Neurocomputing 192 (2016) 128–138

Contents lists available at ScienceDirect

Neurocomputing

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

Optimizing recurrent reservoirs with neuro-evolution

Sebastian Otte [a,*], Martin V. Butz [b], Danil Koryakin [b], Fabian Becker [a], Marcus Liwicki [c], Andreas Zell [a]

[a] University of Tübingen, Cognitive Systems Group, Tübingen, Germany
[b] University of Tübingen, Cognitive Modeling Group, Tübingen, Germany
[c] University of Kaiserslautern, Multimedia Analysis and Data Mining, Kaiserslautern, Germany

ARTICLE INFO

Article history:
Received 11 July 2015
Received in revised form
27 January 2016
Accepted 28 January 2016
Available online 27 February 2016

Keywords:
Recurrent Neural Networks
Reservoir Computing
Echo State Networks
Neuro-Evolution
Learning dynamics

ABSTRACT

This paper extends an efficient and powerful approach for learning dynamics with Recurrent Neural Networks (RNNs). We use standard RNNs with one fully connected hidden layer instead of prestructured RNNs. By combining a variant of Differential Evolution (DE) with least squares optimization, the optimized RNNs are able to learn several common Multiple Superimposed Oscillator (MSO) multiple orders of magnitude better than the achieved results published so far. Furthermore, for new and even more difficult instances up to twelve superimposed waves, our setup achieves lower error rates than reported previously for the best system on just eight waves. Further findings regarding our approach are deepened and a variety of additional experiments are performed, revealing interesting insights into the behavior of the trained networks. MSO reproduction studies show that the resulting RNNs generalize well: the optimized RNNs are able to identify those subcomponents currently active in an MSO sequence, and they are able to do so even when the subcomponents are phase-shifted and have different amplitudes compared to the original training sequence. On the other hand, the generalization to other oscillation frequencies is possible only to a limited extent. In this case, the RNNs fall back to a weighted combination of signals that reflect the frequency spectrum of the originally learned dynamics after the washout phase. Finally, evaluations of the system on the nonlinear *Hénon attractor* suggest that our approach can also be used to predict and generate non-linear dynamics more effectively than previous approaches.

1. Introduction

Sequence generation is an important aspect of neural dynamic systems, such as the generation of locomotive patterns for mobile robots [1], robot arm control [2] or visual attention trajectories [3]. Recurrent Neural Network (RNN) based systems have been used, for example, to generating hand written patterns [4]. This and other applications demonstrate the great potential of RNNs for sequence generation.

Recently [5], a novel and simple method that allows learning recurrent dynamics effectively and highly precisely was presented. The outlined approach evolves the input and the hidden reservoir weights with a variant of *Differential Evolution* (DE) [6], while the output weights are computed as in standard *Echo State Networks* (ESN) [7] using least squares optimization. The work focused on the *Multiple Superimposed Oscillator* (MSO), a superposition of multiple sine waves with different frequencies. The MSO is a typical benchmark in the field of Recurrent Neural Networks (RNNs) for generating dynamics.

Learning to generate a single sine wave is a relatively easy task. However, learning a superimposition of more than one wave is clearly more difficult. In the past, learning even a two-wave superimposition with RNNs was considered as almost impossible [7], at least with ESNs. Although this was later disproved, the problem remained difficult due to the necessity to form multiple independent oscillators in one closed system [8]. Hence, the benchmark was subject to several studies over the last years [8–12].

Of particular importance is the work of Danil et al. [9], in which *compact balanced ESNs* were presented that, for the first time, were able to learn even a difficult MSO with eight superimposed waves with high precision. Thereby, it was shown that the scale of the driving signal is crucial and its tuning is even more important than tuning the spectral radius. Similar findings were discovered in [13]. Apart from the importance of the driving signal, though, DE optimized networks [5] were able to learn even more difficult MSO instances than those presented in [9]. Up to twelve superimposed waves could be approximated within seconds and with a test error much smaller than previously reported. Although a

* Corresponding author.
E-mail address: sebastian.otte@uni-tuebingen.de (S. Otte).

http://dx.doi.org/10.1016/j.neucom.2016.01.088
0925-2312/© 2016 Elsevier B.V. All rights reserved.

systematic study by Ilonen et al. [14] discovered that DE does not provide any distinct advantage particularly compared with gradient-based methods, the DE optimized networks, in which the reservoir weights of an ESN are tuned dependent on the remaining residual after optimizing the output weights by means of least squares optimization, was shown to reach unprecedented levels of performance [5].

