scientific reports

Check for updates

OPEN

Deterministic reservoir computing for chaotic time series prediction

Johannes Viehweg[1]✉, Constanze Poll[1] & Patrick Mäder[1,2]

Reservoir Computing was shown in recent years to be useful as efficient to learn networks in the field of time series tasks. Their randomized initialization, a computational benefit, results in drawbacks in theoretical analysis of large random graphs, because of which deterministic variations are still an open field of research. Building upon Next Generation Reservoir Computing and the Temporal Convolution Derived Reservoir Computing, we propose a deterministic alternative to the higher-dimensional mapping therein, TCRC-LM and TCRC-CM, utilizing the parameterized but deterministic Logistic mapping and Chebyshev maps. To further enhance the predictive capabilities in the task of time series forecasting, we propose the novel utilization of the Lobachevsky function as non-linear activation function. As a result, we observe a new, fully deterministic network being able to outperform TCRCs and classical Reservoir Computing in the form of the prominent Echo State Networks by up to 99.99% for the non-chaotic time series and 87.13% for the chaotic ones.

The task of time series prediction is of interest in a multitude of diverse fields, such as fluid dynamics, to medical data, up to trajectories in the financial market. Among the success of machine learning approaches in recent years, recurrent neural networks (RNN) such as the approach of reservoir computing (RC)[27] have been shown to be beneficial for this task[30], being especially of interest in case of chaotic systems[17,18]. Compared to the most prominent examples of RNNs, i.e., Long Short Term Memory Networks (LSTM)[7] and Gated Recurrent Units[2], RC uses a simplified learning method, allowing for a substantial speed-up in the training, e.g. shown by[30]. For the sake of simplicity we limit ourselves in this work to Echo State Networks (ESN)[10] as the most prominent example of RC[14]. In recent years Feed-Forward Neural Networks (FFNN), such as Time Convolutional Networks (TCN)[1,31] and Transformers[26] have also shown success in regard to time-series, e.g.[32]. A feed-forward analogue to RC (RFF) are hereby the Schmidt Networks (SN)[25] or the Random Vector Functional Link Networks (RVFL)[19], later known as Extreme Learning Machine (ELM)[8]. Recent works bridge the difference between the typical RC approach and their feed-forward analogue, such as Next Generation RC[3] (NGRC) and Temporal Convolution RC (TCRC)[28].

Fixed weight networks

In contrast to the established method of learning the weights in an NN by gradient descent[24], RC and RFF based networks use fixed weights for the mapping from input to inner state and for its recurrent connections. Only the weights to the output state are learned, mostly by a single computation[14]. This leads the basic architecture to consist of three layers instead of several layers, as established for conventional NN[6]. Those three layers are the input, state and output layer $x^{(\cdot)} \in \mathbb{R}^{\text{in}}$, $s^{(\cdot)} \in \mathbb{R}^{\text{res}}$ and $y^{(\cdot)} \in \mathbb{R}^{\text{out}}$, respectively. Fixed weights map the input to the state as $W^{\text{in}} \in \mathbb{R}^{N^{\text{res}} \times N^{\text{in}}}$. In the case of RC, the recurrent weights from the state at one time step to the next are also chosen fix, $W^{\text{res}} \in \mathbb{R}^{N^{\text{res}} \times N^{\text{res}}}$. Both of those sets are mostly drawn randomly from a uniform distribution with symmetric limits $W^{\text{in}} \sim \mathcal{U}(-\sigma, \sigma)$, $W^{\text{res}} \sim \mathcal{U}(-\sigma, \sigma)$, but other approaches also exist in literature[29]. The state is hereby computed as

$$s_{RFF}^{(t)} = f(W^{\text{in}} x^{(t)}) \tag{1a}$$

$$s_{RC}^{(t)} = f(W^{\text{in}} x^{(t)} + W^{\text{res}} s^{(t-1)}) \tag{1b}$$

for ELM and ESN alike, aside from the recurrence introduced in the ESN. This leaves only the output weights $W^{\text{out}} \in \mathbb{R}^{N^{\text{out}} \times N^{\text{res}}}$ to be computed, which is done in a single step by use of Tikhonov regularization. We use the notation $Y \in \mathbb{R}^{N^{\text{out}} \times S_T}$ and $S \in \mathbb{R}^{N^{\text{res}} \times S_T}$ as the collections of the targeted outputs and the state over S_T training steps respectively. We assume a mapping of the state space to the output, without stacking with the

[1]Data-intensive Systems and Visualisation Lab, Technische Universität Ilmenau, Helmholtzplatz 5, 98693 Ilmenau, Germany. [2]Faculty of Biological Sciences, Friedrich Schiller University, Philosophenweg 16, 07743 Jena, Germany. ✉email: johannes.viehweg@tu-ilmenau.de

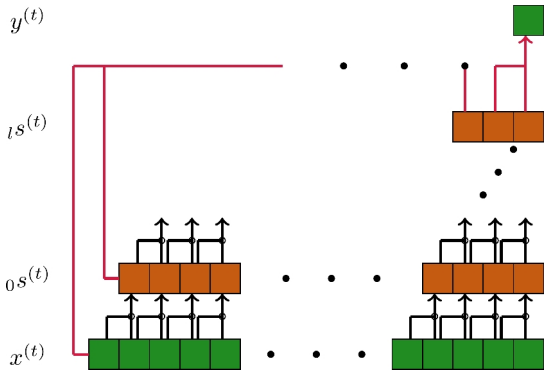


Fig. 1. TCRC architecture; black arrows (→) refer to the multiplied tokens, red arrows () refer to the learned mapping from the state space of each layer ${}_l s^{(t)}$ to the output $\hat{y}^{(t)}$.

input at the same time step for the sake of readability, utilizing the generated state as an ELM of[8] in contrast to the RVFL of[19,22]. With this assumption and use of the Tikhonov regularization the computation is done as

$$W^{\text{out}} = YS^T(SS^T + \beta\mathbb{I})^\dagger, \tag{2}$$

with $(\cdot)^\dagger$ being the Moore-Penrose pseudoinverse, $\beta \in \mathbb{R}$ the regularization coefficient and $\mathbb{I}$ the unity matrix of space $N^{\text{out}} \times N^{\text{res}}$. Regarding the optimization of hyper-parameters the corresponding parameter space is $\Omega^I = \{\rho, N^{\text{res}}, S_T, \beta\}$.

