the application of lemma 2 yields

$$\frac{\|\Delta \boldsymbol{w}^{out}\|}{\|\boldsymbol{w}^{out}\|} \lesssim \frac{\|\boldsymbol{r}\|}{\|\boldsymbol{w}^{out}\boldsymbol{S}^T\|}\frac{\|\Delta \boldsymbol{S}\|}{\|\boldsymbol{S}\|}\text{cond}(\boldsymbol{S})^2 + \frac{\|\Delta \boldsymbol{S}\|}{\|\boldsymbol{S}\|}\text{cond}(\boldsymbol{S}). \tag{4.6}$$

To decrease this bound, we decrease $\text{cond}(\boldsymbol{S})$ assuming that the targets can be well approximated by the network such that $\|\boldsymbol{r}\|/\|\boldsymbol{w}^{out}\boldsymbol{S}^T\|$ is small:

Lemma 3. *Let* $f = id$*,* θ *be the training size and the inputs* $u(i)$ *be independent realizations of the random variable u with zero expectation and variance* $\mathbb{V}(u)$*. Then for* $\theta \to \infty$*, the singular values of* $\boldsymbol{S}$ *asymptotically equal those of* $\sqrt{\theta\,\mathbb{V}(u)}\,\boldsymbol{M}$*. Consequently, the matrices* $\boldsymbol{S}$ *and* $\boldsymbol{M}$ *have asymptotically equal condition numbers.*

Remark 1. A crucial assumption of the lemma 3 is the identical and independent distribution of the inputs $u(n)$. Typically this is not satisfied in practical situations, but it is necessary for our linear algebraic investigation. However, in section 5.3, we also perform an experiment that does not satisfy this assumption, but the results remain comparable to experiments fulfilling it. Of course, there may be situations for which our assertions do not hold at all.

Due to lemma 3, we can minimize the condition number of $\boldsymbol{M}$ instead of $\boldsymbol{S}$ for sufficiently large training sizes. Obviously, $\mathrm{cond}(\boldsymbol{M}) = \frac{\sigma_{\max}(\boldsymbol{M})}{\sigma_{\min}(\boldsymbol{M})} \geq 1$. Here, the minimum 1 can be achieved by matrices $\boldsymbol{M}$ for which holds

$$\forall \boldsymbol{v} \in \mathbb{R}^N : \ \|\boldsymbol{M}\boldsymbol{v}\| = \rho(\boldsymbol{M})\|\boldsymbol{v}\|. \tag{4.7}$$

Below, we assume $\rho(\boldsymbol{M}) = 1$, keeping in mind that we also could scale $\boldsymbol{M}$ by any factor. Since equation 4.7, together with $\rho(\boldsymbol{M}) = 1$, holds for any $\boldsymbol{v} \in \mathbb{R}^N$, $\boldsymbol{M}$ is an orthogonal matrix. Multiplying $\boldsymbol{W}$ by $\boldsymbol{M}$ from the left and from the right yields a surprisingly simple matrix:

$$\begin{aligned} \boldsymbol{M}^T\boldsymbol{W}\boldsymbol{M} &= \boldsymbol{M}^T(\boldsymbol{W}\boldsymbol{w}^{in}|\boldsymbol{W}^2\boldsymbol{w}^{in}|\ldots|\boldsymbol{W}^N\boldsymbol{w}^{in}) \\ &= \begin{pmatrix} 0 & 0 & \ldots & 0 & * \\ 1 & 0 & \ldots & 0 & * \\ 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & * \\ 0 & \ldots & 0 & 1 & * \end{pmatrix} =: \boldsymbol{Q}. \end{aligned} \tag{4.8}$$

For later use, note that $(\boldsymbol{W}\boldsymbol{w}^{in}|\boldsymbol{W}^2\boldsymbol{w}^{in}|\ldots|\boldsymbol{W}^N\boldsymbol{w}^{in})$ contains $N-1$ columns of $\boldsymbol{M}$. The crucial consequence of equation 4.8 is that in order to get a regular matrix $\boldsymbol{M}$ with minimum condition number, $\boldsymbol{W}$ has to be an orthogonal transformation of a matrix of type $\boldsymbol{Q}$. Along the same line, the first column of the transformation matrix contains the appropriate, optimal input weights $\boldsymbol{w}^{in} = \boldsymbol{M}\boldsymbol{e}_1$.

This means that for any orthogonal matrix $\boldsymbol{V}$, $\boldsymbol{W} = \boldsymbol{V}\boldsymbol{Q}\boldsymbol{V}^T$ as above, and $\boldsymbol{w}^{in} = \boldsymbol{V}\boldsymbol{e}_1$, the rank of $\boldsymbol{M}$ is N and $\mathrm{cond}(\boldsymbol{M}) = \mathrm{cond}(\boldsymbol{S}) = 1$ provided that nonzero inputs are also entered in the network for at least N time steps. Hence, we decreased equation 4.6 in terms of $\mathrm{cond}(\boldsymbol{S})$.

Finally, let us take a closer look at $\boldsymbol{\Delta S}$. We assumed that the exact states depend on the previous N inputs, whereas earlier influences are considered to be part of $\boldsymbol{\Delta S}$. But the calculated states $\widehat{\boldsymbol{x}}$ strongly depend on even earlier inputs for those $\boldsymbol{Q}$ with a big norm in the last column. To see this, let $\boldsymbol{\delta}(n) = \widehat{\boldsymbol{x}}(n) - \sum_{i=0}^{N-1} \boldsymbol{W}^i\boldsymbol{e}_1 u(n-i) + \boldsymbol{W}^N\boldsymbol{x}(n-N)$ be the external noise. Then $\|\boldsymbol{\Delta x}(n)\| \leq \|\boldsymbol{\delta}(n)\| + \|\boldsymbol{W}^N\|$. A first step to minimize $\|\boldsymbol{\Delta x}(n)\|$ is to

decrease $\|\boldsymbol{W}^N\| = \|\boldsymbol{Q}^N\| \geq \|\boldsymbol{Q}\boldsymbol{e}_N\|$ since $\boldsymbol{Q}\boldsymbol{e}_N$ appears as the first column of $\boldsymbol{Q}^N$. A small norm is therefore preferable, and choosing the last column to be $\boldsymbol{e}_1$ appears to be reasonable. Note that this construction then corresponds to CyclicSORM.

If even $\boldsymbol{Q}\boldsymbol{e}_N = \boldsymbol{0}$ (and $\boldsymbol{V} = \boldsymbol{I}$), then we finally arrived at the ChainOfNeurons model. It can provide robust weights that yield a small error, which has been derived and shown to be optimal even theoretically at least for a linear activation function and fairly well-suited tasks (e.g., with $\|\boldsymbol{r}\| \leq \|\boldsymbol{S}\boldsymbol{w}^{out}\|$).

In this linear case, it is easy to see that including ChainOfNeurons with different weights does not provide any further improvement because the corresponding states are linearly dependent. However, this is different in the nonlinear case. If $\operatorname{rank}(\boldsymbol{S}) = N$, equation 4.6 holds also in the nonlinear case. Here, two or more separate chains (with different weights of course) may decrease $\boldsymbol{r}$. Due to the nonlinear activation function, the states are not linearly dependent, and thus the rank of $\boldsymbol{S}$ does not decrease. Apparently it has almost the same effect if, instead of injecting the input only to the first hidden neuron, we connect input and hidden layers sparsely with different weights; that is, in the nonlinear case, we apply input weights other than those derived in sections 3.2 and 3.3. Although our experiments seem to confirm this proposition, corresponding theoretical investigations remain to be done for nonlinear activation functions.