Based on the original contribution [5] revisited in Sections 2, 3 and 4.1, this paper gives a comprehensive analysis of the method and the generated networks. Our research covers general investigations concerning the effect of the reservoir size and the feedback scale, the sensitivity related to feedback scale modulation, the influence of the length of the washout phase, and the behavior when noise is added to the feedback signal. Furthermore, we study the behavior of trained networks regarding their performance on unknown dynamics, namely, dropped frequencies and varied amplitudes as well as phase-shifts. It turns out that the networks, even those trained on the default dynamics, are capable of precisely adapting to these varied dynamics immediately and without any further training. We also show that networks driven by a signal consisting of random frequencies fall back to the learned frequency spectrum once it is driven only by its own output. In this context the response in the frequency domain of the networks after they were stimulated by single sine waves of a certain frequency is analyzed. Moreover, we answer the question of whether it is possible to remap the dynamics of trained RNNs to an MSO with varied frequencies. Finally, as an outlook for further research we show that when using DE optimization in the same setup, it is also possible to learn even non-linear chaotic dynamics, particularly the *Hénon attractor*, more accurately than previous approaches.

This paper is organized as follows. The essential aspects of the used neural network architecture are revisited in Section 2. In Section 3 the DE meta-heuristic is briefly presented and the applied mutation scheme is introduced. Section 4 contains all the experiments we performed regarding the MSO benchmark. Afterwards, Section 5 addresses learning the non-linear Hénon attractor. A summary and future work considerations conclude the paper.

2. Recurrent Neural Networks

In this research we used Recurrent Neural Networks for which we avoided any topological pre-structuring or optimization as in Evolino [8], as well as other tuning mechanisms, such as regularizing the spectral radius of the reservoir weights. The latter is an usual procedure for ESNs to force the *echo state property*. This property ensures that information of initial conditions will be washed out over time, which is essential for the principle of ESNs to work.

Our method addresses RNNs, which can be seen as ESNs due to the way we are computing the output weights. An illustration of the general architecture is depicted in Fig. 1. For the particular experiments in this paper, the RNNs have just one single input cell, one linear output cell, and a fully-connected hidden layer (dynamic reservoir), which consists of hyperbolic tangent cells. However, the proposed method is in principle not restricted to this setup.

While the input, hidden, and output feedback weights are generated as candidate solution vectors by the optimizer, which will be discussed in more detail in Sections 3 and 4, the output weights are computed for each such vector via linear least squares, as in ESNs [7] or Evolino [8], thus mapping the internal dynamics

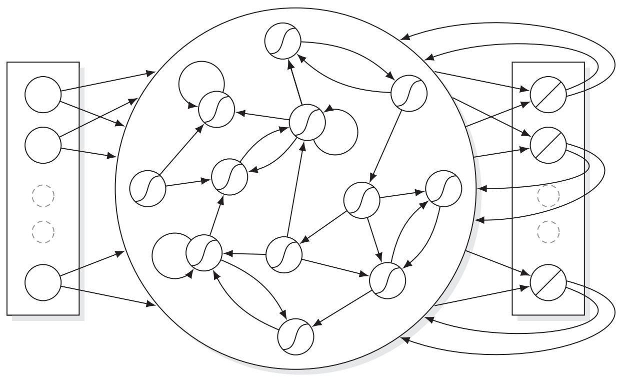

Fig. 1. The general RNN architecture, which can be trained with our method; the core is a dynamic reservoir with recurrent connections (in our case the reservoir is fully connected). As in classic ESNs, the dynamics of the reservoir are mapped linearly onto the linear cells in the output layer. Optionally, an input layer as well as output feedback connections will be used. The input weights, the reservoir weights, and the output feedback weights are optimized with DE. The output weights are computed using least squares.