TCRC

The idea of temporal convolutional RC[28] is a mapping of stacked inputs

$$\hat{x}^{(t)} = \#_{\delta=0}^{\hat{\delta}} x^{(t-\delta)} \tag{3}$$

into the first layer of the state space

$${}_0 s_{TC}^{(t)} = f(g(\hat{x}^{(t)})) \tag{4}$$

as exemplary shown in Fig. 1. We use $f(\cdot)$ as the non-linear activation function, analogous to Section Introduction, while $g(\cdot)$ is the deterministic function, multiplying the inputs pairwise and stacking the results. For following layers $l \geq 1$ we use

$${}_l s_{TC}^{(t)} = f(g({}_{l-1} s^{(t)})). \tag{5}$$

The readout is computed analogous to Eq. 2 with the used state being

$$s^{(t)} = \#_{l=0}^{L} {}_l s_{TC}^{(t)} \tag{6}$$

and a final state space size of $N^{\text{tc}} = \sum_{l=0}^{L-1} {}_l N^{\text{tc}}$, equal to the sum of all sizes for the different layers.

The similarity to Next Generation Reservoir Computing (NGRC) of Gauthier et al.[3], especially the therein utilized non-linear function is trivial to see, but this method uses a combination of inputs inspired by Temporal Convolutional Networks (TCN)[1] with a pairwise multiplication of inputs in a temporal neighborhood. This approach leads to a substantially reduced size of the state space ${}_0 s_{TC}^{(\cdot)}$ cp. Eq. 6 as shown in Fig. 2. The used mapping allows for an increasing delay δ, cp. Eq.3, because of the reduced growth rate of the state space compared to the NGRC of[3]. We argue for it to be more usable for history-dependent time series because of this. In regard to the number of hyper-parameters, the search space increases to $\Omega^{II} = \{\hat{\delta}, L, S_T, \beta\}$ with L layers.

For chaotic time series, it was shown in[28] to be beneficial to use additional mapping as depicted in Fig. 3 of

$$\hat{s}^{(t)} = f(W^{\text{tc}} s^{(t)}). \tag{7}$$

For this, a random map was used, analogous to the input weights in Eq. 1 with $W^{\text{tc}} \sim \mathcal{U}(-\hat{\sigma}, \hat{\sigma}),\ W^{\text{tc}} \in \mathbb{R}^{N^{\text{tc}'} \times N^{\text{tc}}}$. For $N^{\text{tc}'} = n \cdot N^{\text{tc}}$ we use $n \in \mathbb{N}$ as a factor for the state space size and additional hyper-parameter, resulting in $\Omega^{III} = \{n, \hat{\delta}, L, S_T, \beta\}$. In this work we will refer to this architecture as TCRC-ELM.

Deterministic mappings

The original propositions of ESNs and RFFs used random mappings (RM) from the input into the higher dimensional state space. While the optimization of structure of these random matrices is an open field of

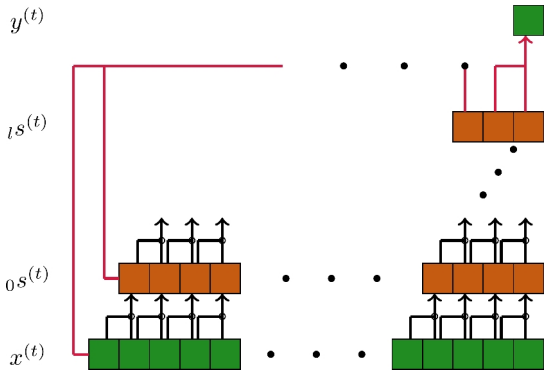


Figure 1. (continued)

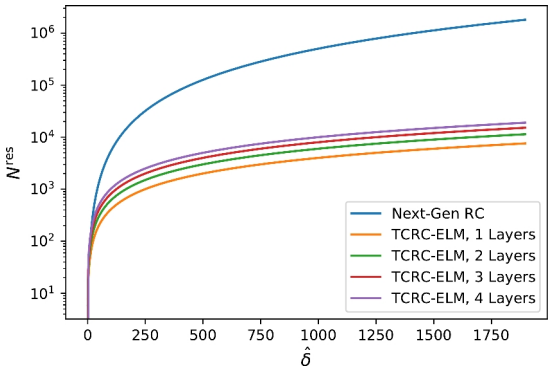


Fig. 2. Comparative depiction of state space size N^{res} of Next Generation RC with the non-linear function utilized by[3] and N^{tc} of TCRC.