At least for the ChainOfNeurons with $\boldsymbol{w}^{in} = \alpha\boldsymbol{e}_1$, the above results can be transferred to the case of nonlinear activation functions ($f = \tanh$). Since theorem 2 is independent of the activation function, the first-order approximation also hold in the nonlinear case for any reservoir type (only $f^{out} = id$ is crucial). Also lemma 2 remains valid for nonlinear f using the ChainOfNeurons. Lemma 3 holds in a slightly modified version for nonlinear f and the ChainOfNeurons. Let $\rho(\boldsymbol{W})$ be the spectral radius of $\boldsymbol{W}$, and let $\varphi(\boldsymbol{x}) = (\tanh \circ \rho(\boldsymbol{W}))(\boldsymbol{x})$. Then lemma 3 holds also under the current assumptions when substituting $\boldsymbol{M}$ by

$$\boldsymbol{M}_{nl} := (\mathbb{E}[\tanh(\alpha u)]\boldsymbol{e}_1 | \mathbb{E}[(\varphi \circ \tanh)(\alpha u)]\boldsymbol{e}_2 | \ldots | \mathbb{E}[\varphi^{N-1} \circ \tanh(\alpha u)]\boldsymbol{e}_N).$$

This means that if $\mathbb{E}[\tanh(\alpha u)]/\mathbb{E}[\varphi^{N-1} \circ \tanh(\alpha u)]$ is small, the relative error $\|\Delta\boldsymbol{w}^{out}\|/\|\boldsymbol{w}^{out}\|$ again mainly depends on $\|\boldsymbol{r}\|/\|\boldsymbol{S}^T\boldsymbol{w}^{out}\|$.

5 Experiments

In this section, we compare the error rates of ESNs equipped with one of the following:

- StandardMat, the original randomly generated reservoirs
- SORM
- CyclicSORM

- RingOfNeurons
- ChainOfNeurons.

All results of the following experiments are reported in tables with the following abbreviations:

- ρ, spectral radius or constant connection weight along the ChainOfNeurons, respectively
- N, number of reservoir neurons
- *RDens*, connection density of the reservoir
- *FScale*, feedback scale: maximum absolute value for weights of output-hidden connections
- *FDens*, feedback density: portion of used output-hidden connections
- *IScale*, input scale: maximum absolute value for weights of input-hidden connections (see below)
- *IDens*, connection density from the input neurons into the reservoir; $\times$ means that only the first input unit or vector will be used

All experiments (except for the first one in section 5.1) were accomplished with the Oger (OrGanic Environment for Reservoir computing) toolbox and the ANN toolbox of PLANET intelligent systems GmbH. We optimize the parameter setup using the convenient grid search functionality of Oger. All of these optimizations are organized in the same way. We provide a number of training sets, including input and target data. Afterward, we take two samples from these data sets. The first is the training set, and the second is used for validation. The plots are generated by fixing all but one parameter of the resulting error arrays.

We test different input strategies. The first strategy is the classical random connection of input and hidden layer. We set a predefined percentage of the connections to uniformly distributed random values from $[-IScale, IScale]$. The second way of connecting input and hidden neurons is to distribute the input connections equally on the neurons; for example, if we use a density of $IDens = 0.25$, only neurons with index $\equiv 1 \pmod 4$ are connected with the input. The input weights of the CyclicSORM additionally are rotated afterward by the matrix $\boldsymbol{V}$ of algorithm 2. Again, the weight strengths are drawn from a uniform distribution over the interval $[-IScale, IScale]$.

The source code of the Oger experiments can be found online at http://www.math.uni-rostock.de/~tstrauss/downloads.html.

5.1 Delayline. The first experiment of this section is the well-known delayline experiment. The neural net gets random inputs (uniformly distributed over $[-0.5, 0.5]$), and the task is to reproduce those inputs after a given time delay at the output of the neural net. (We refer to Jaeger (2001a,

Table 1: Best Parameters for the Delayline Experiment Optimized by Hand.

	ρ	*IScale*	*IDens*
ChainOfNeurons	0.95	1.0	×
RingOfNeurons	0.95	1.0	×
CyclicSORM	0.95	1.0	×
SORM	0.95	1.0	0.1
StandardMat	0.99	1.0	0.1

2001b, 2002) for details.) We only used PLANET's ANN toolbox for this experiment.

We tested all of the considered matrices and improved the parameters roughly for every class of matrices separately by hand. The most important values are given in Table 1. The experiment was processed over 10,000 steps with a washout of 200 steps. We measured the mean squared error (MSE) over the last 100 steps of the experiment. The output activation function is the identity function, and the activation of the hidden units is tanh. We use (N=) 50 internal neurons. For each system, the results are averaged over 50 runs (50 different randomly generated systems and work flows). CyclicSORM, SORM, and StandardMat have a connection density of $RDens = 0.1$.

In Figure 2, the error is plotted against the delay interval. The slope of the dotted curve, which represents the results of SORM, is smoother than the one of the dashed-and-dotted curve achieved by StandardMat. At a delay interval of 30, the random reservoir already produces an output signal that is almost uncorrelated to the target signal. At this point, the SORM reservoir still yields good results. The curves of the CyclicSORM, ChainOfNeurons, and the RingOfNeurons show good short-term memory until a delay interval of $N - 1 = 49$.

In section 4, we proved the relation of $\boldsymbol{S}$ to perturbations. The greater the rank and the smaller the condition number of this matrix, the smaller is the impact of disturbances on the system's predictions in the linear case. In fact, the more precise bound substitutes $\text{cond}(\boldsymbol{S})$ by $\|\boldsymbol{S}\| \, \|\boldsymbol{S}^+\|$. Table 2 shows these values for the matrices used in this experiment. As expected, the ChainOfNeurons, the RingOfNeurons, and the CyclicSORM yield the best values. This coincides with the measured errors of the experiment where these three matrices also yield good performance.