optimally on the desired output sequence by solving

$$\mathbf{XW} = \mathbf{Z} \Leftrightarrow \mathbf{X}^+\mathbf{XW} = \mathbf{X}^+\mathbf{Z} \Leftrightarrow \mathbf{W} = \mathbf{X}^+\mathbf{Z}, \tag{1}$$

where $\mathbf{Z} \in \mathbb{R}^{T\times K}$ (k gives the number of output units) is the target sequence, $\mathbf{X} \in \mathbb{R}^{T\times H}$ (H gives the number of hidden units) is a matrix containing all hidden activations x_h^t over time, whereas $\mathbf{X}^+ \in \mathbb{R}^{H\times T}$ is the pseudoinverse of $\mathbf{X}$. $\mathbf{W} \in \mathbb{R}^{H\times K}$ is the output weight matrix mapping the hidden dynamics to the linear output neurons. In other words, we are looking for a weighted superimposition of the hidden activation sequences to model the target sequence as accurate as possible. In this paper, Eq. (1) is solved using a singular value decomposition of $\mathbf{X}$.

3. Differential Evolution

Differential Evolution (DE) is a population-based evolutionary algorithm for global optimization [6] in continuous space. DE has been shown to yield good results on a variety of benchmark functions as well as on real world problems [15].

In its simplest form, a so-called donor vector $\mathbf{v}_{i,G}$ is created for each target solution $\mathbf{u}_i$, by randomly selecting three unique vectors $\mathbf{u}_{r_1,G}$, $\mathbf{u}_{r_2,G}$ and $\mathbf{u}_{r_3,G}$ from the current population of size NP in generation G.

$$\mathbf{v}_{i,G} = \mathbf{u}_{r_1,G} + F(\mathbf{u}_{r_2,G} - \mathbf{u}_{r_3,G}) \tag{2}$$

The differential weight F scales the difference vector, which is added to the first randomly selected vector. Elements of the donor vector enter the trial vector with probability CR. The fitness of the trial vector is compared to that of the original target solution, and whichever yields a better result is admitted to the next generation.

However, several different and more sophisticated mutation schemes have been proposed in the past (see e.g. [15] for more details). For many problems the `DE/target-to-best/1` scheme performed best because it offers a good trade-off between convergence behavior and the number of fitness evaluations:

$$\mathbf{v}_{i,G} = \mathbf{u}_{i,G} + F(\mathbf{u}_{best,G} - \mathbf{u}_{i,G}) + F(\mathbf{u}_{r_1,G} - \mathbf{u}_{r_2,G}) \tag{3}$$

Hereby, $\mathbf{u}_{best,G}$ refers to the best solution of the current generation. We compared the `DE/target-to-best/1` scheme in preliminary experiments with other popular mutation schemes and studied the influence of the control parameters. Most of them showed good results under different conditions, related to our problem, but sometimes did not perform very efficiently, since they need a high number of problem evaluations (many generations and high population sizes). Over different reservoir sizes we deduced that the following mutation scheme, which can be seen as an adapted version of the `DE/target-to-best/1` mutation, works typically better than the others:

$$\mathbf{v}_{i,G} = \mathbf{u}_{i,G} + F(\mathbf{u}_{best,G} - \mathbf{u}_{r_1,G}) + F(\mathbf{u}_{r_2,G} - \mathbf{u}_{r_3,G}) \tag{4}$$

In contrast to `DE/target-to-best/1`, a random solution is subtracted from $\mathbf{u}_{best,G}$, while $\mathbf{u}_{i,G}$ remains as the base vector. Note that in `random-to-best` strategies a random base is usually used. Surprisingly, this mutation scheme, which was to our knowledge before proposed in [5] not propagated in the common literature as a successful mutation scheme, results in a faster and deeper convergence, especially when using very small populations ($NP < 10$). We therefore used this scheme for all experiments within this paper, seeing that due to the smaller number of required fitness evaluations DE becomes a powerful tool for optimizing the hidden weights of RNNs.