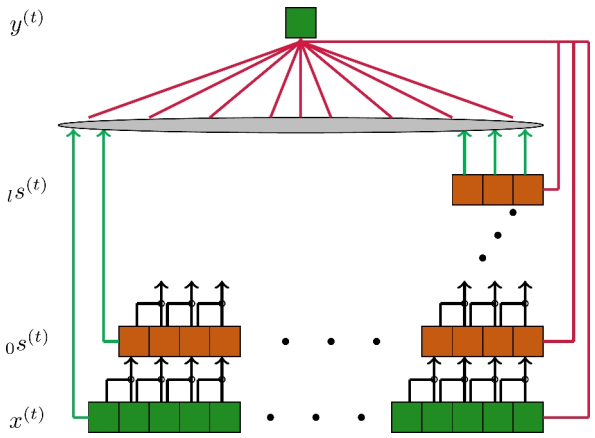


Fig. 3. Exemplary TCRC-ELM architecture; black arrows (→) refer to the multiplied tokens, green arrows () refer to the randomly drawn weights of W^{in}, red arrows () refer to the learned mapping from the state space $\hat{s}^{(t)}$ to the output $\hat{y}^{(t)}$.

→

Figure 3. (continued)

research, deterministic mappings (DM) with known characteristics were proposed in the literature achieving comparable performances to RM, e.g.[33,34]. The use of such known mappings further allows for an optimization by tuning of the parameters, as explained in[33].

Chebyshev mapping

Proposed to be used instead of W^{in} by[34], the Chebyshev map is a deterministic, parameterized approach. This methods introduces additional hyper-parameters $p,\ q,\ k \in \mathbb{R}$. In the frame of this work, we initialize the first row of W^{cheb} as:

$$W_{0,i}^{cheb} = p \cdot \sin\left(\frac{(i-1)\pi}{q(N^{\text{tc}}+1)}\right),\ i \in \mathbb{N}_{i<N^{\text{tc}}} \tag{8}$$

These values are used to fill the remaining values as:

$$W_{j,i}^{cheb} = \cos(k \cdot arccos(W_{j-1,i}^{cheb})), \tag{9}$$

$$i \in \mathbb{N}_{i<N^{\text{tc}}}, j \in \mathbb{N}_{1<j<N^{\text{tc}'}} \tag{10}$$

For the number of hyper-parameters this leads to $\Omega^{IV} = \{n, \hat{\delta}, L, S_T, \beta, p, q, k\}$.

We will refer to the TCRC utilizing the CM as TCRC-CM.

Logistic mapping

An alternative approach to the random mapping of RFFs and ESNs has been proposed by[33]. Their work uses a deterministic logistic map to map the input into a higher dimensional state space. Thereby, we assume a matrix $W^{\text{lm}} \in \mathbb{R}^{N^{\text{tc}'} \times N^{\text{tc}}}$ to be:

$$W^{\text{lm}} = \begin{bmatrix} w_{0,0}^{\text{lm}} \cdots w_{N^{\text{tc}}-1,0}^{\text{lm}} \\ \vdots \ddots \vdots \\ w_{0,N^{\text{tc}'}-1}^{\text{lm}} \cdots w_{N^{\text{tc}}-1,N^{\text{tc}'}-1}^{\text{lm}} \end{bmatrix} \tag{11}$$

In difference to[33], we initialise the weights as

$$w_{i,0}^{\text{lm}} = \mathcal{A} \sin\left(\frac{i\pi}{(N^{\text{tc}'}-1)\mathcal{B}}\right) \tag{12}$$

depending on the column. Hereby $\mathcal{A}, \mathcal{B}$ are additional hyper-parameters for the first row of W^{lm}.

An additional hyper-parameter is r is introduced for the other rows of W^{lm} for

$$w_{i,j}^{\text{lm}} = r w_{i,j-1}^{\text{lm}}(1 - w_{i,j-1}^{\text{lm}}) \tag{13}$$

for $j \in \mathbb{N}_{\geq 1}$. For the discussion about the influence of r we refer to the proposing publication[33], deciding if the mapping is chaotic or not. We will refer to this combination as TCRC-LM. This approach leads to an increased number of hyper-parameters $\Omega^V = \{n, \hat{\delta}, L, S_T, \beta, r, \mathcal{A}, \mathcal{B}\}$ with in total $|\Omega^V| > |\Omega^I|$. In the frame of this work, we further adopted the idea of Parallel ESNs[20]. This results in a sparse matrix of weights for the logistic map, only a fixed number of weights per input, equal to n. In comparison between the maps, the Chebyshev map is dense, while the Logistic map, as used in the frame of this work, is a sparse matrix as exemplary shown in Fig. 4.

Lobachevsky function

Often $tanh(\cdot)$ is used as activation function $f(\cdot)$ as in Section Introduction, shown e.g. in[13]. The observation of values prior to the activation close to 1 for the TCRC leads to the suggestion of a not-saturating mapping onto the state $s^{(\cdot)}$. As a solution, we suggest an activation with intervals of increasing and decreasing values on both sides of the input value 0. An observation of a benefit of non-monotonic activation functions similar to our proposal herein is also reported e.g. by[9]. For this we propose the use of the Clausen function[11]

$$c(s^{(\cdot)}) = \sum_{i=0}^{k} \frac{1}{2^i} \sin(2is^{(\cdot)}) \tag{14}$$

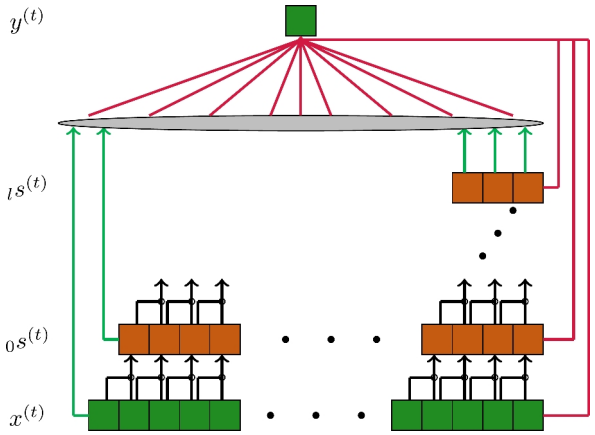
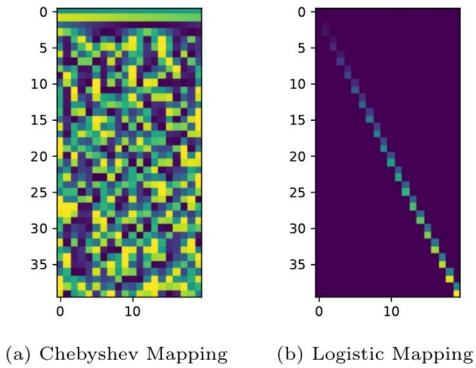


Figure 3. (continued)



(a) Chebyshev Mapping (b) Logistic Mapping

Fig. 4. Exemplary visualizations of the values of the deterministic mappings.