The delayline experiment shows that the newly introduced weight matrices have good short-term memory. The CyclicSORM, the RingOfNeurons, and the ChainOfNeurons provide the best memory and work almost flawless until a delay of $N - 1$ time steps. Although the error of the SORM is soon greater than the error of these matrices, it is significantly lower than the error obtained by the randomly created matrices. This means that for

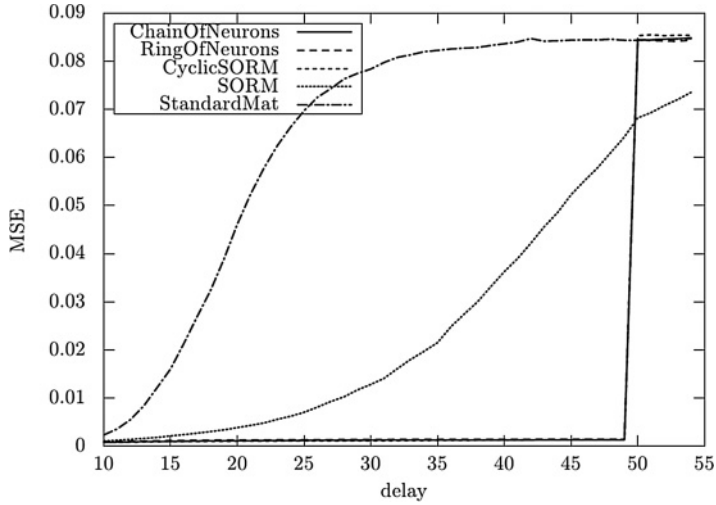

Figure 2: Result of the delayline experiment executed with different reservoir types. The mean squared error is plotted on the y-axis, and the delay times between the input of the signal and its reproduction by the neural net are plotted on the x-axis. Each point of the curve was averaged over 50 different initializations of the weight matrices. The ChainOfNeurons, the RingOfNeurons, and the CyclicSORM are almost identical since they generate only small errors.

Table 2: Averaged Rank of $S^T S$ and $\|S^T S\| \, \|(S^T S)^+\|$ on Average of 50 Runs.

	$\text{rank}(S^T S)$	$\|S^T S\| \, \|(S^T S)^+\|$
ChainOfNeurons	50.0	12.34648
RingOfNeurons	50.0	12.34648
CyclicSORM	50.0	12.34648
SORM	49.86	1584.37109
StandardMat	48.68	$2.53285 \cdot 10^{13}$

tasks requiring a high short-term memory, one should choose CyclicSORM, the RingOfNeurons, or the ChainOfNeurons. Interestingly, it seems that networks equipped with SORMs even have a (blurred) memory of what happened earlier than $N - 1$ time steps before.

5.2 Pattern Detection. The delayline experiment is a simple recognition task to test the short-term memory of the neural net. As shown, the introduced matrices provide better short-term memory. This next experiment shows that the newly introduced matrices are also capable of distinguishing (with a high accuracy) between a known pattern (six consecutive values

Table 3: Best Parameter Setup for the Pattern Detection Experiment Found by Oger Gridsearch.

	bias	ρ	*IScale*	*IDens*
ChainOfNeurons	0.0	0.1	0.8	1/7
RingOfNeurons	0.0	0.1	1.0	1/9
CyclicSORM	0.0	0.15	0.9	1/9
SORM	0.0	0.1	1.0	1.0
StandardMat	0.0	0.05	1.0	1.0

$p_1, \ldots, p_6 \in [-0.5, 0.5]$) and a random sequence. The input is a random sequence merged with this pattern. The intervals between two pattern are of random length. The task is to identify this pattern within the random sequence (for details, see Jaeger, 2001b). This is obviously not a short-term memory test since the important inputs are entered within only the last few steps.

The data sets contain 10,000 elements plus 200 additional washout data points. We use one data set for training and another for validation. The neural net contains one input, one output, and 50 hidden neurons. The grid of the search is four-dimensional, containing the bias scaling value of the reservoir, the input connection density, and the scaling values for the input and the reservoir weights. The best values found by this search are contained in Table 3. We say that the network has detected a pattern if the output neuron has an activation greater than a certain threshold. This threshold is set optimally and separately for each matrix by minimizing the sum of false positives and false negatives after the experiment. For any parameter setup, the experiments are executed with 10 different reservoirs and 6 different data sets each.

Use of a linear activation function $f = \mathrm{id}$ increases the error rates because of the following important feature of the nonlinearity: Assume a pattern of only one number q (not 0), and assume further two hidden neurons that are not connected. Their activation in each step is defined as $\tanh(w_1^{in}x)$ and $\tanh(w_2^{in}x)$ if x is the current input. The output activation y (a linear combination of the activations of the hidden neurons $y = \alpha \tanh(w_1 x) + \beta \tanh(w_2 x)$) should increase if $x = q$ and decrease else. Figure 3 shows the activation of the output neuron for fixed $\boldsymbol{w}^{out}$, w_1^{in}, and w_2^{in}. There is a maximum within the input interval that can be adjusted by changing w_1^{in} and w_2^{in} and learning $\boldsymbol{w}^{out}$ such that the maximum is near q; that is, saving the input within the reservoir several times with different input scalings improves detection of the pattern. This is the reason that the ChainOfNeurons can detect values far away from the boundary. To support this feature, we also use the second input strategy (distribute the input weights equally), which we already mentioned.

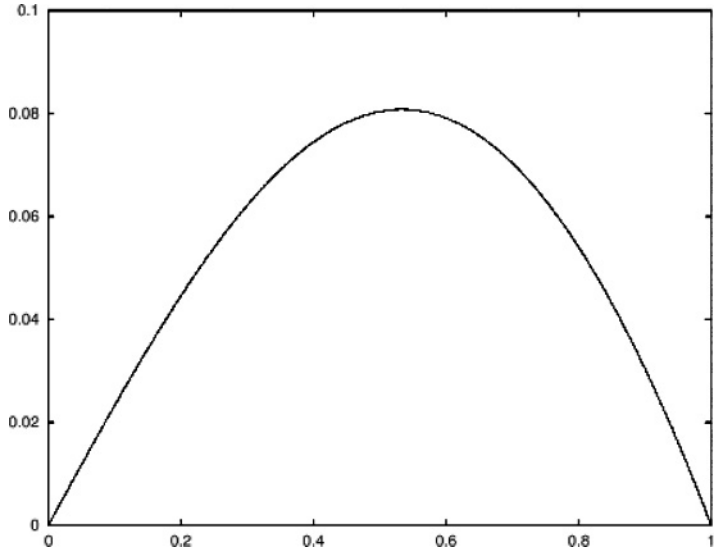

Figure 3: Nonlinearity feature. Output $y = \alpha \cdot \tanh(w_1^{in} \cdot x) - \beta \cdot \tanh(w_2^{in} \cdot x)$ for $x \in [0, 1]$ where $\alpha = 1$, $\beta = \tanh(1)/\tanh(0.1)$, $w_1^{in} = 1$ and $w_2^{in} = 0.1$. The maximum is between 0.5 and 0.6.

Figure 4 shows the number of false detections against the spectral radius. We take the average over 100 different initializations. The minimal number of false detections using StandardMat (random reservoirs) is clearly greater than the minimal number of those neural nets equipped with SORMs. Again, we obtain the best results by using RingOfNeurons or CyclicSORM. They provide similar behavior, which is not surprising since in the linear case, the mappings are identical. We just rotate the coordinate system. Changing the input strategy does not change the error of the networks with randomly generated DR weights (6.7 or 8.8, respectively, false detections minimum using random or equally distributed input weights) and SORM (8.0 or 7.3, respectively, false detections minimum using random or equally distributed input weights) much. On the other side, the RingOfNeurons and the ChainOfNeurons produce similar error rates in the case of random inputs. But if the inputs are equally distributed, the error rates decrease to only one false detection on average for the best setup. In this case, the variance (shown as error bars in Figure 4) is also small in relation to the other weight matrices.

Figure 5 shows the number of correctly detected patterns for different spectral radii. The different matrix types appear to behave very similarly. For a small spectral radius, no reservoir type is able to detect the pattern reliably. For spectral radii greater than 0.3, the curves stabilize at an optimum of about 220.