4. Learning Multiple Super-imposed Oscillators

The first experimental part of this paper addresses the Multiple Superimposed Oscillator (MSO) benchmark. The MSO dynamics used for the experiments are generated using the equation

$$\mathrm{MSO}_n(t) = \sum_{i=1}^{n} a_i \sin(f_i t + \varphi_i), \tag{5}$$

where n gives the number of superimposed waves, f_i the frequency, a_i the amplitude, and φ_i the phase-shift of each particular wave. In this paper, MSO instances of up to twelve waves are investigated using the following frequencies:

$$f_1 = 0.2,\quad f_2 = 0.311,\quad f_3 = 0.42,\quad f_4 = 0.51,$$
$$f_5 = 0.63,\quad f_6 = 0.74,\quad f_7 = 0.85,\quad f_8 = 0.97,$$
$$f_9 = 1.08,\quad f_{10} = 1.19,\quad f_{11} = 1.27,\quad f_{12} = 1.32.$$

For instance, MSO_5 contains the frequencies from f_1 to f_5, MSO_6 contains the frequencies from f_1 to f_6 and so on. The first eight frequencies are taken from [9], and the remaining four were added in [5]. In the default benchmark, which we used for training, all amplitudes are 1 and all phase-shifts are 0.

The usual MSO benchmark consists of the first 700 time steps of one particular time series. The first 100 time steps are used as a *washout phase*. Here, the target signal of the previous time step is fed into the network but the network output is completely ignored. The next 300 time steps are the *training phase*, in which the target signal is also injected and the output prediction error is measured. The last 300 time steps are used as the *test phase*, where the network is fed with its own output from the previous time step. An entire fitness evaluation works as follows: the hidden weights provided by the optimizer are copied into the RNN. Next, washout and a training phase are performed. Using the activation dynamics during the training phase the output weights are computed by solving Eq. (1). With these output weights copied into the RNN, a second washout and training phase are performed. The performance in the second training phase is then used as the fitness signal for the optimizer. The entire procedure of computing the fitness for an individual is sketched-out in Fig. 2.

To measure the test phase performance and the fitness we computed the Normalized Root Mean Square Error (NRMSE). Given a target sequence $\mathbf{z}$ and the generated output sequence $\mathbf{y}$ it

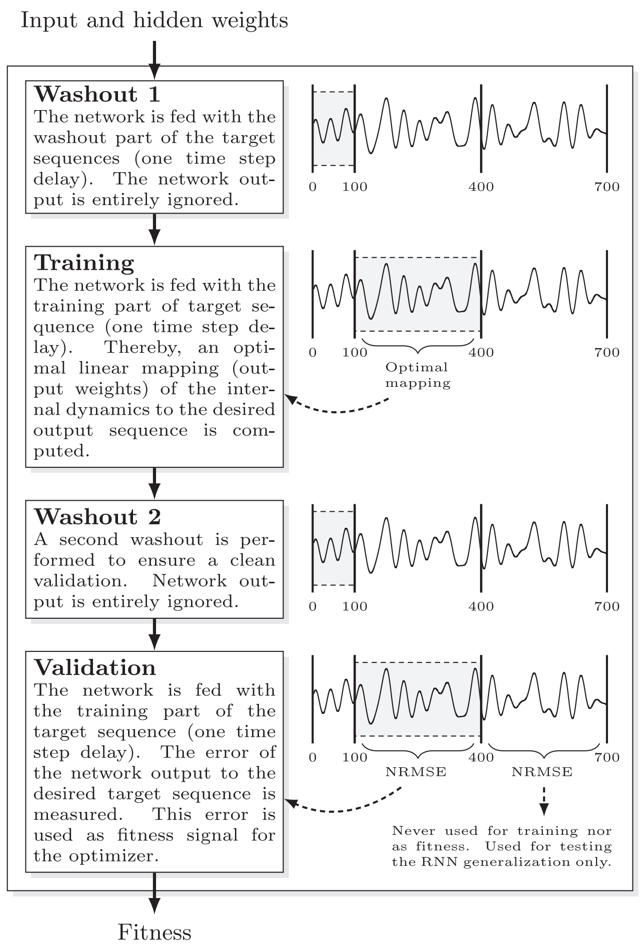
Input and hidden weights
Washout 1
The network is fed with the washout part of the target sequences (one time step delay). The network output is entirely ignored.
Training
The network is fed with the training part of target sequence (one time step delay). Thereby, an optimal linear mapping (output weights) of the internal dynamics to the desired output sequence is computed.
Optimal mapping
Washout 2
A second washout is performed to ensure a clean validation. Network output is entirely ignored.
Validation
The network is fed with the training part of the target sequence (one time step delay). The error of the network output to the desired target sequence is measured. This error is used as fitness signal for the optimizer.
NRMSE
NRMSE
Never used for training nor as fitness. Used for testing the RNN generalization only.
0
100
400
700
Fitness