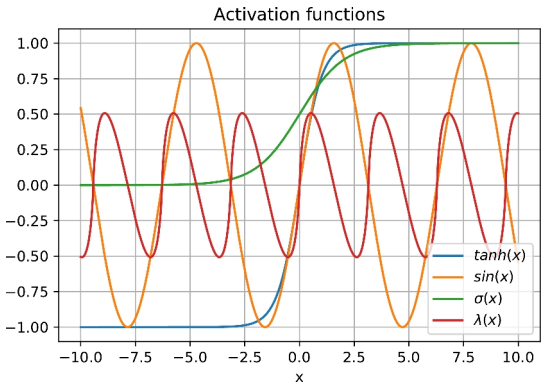


Fig. 5. Plot of the activated value for inputs in the range $[-10, 10]$.

as activation function. It is easy to see that the shape of the function depends on the value of $k \in \mathbb{N}$. In the frame of this work we use this to define an approximation of the Lobachevsky function as

$$\lambda(s^{(\cdot)}) = \frac{c(2s^{(\cdot)})}{2}, \tag{15}$$

as justified by Kellerhals[12] and depicted in Fig. 5. We argue for the use of a periodic function for activation with the observation in[23].

Evaluation

In the following, we will introduce the used metrics and datasets before we report our observations. We discuss those afterwards.

Preprocessing and metric

We preprocess our data by normalizing via the z-score as in Eq. 16[5] prior to using it as input and as evaluation. In the frame of this work, we define the complete set of potential inputs from the unprocessed data as $\hat{X}$, the processed set as $\tilde{X}$ with the actual set of inputs $x^{(\cdot)} \in X \subset \tilde{X}$. The functions $\mu(\cdot)$ and $std(\cdot)$ refer to the mean and standard deviation of the input, respectively. The z-normalization, for sake of simplicity referred to normalization from here onward, itself is defined as:

$$\tilde{X} = \frac{\hat{X} - \mu(\hat{X})}{std(\hat{X})}. \tag{16}$$

For the evaluation we use the Mean Squared Error (MSE) $\mathcal{L}(\hat{Y}, Y)$, defined for a set of outputs $\hat{Y} = \#_{\hat{t}=0}^{S_P-1} \hat{y}^{(\hat{t})}$ and targets $Y = \#_{\hat{t}=0}^{S_P-1} y^{(\hat{t})}$ as:

$$\mathcal{L}(\hat{Y}, Y) = \frac{\sum_{\hat{t}=0}^{S_P-1} (\hat{y}^{(\hat{t})} - y^{(\hat{t})})^2}{S_P}. \tag{17}$$

The error (cp. Eq. 17) is computed directly with the output and the normalized ground truth to achieve a comparability between different datasets. We have chosen ten subsamples from each time series to learn and predict. Additionally we use 15 randomized initializations for each model utilizing randomness. For both of these reasons, we report the mean over the error for multiple runs.

To determine the hyper-parameters, we use Bayesian optimization[6], chapter 11.4.5, p. 423 with the MSE as targeted metric. We compare the found optimal results with the ones reported in the literature[28] for the ESN. Because of the time constraints imposed on the Bayesian Optimization, we argue that this method does not necessarily result in a global optimum, the literature being proof of a potentially better prediction. Even with the compared work not utilizing all ten trajectories for each τ we have chosen to report the lower MSE between the Bayesian Optimization and the literature. For the TCRC, TCRC-ELM, TCRC-CM and TCRC-LM we report only the results of the Bayesian Optimization, even if worse than findings in the literature. In regard to the Next Generation RC of[3] we refer to the explanation by[28] of the best prediction with regard to the MSE being the mean of each test datasets. Because of this we do not further compare our suggested methods to it.

Dataset

As datasets for the evaluation of our approach, we have chosen to use multiple chaotic and non-chaotic time series, derived from the Mackey-Glass equation[4,15] with varying delays $\tau \in \{5, 10, 15, 17, 20, 25\}$ given as:

$$\dot{x}^{(t)} = \frac{\beta_{MG} \Theta x^{(t-\tau)}}{\Theta^n + x^{(t-\tau)^n}} - \gamma x^{(t)}. \tag{18}$$

In accordance with the literature and for the sake of comparability we have chosen $[\beta_{MG}, \Theta, \gamma, n] = [0.2, 1, 0.1, 10]$. At $\tau = 17$ in Eq. 18 the dynamics switch from non-chaotic to chaotic[16].

Predictive capabilities

In the following we want to report the error of the predictions. For this, we divide between the used mappings for sake of readability.