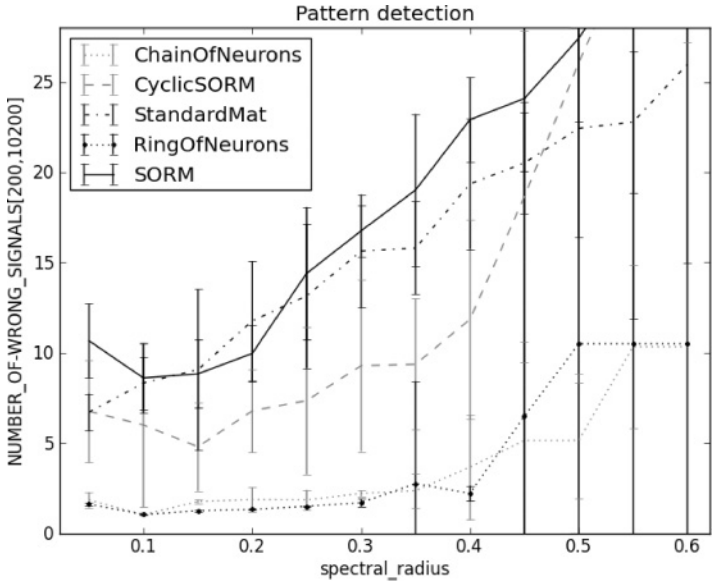

Figure 4: Pattern detection experiment. Error rates (false positives and false negatives) averaged over 60 different initializations for 10,000 time steps after 200 steps washout. The error bars denote the variance. The spectral radius of the reservoirs weight matrix is plotted on the x-axis.

At this point, we note an observation we made during this experiment. We tested with different bias scales and interpreted a bias as input (expectation value) shift, which possibly increases $\mathrm{cond}(S^{\mathrm{T}}S)$. We obtained the best results with no bias. This supports one assumption for the estimates of sections 4; $\mathbb{E}(u) = 0$, which is (almost) satisfied with no bias.

5.3 Mackey-Glass. The Mackey-Glass experiment is a prediction task, where the system has to do a one-step-ahead prediction of the Mackey-Glass time series (a nonlinear chaotic time series; see Jaeger (2001a) for details and parameters). In the "free run" mode, the system will perform several of these prediction steps and therefore produce a given number of steps ahead into the future; in other words: it will "generate" the time series. For our experiments, the Mackey-Glass time series is generated with $\tau = 17$, the traditional value for cautiously chaotic behavior.

The system is designed with 400 reservoir neurons connected as before by one of the corresponding connection matrices and one output neuron, aiming to predict the next step of the time series. The system's input is organized by feeding back the output of the system: the single output

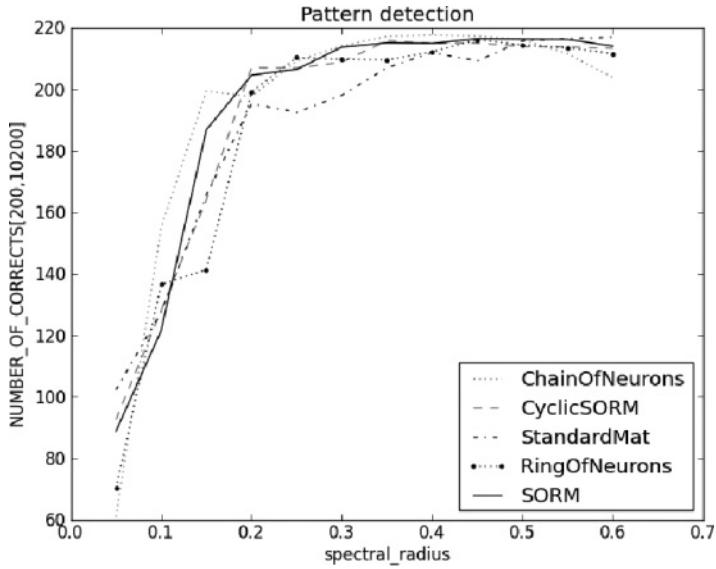

Figure 5: Pattern detection experiment. Correctly detected patterns averaged over 60 different initializations for 10,000 time steps after 200 steps washout. The spectral radius of the reservoir's weight matrix is plotted on the x-axis.

neuron is connected back into the reservoir by feedback connections. These feedback connections are treated like input connections.

Except for the "free run" mode, the output value of the system (activation of the output neuron) is set to the sequence value (teacher forcing mode for washout and training). We do not touch the system in the free run mode. The size of the training and test data sets is 5000, including 500 steps washout. The free run mode lasts 200 steps. As typically done for the Mackey-Glass experiment, we also measure the NRMSE over 12 different matrix initializations with 12 different runs for each parameter setup. The search includes the bias scaling factor, the number of input connections, and scaling factors of the input and the reservoir. Both the random and the equidistant input connection modes are tested. For this experiment, we use leaky integrator neurons within the reservoir, which after the update according to equation 2.4, additionally perform the following second step:

$$\boldsymbol{x}'(n+1) = (1-\delta)\boldsymbol{x}'(n+1) + \delta\boldsymbol{x}(n+1).$$

The value $\boldsymbol{x}'(n)$ is used as a neuron activation instead of $\boldsymbol{x}(n)$ at time n. We use constant $\delta = 0.4$ as the Oger samples suggest. Table 4 shows other parameters for obtaining a minimum error rate.

Table 4: Best Parameters for the Mackey-Glass Experiment (Found by Oger Gridseach).

	bias	ρ	*FScale*	*FDens*
ChainOfNeurons	0.15	0.85	1.1	1.0
RingOfNeurons	0.05	0.95	1.1	1.0
CyclicSORM	0.2	1.0	1.2	1.0
SORM	0.2	1.05	1.2	1.0
StandardMat	0.2	1.1	0.9	1.0

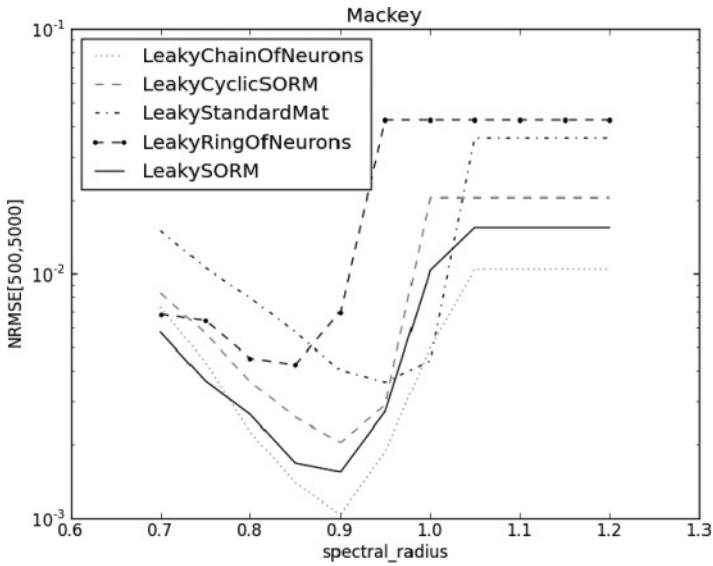

Figure 6: NRMSE of the Mackey-Glass experiment executed with different reservoir types and averaged over 144 different initializations. The network's task is to predict the next 200 time steps. Afterward, the output activation is reset to the target value to enable the prediction of the next 200 steps without error accumulation. The washout time lasts 500 steps. The spectral radius of the reservoir matrix is plotted on the *x*-axis.