Fig. 2. Illustration of the fitness computation process of our method exemplarily shown for the MSO benchmark. When the termination criterion of the optimizer is met (maximum number of generations, target error), that reservoir is taken that achieved the best fitness so far, i.e., the lowest NRMSE during the validation (the error measured on the training phase, after the output mapping was computed).

is calculated by

$$NRMSE(\mathbf{y},\mathbf{z}) = \sqrt{\frac{1}{T\sigma_z^2}\left[\sum_{1 \le t \le T} (z^t - y^t)^2\right]}, \tag{6}$$

where σ_z^2 is the variance of the target signal.

For training initialization, we use the setup, which we already used in [5]. The DE population was initialized with values uniformly distributed in default interval $[-0.1, 0.1]$. We also used $F=0.2$, $CR=0.4$ and $NP=5$, whereas the optimization was aborted after 100 generations. All experiments were performed with the JANNLab Toolkit [16].

4.1. Qualitative results and comparison

In this section the qualitative results of selected RNNs [5] are summarized and compared with results previously discussed in the literature. Table 1 contains the error achieved with our system for various MSO instances (averaged over 10 runs) as well as previously reported results in the literature. For each instance we used an RNN with $5n$ hidden neurons (n gives the number of frequencies) with hyperbolic tangent activation. The output feedback was scaled with 10^{-12}. The results show very clearly that the RNNs in our setup significantly outperform the previous methods on all listed dynamics. For example, on MSO_8 we achieved an NRMSE of $6.14 \cdot 10^{-8}$ compared to $2.73 \cdot 10^{-4}$ achieved with a

Table 1
Prediction error (NRMSE) on test phase for various MSO instances in comparison with previous results (the number of units is given in parentheses).

Dynamics	This paper	Koryakin et al. [9]	Roeschies et al. [10]	Holzmann et al. [11]	Schmidhuber et al. [8]
MSO_5	$4.16 \cdot 10^{-10}$ (25)	$1.06 \cdot 10^{-6}$ (11)	$2.54 \cdot 10^{-2}$ (25)	$\approx 8 \cdot 10^{-5}$ (100)	$1.66 \cdot 10^{-1}$ (80)
MSO_6	$9.12 \cdot 10^{-9}$ (30)	$8.43 \cdot 10^{-5}$ (14)	–	–	–
MSO_7	$2.39 \cdot 10^{-8}$ (35)	$1.01 \cdot 10^{-4}$ (18)	–	–	–
MSO_8	$6.14 \cdot 10^{-8}$ (40)	$2.73 \cdot 10^{-4}$ (68)	$4.96 \cdot 10^{-3}$ (25)	–	–
MSO_9	$1.11 \cdot 10^{-7}$ (45)	–	–	–	–
MSO_{10}	$1.12 \cdot 10^{-7}$ (50)	–	–	–	–
MSO_{11}	$1.22 \cdot 10^{-7}$ (55)	–	–	–	–
MSO_{12}	$1.73 \cdot 10^{-7}$ (60)	–	–	–	–