Non-chaotic dynamics

For all $\tau \in [5, 10, 15]$, read all datasets with a delay below the threshold of observed chaosticity, we observe the TCRC-LM to outperform the ESN by multiple magnitudes of error, as reported in Table 1. Limited to the activation via $tanh(\cdot)$ we observe the ESN to result in a better average prediction for $\tau = 15$. Compared to the ESN the performance of the TCRC-LM decreases by 53.36%. As reported in Table 1, we observe the same trend for the comparison of ESN of TCRC-CM. In case of $\tau = 5$ we observe from the worst to best performing TCRC-LM, from an activation with $\sin(\cdot)$ to an activation utilizing $\sigma(\cdot)$ a decrease in the MSE of 89.68%. This corresponds to a decrease compared to the ESN of 99.96% and to the GRU of 99.88%. The TCRC-CM outperforms the ESN and GRU only with use of the $\sigma(\cdot)$ activation, reducing the error by 98.45% and 95.32%, respectively.

The time series computed with a delay of $\tau = 10$ is best predicted utilizing the TCRC-LM with $\sin(\cdot)$ activation, reducing the error by 99.99% compared to the ESN and 76.67% compared to the worst performing TCRC-LM. The GRU shows a benefit compared to the ESN with the error reduced by 94.57%, but compared to the $\sin(\cdot)$-TCRC-LM increased by multiples orders of magnitude. The TCRC-CM outperforms the ESN with a $tanh(\cdot)$ and $\sigma(\cdot)$ activation by 96.93% and 98.23%, respectively. In comparison to the GRU this benefit reduces to 37.95% and 67.38%, respectively.

For the highest used delay below chaosticity the only activation with which the TCRC-LM outperforms the ESN is the proposed λ activation by 64.88%. Compared to the GRU all tested TCRC-LMs result in a better

τ	ESN	GRU	TCRC-CM				TCRC-LM			
	$tanh(\cdot)$	$tanh(\cdot)$	$tanh(\cdot)$	$\sin(\cdot)$	$\sigma(\cdot)$	$\lambda(\cdot)$	$tanh(\cdot)$	$\sin(\cdot)$	$\sigma(\cdot)$	$\lambda(\cdot)$
5	$5.72 \cdot 10^{-3}$	$1.90 \cdot 10^{-3}$	$6.68 \cdot 10^{-3}$	$9.70 \cdot 10^{-1}$	$8.89 \cdot 10^{-5}$	$9.98 \cdot 10^{-1}$	$1.09 \cdot 10^{-5}$	$2.17 \cdot 10^{-5}$	$\mathbf{2.24 \cdot 10^{-6}}$	$9.81 \cdot 10^{-6}$
10	$3.59 \cdot 10^{-2}$	$1.95 \cdot 10^{-3}$	$1.21 \cdot 10^{-3}$	$9.89 \cdot 10^{-1}$	$6.36 \cdot 10^{-4}$	1.00	$9.74 \cdot 10^{-7}$	$\mathbf{6.16 \cdot 10^{-7}}$	$2.64 \cdot 10^{-6}$	$9.09 \cdot 10^{-7}$
15	$5.41 \cdot 10^{-2}$	$2.48 \cdot 10^{-1}$	$1.28 \cdot 10^{-1}$	1.01	$8.25 \cdot 10^{-2}$	1.02	$1.16 \cdot 10^{-1}$	$1.20 \cdot 10^{-1}$	$1.06 \cdot 10^{-1}$	$\mathbf{1.90 \cdot 10^{-2}}$
17	$2.99 \cdot 10^{-1}$	$1.59 \cdot 10^{-1}$	$3.46 \cdot 10^{-1}$	$9.78 \cdot 10^{-1}$	$3.89 \cdot 10^{-1}$	$9.82 \cdot 10^{-1}$	$3.32 \cdot 10^{-1}$	$3.77 \cdot 10^{-1}$	$3.01 \cdot 10^{-1}$	$\mathbf{1.12 \cdot 10^{-1}}$
20	$3.20 \cdot 10^{-1}$	$9.59 \cdot 10^{-1}$	$7.42 \cdot 10^{-1}$	$9.81 \cdot 10^{-1}$	$7.28 \cdot 10^{-1}$	$9.81 \cdot 10^{-1}$	$2.00 \cdot 10^{-1}$	$3.35 \cdot 10^{-1}$	$1.93 \cdot 10^{-1}$	$\mathbf{4.12 \cdot 10^{-2}}$
25	$3.45 \cdot 10^{-1}$	$1.23 \cdot 10^{-1}$	$8.29 \cdot 10^{-1}$	1.01	$7.78 \cdot 10^{-1}$	1.01	$3.00 \cdot 10^{-1}$	$3.64 \cdot 10^{-1}$	$3.41 \cdot 10^{-1}$	$\mathbf{1.16 \cdot 10^{-1}}$

Table 1. MSE of best performing configurations of ESN, GRU, TCRC-CM and TCRC-LM for prediction of 286 time steps for different τ of the Mackey-Glass equation. We present each best configuration for the activation functions $tanh(\cdot)$ and $\lambda(\cdot)$. The best performing model is marked in bold.

$\simeq N^{\text{res}}$	ESN	TCRC	TCRC-ELM	TCRC-CM	TCRC-LM
300	**0.97**	6.88	5.18	50.47	49.87
2000	**22.22**	693.02	176.92	180.09	175.84

Table 2. Runtime of each tested model in seconds. The lowest values are marked in bold.

predictions from 51.61% to 92.34%. For the TCRC-CM no activation results in a better prediction than the ESN and only the $\sigma(\cdot)$ activation outperforms the GRU by 66.73% For the comparison with TCRC and TCRC-ELM, as reported in Table 4, we observe the TCRC and TCRC-ELM to outperform the TCRC-CM in all three cases. The TCRC-LM is also outperformed by TCRC as well as TCRC-ELM in case of the $tanh(\cdot)$ activation for the cases $\tau \in \{5,\ 15\}$. In case of $\tau = 10$ the TCRC-LM outperforms the TCRC and TCRC-ELM with all activation functions. For the TCRC-LM the activation via $tanh(\cdot)$ results in the second worst predictions for all delays. Across the set of activation functions the best prediction is achieved for $\tau = \{5,\ 10,\ 15\}$ with the activations $\sigma(),\ \sin(\cdot) and \lambda(\cdot)$, respectively.