Figure 6 shows the experimental results. Since the variances are very small (at least until the spectral radius reaches the minimum), we omit them in the plot. Again, the standard ESNs do not perform as well as the other matrices. Interestingly, a complete connection of output and hidden layer yields the best performance for all reservoir types. A sparse but equal distribution did not lead to an improvement.

Table 5: Best Parameters for the NARMA Experiment (Found by Oger Gridsearch).

	ρ	*IScale*	*IDens*
ChainOfNeurons	0.75	0.95	1/9
RingOfNeurons	0.75	0.95	1/9
CyclicSORM	0.8	0.75	1/9
SORM	0.65	0.45	1/5
StandardMat	0.75	0.2	1/8

5.4 NARMA. The target signal in this section will be the tenth-order nonlinear autoregressive moving average (NARMA):

$$t(n+1) = 0.3t(n) + 0.05t(n)\left(\sum_{i=0}^{9} t(n-i)\right) + 1.5u(n-9)u(n) + 0.1.$$

The task is to predict this sequence for one future step. The parameters of this sequence were taken from Jaeger (2003), where we refer to a detailed description of this experiment.

Each neural net has 200 hidden neurons, one input, and one output neuron. The data sets contain 2200 elements each, whereby the first 200 steps are for washout. Again, one data set is used to train the neural net, and we validate on a second one. We measure the NMSE error as usual for this experiment. The grid of the parameter search consists of the input scaling, the weight scaling value of the reservoir, and the number of input connections. For every parameter set, 5 different reservoir initializations, each with 10 different data sets, are tested. Again, we connect input and hidden layer in the two ways mentioned above. Table 5 provides the optimal parameters resulting from the grid search.

Figure 7 plots the average NMSE and the minimum NMSE against the spectral radius. The CyclicSORM, the RingOfNeurons, and the ChainOfNeurons generate the smallest error rates. In contrast to the above experiments, the average error rates of all SORMs this time are worse than those for the random matrices. The variances of the errors are all small (of magnitude 10^{-6}) except for the ESNs with small spectral radius (0.6, 0.65). So we omit the error bars.

6 Conclusion and Outlook

We have introduced design strategies for reservoir weight matrices and analyzed their linear algebraic properties. In contrast to many state-of-the-art articles, we consider superpositions as a source of errors instead of

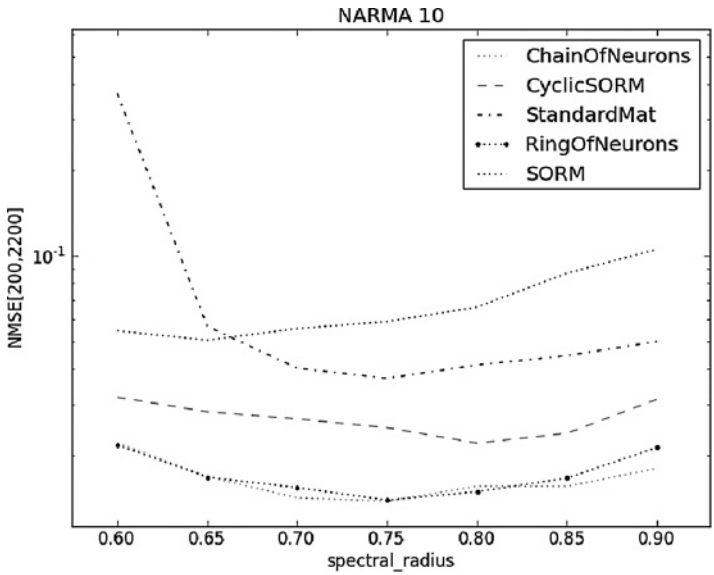

Figure 7: NMSE of the NARMA experiment executed with different reservoir types and averaged over 50 different initializations. The error is measured over 2000 steps after a washout of 200. The spectral radius of the reservoir's weight matrix is plotted on the x-axis.

"interesting echos." In this letter, we compare the different approaches at four academic test scenarios:

- Short-term memory test (delayline experiment)
- Pattern detection
- Discretization of the Mackey-Glass differential equation
- NARMA

Finally, we come up with the surprisingly simple ChainOfNeurons, which is shown to be the most robust reservoir in the linear case and performed very well in our experiments. We point out that the ChainOfNeurons is not even a recurrent system. As we indicated in remark 1, we emphasize that the presented design strategies may not outperform traditional or even all design strategies in all possible scenarios where ESNs can be applied. But we suggest trying the ChainOfNeurons as one possible alternative whenever using ESNs. It appears that the reservoirs do not require the internal dynamics for many tasks, which is basic for the original notion of ESNs. In comparison to the recent echo state approach, the ChainOfNeurons has the following advantages:

- Extremely sparse (1 input connection per unit) and computationally very fast
- Maximum robustness against noisy training data
- No random variants of different reservoirs
- Clearly defined and maximum short-term memory
- Much easier for applying additional approaches as, for example, leaky or bandpass filters (which make the ChainOfNeurons slower from the beginning of the chain to its end)
- Much more intuitive, and therefore easier-to-handle, "echos" (e.g., the historical inputs)

We add the following remarks. First, in case of multidimensional inputs, we recommend a separate ChainOfNeurons for each dimension or a maximum distance between the input vectors on the ChainOfNeurons because the idea is to reduce superpositions as much as possible. Second, remembering previous research, we tested special reservoir topologies, for example, sparse versus densely connected matrices or small-world and fractal connection topologies. We discovered that it does not matter what topology we use for certain tasks. We think that the results presented in this letter fit with this experience. It seems that linear algebraic properties are more essential than the topology of the connections.

From our perspective, there are several interesting open points for further research:

- Training algorithms. It turns out that echo state networks do not work well with gradient descent learning techniques. We think that this is related to the spectrum of eigenvalues of these reservoirs. Different inputs from different timescales will be represented by the echos of the reservoir with extremely different magnitudes. The ChainOfNeurons, as well as SORM and CyclicSORM, have different properties due to representing previous inputs. We think it could be interesting to evaluate different gradient descent learning techniques for the ChainOfNeurons approach. This is also related to the next point.
- Time-delay neural networks (TDNN). The ChainOfNeurons and the TDNN approaches are quite similar. Indeed a ChainOfNeurons could be considered as a certain kind of TDNN at least in the linear case. So we think it will be interesting to evaluate TDNNs and their recent learning techniques in comparison with the ChainOfNeurons approach.
- Implications for reservoir computing technologies. From our perspective, it is an open question of how the presented results will affect the research of other reservoir computing technologies, for example, liquid state machines (LSMs). It would be interesting to evaluate and, if possible, adapt or generalize the presented ideas for related research areas.