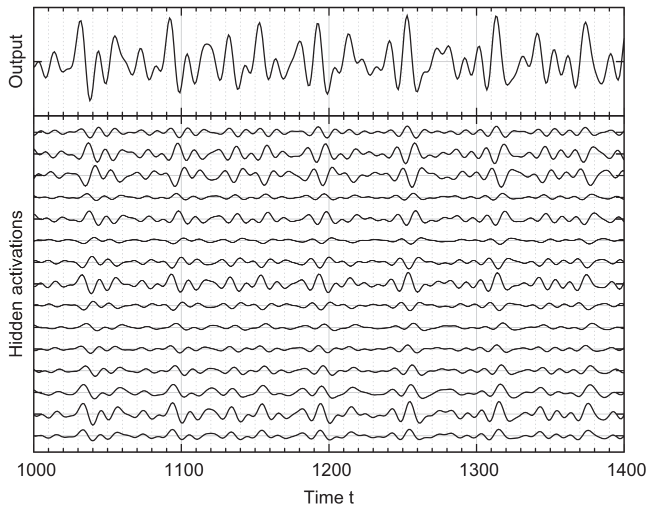

Fig. 3. Visualization of the output signal (upper curve) and the hidden dynamics in an exemplary RNN with 15 hidden neurons that learned MSO_5 after oscillating for 1000 time steps. The output signal has a variance of ≈ 2.5 the internal dynamics have a variance of $\approx 3 \times 10^{-25}$.

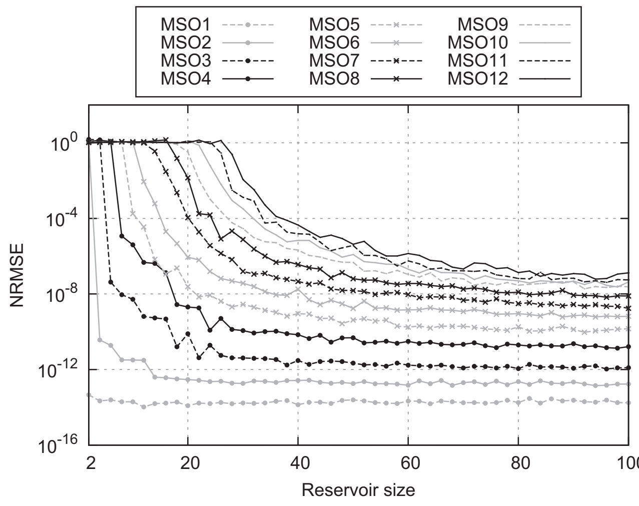

Fig. 4. Relationship between reservoir size and network performance. Each curve shows the mean error rate achieved with different reservoir sizes for all twelve MSOs during the test phase.

balanced ESN. In fact, the NRMSE we obtained on MSO_{12} is still much better than the best NRMSE for MSO_5 reported so far. Note that good error rates ($< 10^{-5}$) are reached even after only a few generations ($G \approx 20$). Due to this only 1–2 s (all experiments were performed using Java™ 1.7 (64-bit) on an Intel® Core™ i7 at 2.3 GHz with 16 GB memory) are required to learn MSO_{12}. When plotting the network output against ground truth, there is still no visual difference noticeable even after hundreds of thousands of time steps (verified on MSO_{12}), although the network was stimulated with the target signal only during the washout phase for this verification.

To get an idea of the interplay of the hidden neurons generating the output signal, Fig. 3 gives insight into an exemplary RNN with 15 hidden neurons that learned MSO_5. Clearly, individual groups of neurons do not reflect particular frequencies and amplitudes, but the output signal is intertwined in the 15 hidden neurons.