Chaotic Dynamics For the datasets $\tau > 17$ we observe the TCRC-LM with $\lambda(\cdot)$ to be the best predictor for all tested ones as shown in Table 1. In comparison to the ESN the TCRC-LM with $tanh(\cdot)$ results in a better prediction only for the two most chaotic datasets. In case of $\tau = 17$ the performance decreases from the ESN to the TCRC-LM by 9.94%. We observe for the more chaotic cases an increase in performance, measured as a decrease in MSE by 37.50% and 13.04% for $\tau = 20$ and $\tau = 25$, respectively. In comparison to the GRU the TCRC-LM is inferior in all cases if activated via $tanh(\cdot)$ except $\tau = 20$. Compared to the ESN the error decreases by use of the $\lambda(\cdot)$ activated TCRC-LM by 62.54%, 87.13% and 66.38% for the values of τ in increasing order.

For $\tau = 17$ the GRU outperforms the ESN and the ESN all TCRC-LMs except the $\lambda(\cdot)$ activated one. Only the $tanh(\cdot)$ and $\sigma(\cdot)$ activated TCRC-CMs show to have learned a dynamic, with only the $tanh(\cdot)$ activated one outperforming the worst performing TCRC-LM by 8.22%.

For $\tau = 20$ we observe the GRU to be seemingly not being capable to learn the dynamics of the time series, resulting in the worst prediction after the $\sin(\cdot)$ and $\lambda(\cdot)$ activated TCRC-CM. In case of $\tau = 25$ the GRU is the second best performing predictor after the $\lambda(\cdot)$ activated TCRC-LM with a benefit of latter of 5.69%. In regard to the baselines of TCRC and TCRC-ELM, reported in Table 4, both of them outperform the TCRC-CM for all activation functions. The TCRC-LM is outperformed for $\tau \in \{17, 20\}$ in case of the $tanh(\cdot)$ activation by $\{213.21\%,\ 88.68\%\}$ and $\{3.43\%,\ 53.85\%\}$ compared to TCRC and TCRC-ELM, respectively. Utilizing the $\lambda(\cdot)$ activation the TCRC-LM outperforms the TCRC-ELM for all three datasets by 65.11%, 68.31%, and 63.52% in order of increasing values of τ. It also outperforms the TCRC for $\tau \in \{20, 25\}$ by 60.38% and 65.17%, respectively.

Computational cost

For the comparison of the computational cost all models were tested on a computation node with an Intel Xeon Gold 5318Y at 2.10 GHz and for the runtime on a NVidia A40 GPU. The tests were run in a Jupyter Notebook terminal on Ubuntu 22.04 LTS. We observe a drastic increase of the computational time for a single run with a large N^{res} for the TCRC-based approaches as shown in Table 2. The smaller delay resulting in a larger state space for TCRC-ELM, -CM and -LM results in those being substantially faster than the TCRC compared on the state space size. The TCRC-ELM is on average faster than the deterministic approaches for a smallest N^{res}, while the TCRC-LM is faster for the largest one. We argue this to be because of the difference in sparsity between the random matrix and the Logistic map with the latter being sparse. The slow down compared to the ESN is of factor 7.91. Albeit a substantial increase of 153.62 seconds for a single run the number of total runs is decreased from the number of seeds, e.g. 15 in the frame of this work, up to 100 in the literature,[29] to one with use of the TCRC-LM.

We observe the ESN to be the most memory-saving predictor with a substantial difference of 15.10% compared to the second smallest predictor, the TCRC-CM and 17.94% smaller than the TCRC as shown in Table 3. In total the ESN saves up to 417.25 MiB. In combination with the runtime reported in Table 2 we observe a substantial benefit of ESNs in case of limitations in regard to time and also an advantage in regard to memory.

$\simeq N^{\mathrm{res}}$	ESN	TCRC	TCRC-ELM	TCRC-CM	TCRC-LM
2000	**2447.18**	2864.43	2833.801	2816.79	2831.21

Table 3. Maximum memory needed by each tested model in MBit. The lowest values are marked in bold.

Discussion

For the non-chaotic time series, we observe a benefit for the TCRC-LM compared to the GRU and ESN as shown in Table 1. Hereby, the ESN shows the smallest change in error getting close to chaos. The decrease in error for the TCRC-LM, with the exception of the $\sigma(\cdot)$ activation, being universal, we argue to show a shortcoming of our proposal in regard to handling high frequencies, being better at the slower period of $\tau = 10$.

In regard to the chaotic time series, we argue for $\tau = 20$ being a special case with the $\lambda(\cdot)$ activated TCRC-LM reducing the error substantially and the GRU showing an also substantially worse prediction than the ESN. In total, our proposed approach shows not the clear trend to an increasing error as observable for the ESN, closer to the self-adaptation of the GRU. Compared to TCRC and TCRC-ELM our proposed approaches do not result in a substantial increase of predictive performance. The same we observe in the comparison with the ESN for the chaotic time series.