Appendix

In this section, we need the generalized or pseudoinverse of a matrix S. Let

$$S = U \begin{pmatrix} \Sigma & 0 \\ 0 & 0 \end{pmatrix} V^*, \quad \Sigma = \begin{pmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \sigma_r \end{pmatrix} \tag{A.1}$$

be the singular value decomposition (SVD) of S with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$, where U and V are unitary, and $\Sigma \in \mathbb{R}^{r \times r}$ (i.e., $r = \operatorname{rank}(S)$). Then the pseudoinverse S^+ of S is defined as

$$S^+ = V \begin{pmatrix} \Sigma^{-1} & 0 \\ 0 & 0 \end{pmatrix} U^*. \tag{A.2}$$

S^+ is uniquely determined by the Moore–Penrose criteria: $SS^+S = S$, $S^+SS^+ = S^+$, $(SS^+)^* = SS^+$ and $(S^+S)^* = S^+S$.

Lemma 4. *Let $V \begin{pmatrix} \Sigma & 0 \\ 0 & 0 \end{pmatrix} V^*$ be the SVD of $A \in \mathbb{R}^{n \times n}$ where $\Sigma \in \mathbb{R}^{r \times r}$ is a regular $r \times r$ diagonal matrix and $J_u = \begin{pmatrix} I_r & 0 \\ 0 & 0 \end{pmatrix}$ ($I_r \in \mathbb{R}^{r \times r}$ is the identity matrix). Let $B \in \mathbb{R}^{n \times m}$ for any $m \in \mathbb{N}$ and $\|A^+B\| < 1$ for some appropriate, submultiplicative matrix norm $\|\cdot\|$. Then*

$$[A(VJ_uV^* + A^+BVJ_uV^*)]^+ = \left(\sum_{i=0}^{\infty} (-1)^i (A^+B)^i \right) A^+. \tag{A.3}$$

Proof. We denote $C := A(VJ_uV^* + A^+BVJ_uV^*)$ and $C^- := (\sum_{i=0}^{\infty}(-1)^i (A^+B)^i)A^+$. Furthermore, we define $(A^+B)^0 = VJ_uV^*$:

$$CC^- = A \left(\sum_{i=0}^{\infty} (-1)^i (A^+B)^i + (-1)^i (A^+B)^{i+1} \right) A^+ \tag{A.4}$$

$$= A \left(VJ_uV^* \right) A^+ = VJ_uV^*. \tag{A.5}$$

In the same way, we get

$$\boldsymbol{C}^{-}\boldsymbol{C} = \left(\sum_{i=0}^{\infty}(-1)^i(\boldsymbol{A}^{+}\boldsymbol{B})^i\right)\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^{*}\left[\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^{*} + \boldsymbol{A}^{+}\boldsymbol{B}\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^{*}\right] \tag{A.6}$$

$$= \left(\sum_{i=0}^{\infty}(-1)^i(\boldsymbol{A}^{+}\boldsymbol{B})^i\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^{*} + (-1)^i(\boldsymbol{A}^{+}\boldsymbol{B})^{i+1}\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^{*}\right) = \boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^{*}. \tag{A.7}$$

Therefore, $(\boldsymbol{C}^{-}\boldsymbol{C})^{*} = \boldsymbol{C}^{-}\boldsymbol{C}$ and $(\boldsymbol{C}\boldsymbol{C}^{-})^{*} = \boldsymbol{C}\boldsymbol{C}^{-}$. Furthermore, $\boldsymbol{C}\boldsymbol{C}^{-}\boldsymbol{C} = \boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^{*}$ $\boldsymbol{A}(\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^{*} + \boldsymbol{A}^{+}\boldsymbol{B}\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^{*}) = \boldsymbol{C}$ and $\boldsymbol{C}^{-}\boldsymbol{C}\boldsymbol{C}^{-} = (\sum_{i=0}^{\infty}(-1)^i(\boldsymbol{A}^{+}\boldsymbol{B})^i)\boldsymbol{A}^{+}\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^{*} = \boldsymbol{C}^{-}$. Thus, the Moore–Penrose criteria are satisfied, and $\boldsymbol{C}^{-}$ is the pseudoinverse of $\boldsymbol{C}$.

It remains to show that the sum converges. Let $\|\boldsymbol{M}\|_{\max} := \max_{1\le i,j\le n}|m_{i,j}|$. The sum $\sum_{i=0}^{\infty}(-1)^i(\boldsymbol{A}^{+}\boldsymbol{B})^i$ converges since for any $\epsilon > 0$, there is some $c \in \mathbb{N}$, such that for any $k \ge l > c$,

$$\left\|\sum_{i=0}^{k}(-1)^i(\boldsymbol{A}^{+}\boldsymbol{B})^i - \sum_{i=0}^{l}(-1)^i(\boldsymbol{A}^{+}\boldsymbol{B})^i\right\|_{\max} = \left\|\sum_{i=l+1}^{k}(-1)^i(\boldsymbol{A}^{+}\boldsymbol{B})^i\right\|_{\max}. \tag{A.8}$$

Because of the equivalence of all norms, there is an α such that

$$\le \alpha\left\|\sum_{i=l+1}^{k}(-1)^i(\boldsymbol{A}^{+}\boldsymbol{B})^i\right\| \tag{A.9}$$

$$\le \alpha\sum_{i=l+1}^{k}\|\boldsymbol{A}^{+}\boldsymbol{B}\|^i \le \epsilon. \tag{A.10}$$

Proof of Theorem 2. Remember that $\widehat{\boldsymbol{w}}^{out} = \boldsymbol{w}^{out} + \boldsymbol{\Delta}\boldsymbol{w}^{out}$ is the solution of $(\boldsymbol{S} + \boldsymbol{\Delta}\boldsymbol{S})\widehat{\boldsymbol{w}}^{out} = \boldsymbol{t}$, whereas $\boldsymbol{w}^{out} := \boldsymbol{t}^{T}\boldsymbol{S}^{+}$. Below, we neglect terms containing $\boldsymbol{\Delta}\boldsymbol{S}$ of higher order:

$$(\boldsymbol{S} + \boldsymbol{\Delta}\boldsymbol{S})(\widehat{\boldsymbol{w}}^{out})^{T} = \boldsymbol{t}, \tag{A.11}$$

$$(\boldsymbol{S} + \boldsymbol{\Delta}\boldsymbol{S})^{T}(\boldsymbol{S} + \boldsymbol{\Delta}\boldsymbol{S})(\widehat{\boldsymbol{w}}^{out})^{T} = (\boldsymbol{S} + \boldsymbol{\Delta}\boldsymbol{S})^{T}\boldsymbol{t}. \tag{A.12}$$

Let

$$\boldsymbol{S} = \boldsymbol{U}\begin{pmatrix}\boldsymbol{\Sigma} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{pmatrix}\boldsymbol{V}^{*} \tag{A.13}$$

the SVD of S. We multiply equation A.12 by $V(J_u + J_l)V^* = I_N$, whereas $J_u = \begin{pmatrix} I_r & 0 \\ 0 & 0 \end{pmatrix}$ and $J_l = \begin{pmatrix} 0 & 0 \\ 0 & I_{N-r} \end{pmatrix}$ are the upper and lower part of the identity matrix. Obviously J_l is a diagonal matrix with $N-r$ one entries and zeros else:

$$\begin{aligned} & V\left(J_u + J_l\right)V^*(S+\Delta S)^T(S+\Delta S)V\left(J_u + J_l\right)V^*(\widehat{\boldsymbol{w}}^{out})^T \\ & \quad = V\left(J_u + J_l\right)V^*(S+\Delta S)^T t \end{aligned} \tag{A.14}$$

We separate this into four equations:

$$VJ_uV^*(S+\Delta S)^T(S+\Delta S)VJ_uV^*(\widehat{\boldsymbol{w}}_1^{out})^T = VJ_uV^*(S+\Delta S)^T t, \tag{A.15}$$

$$VJ_uV^*(S+\Delta S)^T(S+\Delta S)VJ_lV^*(\widehat{\boldsymbol{w}}_2^{out})^T = VJ_uV^*(S+\Delta S)^T t, \tag{A.16}$$

$$VJ_lV^*(S+\Delta S)^T(S+\Delta S)VJ_uV^*(\widehat{\boldsymbol{w}}_3^{out})^T = VJ_lV^*(S+\Delta S)^T t, \tag{A.17}$$

$$VJ_lV^*(S+\Delta S)^T(S+\Delta S)VJ_lV^*(\widehat{\boldsymbol{w}}_4^{out})^T = VJ_lV^*(S+\Delta S)^T t. \tag{A.18}$$

For small $\|\Delta S\|$, we approximate

$$(S+\Delta S)^T(S+\Delta S) \approx (S^TS + \Delta S^TS + S^T\Delta S). \tag{A.19}$$

We analyze equation A.15 first:

$$(\widehat{\boldsymbol{w}}_1^{out})^T \approx [VJ_uV^*(S^TS + S^T\Delta S + \Delta S^TS)VJ_uV^*]^+VJ_uV^*(S+\Delta S)^T t \tag{A.20}$$

$$\begin{aligned} &= [S^TS(VJ_uV^* + (S^TS)^+(\Delta S^TS \\ &\qquad + S^T\Delta S))VJ_uV^*]^+VJ_uV^*(S+\Delta S)^T t. \end{aligned} \tag{A.21}$$

Due to lemma 4, the generalized inverse of $S^TS(VJ_uV^* + (S^TS)^+(\Delta S^TS + S^T\Delta S))VJ_uV^*$ is

$$\begin{aligned} &\left(\sum_{k=0}^{\infty}(-1)^k((S^TS)^+(\Delta S^TS + S^T\Delta S))^k\right)(S^TS)^+ \text{ if } \|\Delta S\| \\ &< \frac{1}{2}\sigma_{\max}(S)^{-1}\sigma_{\min}(S)^2. \end{aligned}$$

We approximate the pseudoinverse by the first two terms:

$$\approx \left[\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^* - (\boldsymbol{S}^T\boldsymbol{S})^+(\boldsymbol{S}^T\boldsymbol{\Delta S} + \boldsymbol{\Delta S}^T\boldsymbol{S})\right](\boldsymbol{S}^T\boldsymbol{S})^+(\boldsymbol{S}+\boldsymbol{\Delta S})^T\boldsymbol{t}, \tag{A.22}$$

$$\approx \underbrace{(\boldsymbol{S}^T\boldsymbol{S})^+\boldsymbol{S}^T\boldsymbol{t}}_{=(\boldsymbol{w}^{out})^T} + (\boldsymbol{S}^T\boldsymbol{S})^+\boldsymbol{\Delta S}^T\underbrace{(\boldsymbol{t}-\boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S})^+\boldsymbol{S}^T\boldsymbol{t})}_{=\boldsymbol{t}-\boldsymbol{S}(\boldsymbol{w}^{out})^T=:\boldsymbol{r}} - (\boldsymbol{S}^T\boldsymbol{S})^+\boldsymbol{S}^T\boldsymbol{\Delta S}\underbrace{(\boldsymbol{S}^T\boldsymbol{S})^+\boldsymbol{S}^T\boldsymbol{t}}_{=(\boldsymbol{w}^{out})^T}. \tag{A.23}$$

The term $\boldsymbol{r}$ is called residuum and describes the approximation error. We obtain a first-order approximation of $\boldsymbol{\Delta w}_1^{out} := \widehat{\boldsymbol{w}}_1^{out} - \boldsymbol{w}^{out}$:

$$(\boldsymbol{\Delta w}_1^{out})^T \approx (\boldsymbol{S}^T\boldsymbol{S})^+\boldsymbol{\Delta S}^T\boldsymbol{r} - (\boldsymbol{S}^T\boldsymbol{S})^+\boldsymbol{S}^T\boldsymbol{\Delta S}(\boldsymbol{w}^{out})^T \tag{A.24}$$

$$= (\boldsymbol{S}^T\boldsymbol{S})^+\boldsymbol{\Delta S}^T\boldsymbol{r} - \boldsymbol{S}^+\boldsymbol{\Delta S}(\boldsymbol{w}^{out})^T. \tag{A.25}$$

For $i \in \{2,3,4\}$, $\widehat{\boldsymbol{w}}_i^{out}$ contains $\boldsymbol{J}_l$ such that we obtain (together with equation A.19),

$$(\widehat{\boldsymbol{w}}_2^{out})^T \approx \left(\boldsymbol{S}^T\boldsymbol{\Delta S}\boldsymbol{V}\boldsymbol{J}_l\boldsymbol{V}^*\right)^+\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*(\boldsymbol{S}+\boldsymbol{\Delta S})^T\boldsymbol{t}, \tag{A.26}$$

$$(\widehat{\boldsymbol{w}}_3^{out})^T \approx \left(\boldsymbol{V}\boldsymbol{J}_l\boldsymbol{V}^*\boldsymbol{\Delta S}^T\boldsymbol{S}\right)^+\boldsymbol{V}\boldsymbol{J}_l\boldsymbol{V}^*\boldsymbol{\Delta S}^T\boldsymbol{t}, \tag{A.27}$$

$$(\widehat{\boldsymbol{w}}_4^{out})^T \approx \boldsymbol{0}. \tag{A.28}$$

The last approximation results from $\boldsymbol{S}\boldsymbol{V}\boldsymbol{J}_l = \boldsymbol{0}$ and $\boldsymbol{J}_l\boldsymbol{V}^*\boldsymbol{S} = \boldsymbol{0}$. Finally, we obtain

$$\boldsymbol{\Delta w}^{out} = \widehat{\boldsymbol{w}}_1^{out} + \widehat{\boldsymbol{w}}_2^{out} + \widehat{\boldsymbol{w}}_3^{out} + \widehat{\boldsymbol{w}}_4^{out} - \boldsymbol{w}^{out} \tag{A.29}$$

$$\begin{aligned}\approx\ & \boldsymbol{r}^T\boldsymbol{\Delta S}(\boldsymbol{S}^T\boldsymbol{S})^+ - \boldsymbol{w}^{out}\boldsymbol{\Delta S}^T(\boldsymbol{S}^+)^T \\ & + \boldsymbol{t}^T[(\boldsymbol{S}+\boldsymbol{\Delta S})\boldsymbol{V}\boldsymbol{J}_u\boldsymbol{V}^*(\boldsymbol{V}\boldsymbol{J}_l\boldsymbol{V}^*\boldsymbol{\Delta S}^T\boldsymbol{S})^+ + \\ & + \boldsymbol{\Delta S}\boldsymbol{V}\boldsymbol{J}_l\boldsymbol{V}^*(\boldsymbol{S}^T\boldsymbol{\Delta S}\boldsymbol{V}\boldsymbol{J}_l\boldsymbol{V}^*)^+].\end{aligned} \tag{A.30}$$