Regarding the runtime of a single prediction, the ESN shows a substantial speedup, albeit decreasing with an increase in state space size. We argue for our approach, while being the second slowest, as shown in Table 2, to be in application be able to be the fastest, by elimination of the randomness implied necessity for multiple runs, especially with the needed memory being comparable between the methods when scaled to the same state space size as shown in Table 3. We acknowledge the substantial increase in runtime compared to the ESNs to limit the use-cases of the proposed architecture. In case of a limited time to achieve a sufficient prediction, the ESN shows to be beneficial. In use-cases with a sufficient amount of memory and limitations in regard to the time parallelized ESNs are still beneficial. We argue because of this for a benefit of our proposal in case of sufficient runtime and a limited number of ESNs that can be applied in parallel. In regard to works aiming to optimize the networks our approaches limit this task to an optimization of hyper-parameters, without the need to analyze the random graphs.

Limitations

Our harshest limitation we argue is the comparability of runtimes between the established and proposed approaches. For the generation of random matrices highly optimized standard libraries are used, functions non-existent for the Chebyshev and Logistic map, as well as to be found as difference between the established activation functions and the Lobachevsky activation. An in-depth study of the memory and runtime differences would need further studies about the influence of used programming language, wrappers, precompiled code and can even break down to differences in the support on hardware level. The focus of our work is the report of the possibility of a deterministic approach to Reservoir Computing but not a discussion of the behavior of the network because of characteristics of the mappings.

Conclusion

In this work we have proposed an adaptation of high-dimensional mappings to the approach of Reservoir Computing as well as a novel activation function. The combination of the deterministic approach of a Logistic map with the Lobachevsky activation resulted in the smallest predictive error for the highest tested delays, in all tested cases superior to the ESN. Compared to the TCRC-ELM our approach reduces the error for chaotic time series by up to 68.08%. The needed memory is slightly increased for our proposed approach with a runtime, substantially reduced compared to multiple needed runs of random mappings. We have further given an intuitive understanding of a reasoning why the proposed combination results in better prediction while contrasting the current best practice in regard to activation functions.

τ	TCRC	TCRC-ELM	TCRC-CM				TCRC-LM			
	$tanh(\cdot)$	$tanh(\cdot)$	$tanh(\cdot)$	$sin(\cdot)$	$\sigma(\cdot)$	$\lambda(\cdot)$	$tanh(\cdot)$	$sin(\cdot)$	$\sigma(\cdot)$	$\lambda(\cdot)$
5	$7.40 \cdot 10^{-6}$	$\mathbf{1.24 \cdot 10^{-6}}$	$6.68 \cdot 10^{-3}$	$9.70 \cdot 10^{-1}$	$8.89 \cdot 10^{-5}$	$9.98 \cdot 10^{-1}$	$1.09 \cdot 10^{-5}$	$2.17 \cdot 10^{-5}$	$2.24 \cdot 10^{-6}$	$9.81 \cdot 10^{-6}$
10	$6.66 \cdot 10^{-5}$	$1.03 \cdot 10^{-5}$	$1.21 \cdot 10^{-3}$	$9.89 \cdot 10^{-1}$	$6.36 \cdot 10^{-4}$	1.00	$9.74 \cdot 10^{-7}$	$\mathbf{6.16 \cdot 10^{-7}}$	$2.64 \cdot 10^{-6}$	$9.09 \cdot 10^{-7}$
15	$4.21 \cdot 10^{-2}$	$\mathbf{1.21 \cdot 10^{-2}}$	$1.28 \cdot 10^{-1}$	1.01	$8.25 \cdot 10^{-2}$	1.02	$1.16 \cdot 10^{-1}$	$1.20 \cdot 10^{-1}$	$1.06 \cdot 10^{-1}$	$1.90 \cdot 10^{-2}$
17	$\mathbf{1.06 \cdot 10^{-1}}$	$3.21 \cdot 10^{-1}$	$3.46 \cdot 10^{-1}$	$9.78 \cdot 10^{-1}$	$3.89 \cdot 10^{-1}$	$9.82 \cdot 10^{-1}$	$3.32 \cdot 10^{-1}$	$3.77 \cdot 10^{-1}$	$3.01 \cdot 10^{-1}$	$1.12 \cdot 10^{-1}$
20	$1.04 \cdot 10^{-1}$	$1.30 \cdot 10^{-1}$	$7.42 \cdot 10^{-1}$	$9.81 \cdot 10^{-1}$	$7.28 \cdot 10^{-1}$	$9.81 \cdot 10^{-1}$	$2.00 \cdot 10^{-1}$	$3.35 \cdot 10^{-1}$	$1.93 \cdot 10^{-1}$	$\mathbf{4.12 \cdot 10^{-2}}$
25	$3.33 \cdot 10^{-1}$	$3.18 \cdot 10^{-1}$	$8.29 \cdot 10^{-1}$	1.01	$7.78 \cdot 10^{-1}$	1.01	$3.00 \cdot 10^{-1}$	$3.64 \cdot 10^{-1}$	$3.41 \cdot 10^{-1}$	$\mathbf{1.16 \cdot 10^{-1}}$

Table 4. MSE of best performing configurations of TCRC, TCRC-ELM, TCRC-CM and TCRC-LM for prediction of 286 time steps for different τ of the Mackey-Glass equation. We present each best configuration for the activation functions $tanh(\cdot)$ and $\lambda(\cdot)$. The best performing model is marked in bold. The lowest values are marked in bold.

Data availability

The datasets and implementations of the utilized models are shared at https://doi.org/10.7910/DVN/3JUBLR[21].