Proof of Lemma 2. If $\operatorname{rank}(\boldsymbol{M}) = \operatorname{rank}(\boldsymbol{S})$, the proposition is obvious because then $\boldsymbol{J}_l$ contains just zeros by definition. So let us prove $\operatorname{rank}(\boldsymbol{M}) = \operatorname{rank}(\boldsymbol{S})$. Recall $\boldsymbol{S}^T = (\boldsymbol{x}(1), \boldsymbol{x}(2), \ldots, \boldsymbol{x}(\theta))$. We assumed that $f = \mathrm{id}$. Thus,

$$\begin{aligned}\operatorname{rank}(\boldsymbol{S}) &= \dim\left\{\sum_{i=0}^{\theta}\alpha_i\boldsymbol{x}(i) \mid \alpha_i \in \mathbb{R}\right\} \\ &= \dim\left\{\sum_{i=0}^{\theta}\sum_{j=0}^{N-1}\alpha_i u(i-j)\boldsymbol{W}^j\boldsymbol{w}^{in} \mid \alpha_i \in \mathbb{R}\right\} \\ &= \dim\left\{\sum_{j=0}^{N-1}\boldsymbol{W}^j\boldsymbol{w}^{in}\sum_{i=0}^{\theta}\alpha_i u(i-j) \mid \alpha_i \in \mathbb{R},\ u(k) = 0 \text{ if } k < 0\right\}.\end{aligned}$$

Due to the N nonzero inputs, for any $j \in \{0, 1, \ldots, N-1\}$, there exist real values $(\alpha_i)_{i=0,\ldots,\theta}$ such that for any $\beta_j \in \mathbb{R}$: $\beta_j = \sum_{i=0}^{\theta} \alpha_i u(i-j)$, such that

$$\operatorname{rank}(\boldsymbol{S}) = \dim(\operatorname{span}\{\boldsymbol{W}^i \boldsymbol{w}^{in} \mid i = 0, 1 \ldots, N-1\}) = \operatorname{rank}(\boldsymbol{M}).$$

Proof of Lemma 3. Let $\boldsymbol{S} \in \mathbb{R}^{\theta \times N}$, that is, θ is the training size.

$$\frac{1}{\theta}\boldsymbol{S}^T\boldsymbol{S} = \frac{1}{\theta}\sum_{t=0}^{\theta}\boldsymbol{x}(t)\boldsymbol{x}(t)^T \tag{A.31}$$

$$= \frac{1}{\theta}\sum_{t=0}^{\theta}\left(\sum_{i=0}^{N-1}\boldsymbol{W}^i\boldsymbol{w}^{in}u(t-i)\right)\left(\sum_{k=0}^{N-1}\boldsymbol{W}^k\boldsymbol{w}^{in}u(t-k)\right)^T \tag{A.32}$$

$$= \frac{1}{\theta}\sum_{i=0}^{N-1}\sum_{k=0}^{N-1}\boldsymbol{W}^i\boldsymbol{w}^{in}(\boldsymbol{W}^k\boldsymbol{w}^{in})^T\sum_{t=0}^{\theta}u(t-k)u(t-i) \tag{A.33}$$

$$= \boldsymbol{M}\boldsymbol{M}^T\sum_{t=0}^{\theta}\frac{(u(t))^2}{\theta} + \sum_{i=0}^{N-1}\sum_{\substack{k=0\\k\neq i}}^{N-1}\boldsymbol{W}^i\boldsymbol{w}^{in}(\boldsymbol{W}^k\boldsymbol{w}^{in})^T\sum_{t=0}^{\theta}\frac{u(t-k)u(t-i)}{\theta}. \tag{A.34}$$

For $k \neq i$,

$$\lim_{\theta\to\infty}\frac{1}{\theta}\sum_{t=0}^{\theta}u(t-k)u(t-i) = \mathbb{E}(u)^2 = 0 \tag{A.35}$$

because $u(t-k)$ and $u(t-i)$ are independently distributed with zero expectation. In case of $k = i$,

$$\lim_{\theta\to\infty}\frac{1}{\theta}\sum_{t=0}^{\theta}(u(t-k))^2 = \mathbb{E}(u^2) = \mathbb{V}(u). \tag{A.36}$$

Let $\sigma_i(\boldsymbol{S})$ and $\lambda_i(\boldsymbol{S})$ be the ith singular value of $\boldsymbol{S}$ and the ith eigenvalue of $\boldsymbol{S}$, respectively. For sufficiently large θ, we approximate

$$\sigma_i\left(\frac{1}{\sqrt{\theta}}\boldsymbol{S}\right) = \sqrt{\frac{1}{\theta}\lambda_i(\boldsymbol{S}^T\boldsymbol{S})} \approx \sqrt{\mathbb{V}(u)\lambda_i(\boldsymbol{M}\boldsymbol{M}^T)} = \sqrt{\mathbb{V}(u)}\sigma_i(\boldsymbol{M}). \tag{A.37}$$

Acknowledgments

This research was entirely funded by the research grant no. V220-630-08-TFMV-S/F-059 (Verbundvorhaben, Technologieförderung Land Mecklenburg-Vorpommern) in European Social/Regional Development Funds. We thank the reviewers for their valuable comments, which improved the letter.

References

Čerňanský, M., & Tiňo, P. (2008). Predictive modeling with echo state networks. In *Proceedings of the 18th International Conference on Artificial Neural Networks, Part I* (pp. 778–787). Berlin: Springer-Verlag.

Jaeger, H. (2001a). *The "echo state" approach to analysing and training recurrent neural networks* (Tech. Rep. GMD Report 148). Sankt Augustin: German National Research Center for Information Technology.

Jaeger, H. (2001b). *Short term memory in echo state networks* (Tech. Rep. GMD Report 152). Sankt Augustin: German National Research Center for Information Technology.

Jaeger, H. (2002). *A tutorial on training recurrent neuronal networks, covering BPPT, RTRL, EKF and the "echo state network" approach* (Tech. Rep. GMD Report 159). Sankt Augustin: German National Research Center for Information Technology.

Jaeger, H. (2003). Adaptive nonlinear system identification with echo state networks. In S. Becker, S. Thrün, & K. Obermayer (Eds.), *Advances in neural information processing systems, 15* (pp. 593–600). Cambridge, MA: MIT Press.

Lukoševičius, M., & Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, *3*(3), 127–149.

Ozturk, M. C., Xu, D., & Príncipe, J. C. (2007). Analysis and design of echo state networks. *Neural Computation*, *19*, 111–138.

Rodan, A., & Tiňo, P. (2011). Minimum complexity echo state network. *IEEE Transactions on Neural Networks*, *22*(1), 131–144.

White, O. L., Lee, D. D., & Sompolinsky, H. (2004). Short-term memory in orthogonal neural networks. *Physical Review Letters*, *92*, 148102.

Received October 12, 2011; accepted July 4, 2012.