Comparison with TCRC and TCRC-ELM

See (Table 4)

Received: 14 November 2024; Accepted: 9 April 2025
Published online: 21 May 2025

References

1. Bai, S., Kolter, J. Z. & Koltun, V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint*arXiv:1803.01271, (2018).
2. Cho, K., Merriënboer, Bart Van, Bahdanau, D. & Bengio, Y. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint*arXiv:1409.1259, (2014).
3. Gauthier, D. J., Bollt, E., Griffith, A. & Barbosa, W. A. S. Next generation reservoir computing. *Nat. Commun.* **12**(1), 1–8 (2021).
4. Glass, L. & Mackey, M. *Mackey-glass equation. Scholarpedia***5**(3), 6908 (2010).
5. Gökhan, A., Güzeller, C. O. & Eser, M. T. The effect of the normalization method used in different sample sizes on the success of artificial neural network model. *Int. J. Assess. Tools Educ.* **6**(2), 170–192 (2019).
6. Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning*. MIT Press, (2016). http://www.deeplearningbook.org.
7. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **9**(8), 1735–1780 (1997).
8. Huang, G. B., Zhu, Q.Y. & Siew, C. K. Extreme learning machine: a new learning scheme of feedforward neural networks. In *2004 IEEE international joint conference on neural networks (IEEE Cat. No. 04CH37541)*, volume 2, pages 985–990. IEEE, (2004).
9. Hurley, L. A., Restrepo, J. G. & Shaheen, S. E. Tuning the activation function to optimize the forecast horizon of a reservoir computer. *J. Phys. Complex.* **5**(3), 035004 (2024).
10. Jaeger, H. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report***148**(34), 13 (2001).
11. *Journal für die reine und angewandte Mathematik*. de Gruyter, (1832).
12. Kellerhals, R. The dilogarithm and volumes of hyperbolic polytopes. *AMS Math. Surveys Monogr.* **37**, 301–336 (1991).
13. Liao, Y. & Li, H. Deep echo state network with reservoirs of multiple activation functions for time-series prediction. *Sādhanā* **44**, 1–12 (2019).
14. Lukoševičius, M. & Jaeger, H. Reservoir computing approaches to recurrent neural network training. *Comput. Sci. Rev.* **3**(3), 127–149 (2009).
15. Mackey, M. C. & Glass, L. Oscillation and chaos in physiological control systems. *Science* **197**(4300), 287–289 (1977).
16. Moon, J., Yuting, W. & Lu, W. D. Hierarchical architectures in reservoir computing systems. *Neuromorphic Comput. Eng.* **1**(1), 014006 (2021).
17. Pandey, S. & Schumacher, J. Reservoir computing model of two-dimensional turbulent convection. *arXiv preprint*arXiv:2001.10280, (2020).
18. Pandey, S., Teutsch, P., Mäder, P. & Schumacher, J. Direct data-driven forecast of local turbulent heat flux in rayleigh-bénard convection. *Phys. Fluids* **34**(4), 045106 (2022).
19. Pao, Y.-H., Park, G.-H. & Sobajic, D. J. Learning and generalization characteristics of the random vector functional-link net. *Neurocomputing* **6**(2), 163–180 (1994).
20. Pathak, J., Hunt, B., Girvan, M., Zhixin, L. & Ott, E. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Phys. Rev. Lett.* **120**(2), 024102 (2018).
21. Replication data for: Deterministic reservoir computing for chaotic time series prediction.
22. Ribeiro, V. H. A., Santana, R. & Reynoso-Meza, G. Random vector functional link forests and extreme learning forests applied to uav automatic target recognition. *Eng. Appl. Artif. Intell.* **117**, 105538 (2023).
23. Merino, E. R., Sopena, J. M., Mancho, R. A. & Moliner, J. L. Neural networks with periodic and monotonic activation functions: a comparative study in classification problems. (2000).
24. Rumelhart, D. E., Hinton, G. E & Williams, R. J. Learning internal representations by error propagation, parallel distributed processing, explorations in the microstructure of cognition, ed. de rumelhart and j. mcclelland. vol. 1. 1986. *Biometrika*, 71:599–607, (1986).
25. Schmidt, W. F., Kraaijveld, M. A., Duin, R. P.W., et al. Feed forward neural networks with random weights. In *International conference on pattern recognition*, pages 1–1. IEEE Computer Society Press, (1992).
26. Vaswani, A. et al. Attention is all you need. *arXiv preprint* (2017) arXiv:1706.03762.
27. Verstraeten, D., Schrauwen, B., D'Haene, M. & Stroobandt, D. An experimental unification of reservoir computing methods. *Neural Netw.* **20**, 391–403 (2007).

28. Viehweg, J., Walther, D., Mäder, P. Temporal convolution derived multi-layered reservoir computing. *arXiv preprint*. arXiv:2407.06771 (2024).
29. Viehweg, J., Worthmann, K. & Mäder, P. Parameterizing echo state networks for multi-step time series prediction. *Neurocomputing* (2022).
30. Vlachas, P. R. et al. Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics. *Neural Networks* (2020).
31. Walther, D. et al. Automatic detection and prediction of discontinuities in laser beam butt welding utilizing deep learning. *J. Adv. Joining Process.* **6**, 100119 (2022).
32. Walther, D., Viehweg, J., Haueisen, J. & Mäder, P. A systematic comparison of deep learning methods for eeg time series analysis. *Frontiers in Neuroinformatics***17**, 6.
33. Wang, H. et al. Echo state network with logistic mapping and bias dropout for time series prediction. *Neurocomputing* **489**, 196–210 (2022).
34. Xie, M., Wang, Q. & Simin, Yu. Time series prediction of esn based on chebyshev mapping and strongly connected topology. *Neural Process. Lett.* **56**(1), 1–22 (2024).

Funding

Open Access funding enabled and organized by Projekt DEAL.
We are funded by the Carl Zeiss Foundation's grant: P2018-02-001. the German Federal Ministry of Education and Research (BMBF) grant: 02P22A040. the Federal Ministry for the Environment, Nature Conservation, Nuclear Safety and Consumer Protection grant: 67KI2086A.

Additional information

Correspondence and requests for materials should be addressed to J.V.