Furthermore, we made several other observations that are briefly outlined below. Using the network output as feedback signal during the training phase results in more stable networks concerning long-term generalization. With this method training and test error do not differ as much (only 1–2 magnitudes) as with teacher forced input (2–4 magnitudes). Note that our setup also works with linear hidden units. Hereby, the relatively small scale of the output-feedback (10^{-12}) is not required and can be omitted. We investigate this issue in more detail in Section 4.3.

With a higher number of hidden neurons ($> 10n$) very often networks occur that perform well without any further optimization besides the least squares optimization of the output weights. Their performance is not as high as with DE optimization (3–4 magnitudes lower), but it is noticeably better than ESNs on the MSO benchmark so far. For instance, at a reservoir size of $10n$ and weights initialized from $[-0.1, 0.1]$ the best randomly chosen RNN out of 1000 yielded a NRMSE of 3.25×10^{-6} on MSO_8 and a NRMSE of 9.78×10^{-5} on MSO_{12}, respectively. On the other hand, using the same reservoir size as with DE ($5n$), the best randomly chosen RNN out of 1000 yielded a poor NRMSE of 1.04 in the test phase on MSO_{12}. This confirms the optimization performance in our setup.

4.2. *Reservoir size*

In this experiment we investigate the dependence of the achieved test NRMSE on the reservoir size for all twelve MSO instances. Specifically, this investigation addresses two aspects: first, the minimum required reservoir size to effectively learn a particular MSO instance; and second, whether there is a point at which the reservoir becomes too large, leading to the multitude of internal dynamics affecting the overall performance. To investigate this, we trained networks with increasing hidden layer size on all MSO instances. For each single experiment, the mean NRMSE out of ten repetitions was taken. The achieved error rates are summarized in Fig. 4.

As mentioned above, in the original study [5] networks with a hidden layer size of $5n$ (n is the number of frequencies) were used to consistently produce good results. The curves in Fig. 4 show that each particular MSO instance can be learned well starting at a reservoir size of roughly $2.5n$. At $5n$ the error is relatively small and does not further decrease significantly with more hidden units. We therefore kept the hidden layer size at $5n$ for the subsequent experiments. Our results confirm the observations concerning the minimal reservoir size reported [9] for balanced ESNs, where 60 neurons was proposed as good size for well performing reservoirs to handle MSO_8.

Furthermore, it seems that there is no upper limit for well-performing reservoirs, at least in the covered range. This is particularly interesting because when using ESNs without reservoir optimization, reservoirs that are too large on average effectively provided a reduced quality as reported in [9]. It appears that this problem is solved by optimization, which is able to remove superfluous internal dynamics.

It should be mentioned that the weight initialization may also play a role in the effective learning capability, but we argue that this effect is less important for neurally evolved reservoirs, since disturbing connections can, in principle, be suppressed by the optimizer – the optimizer can tune the hidden weights and thus also “disable” connections, if it was experienced as beneficial for minimizing the fitness signal. Nonetheless, this issue should be further studied and is left for future research.

4.3. *Feedback scale and linearity*

One aspect of the reservoirs capable of learning difficult MSO instances is that they require a relatively low output feedback scale. This important information was originally discovered in [9] and led to the particular choice of 10^{-12} as the feedback scale used for DE optimized reservoirs in [5]. However, whether there are other scale ranges for which good networks can be found with the optimizer was not studied. Hence, the aim in the following experiment is to reveal output feedback scale dependencies.

As pointed out in [17] the small feedback effectively limits the covered domain of the activation function to an interval in which it behaves nearly linearly. In addition, as mentioned in [5] also explicitly linear reservoirs (with identity activation function instead of hyperbolic tangent) can learn difficult MSOs. We therefore include such a linear architecture in the experiment. For both architectures we investigated a broad range of scale values from 10^2 down to 10^{-22}. For each value we took the test error of the best network out of ten runs, where each run comprises 100 generations of optimization. The achieved results are depicted in Fig. 5.

Compared to pure ESNs [9], the ranges in which good reservoirs can be found are noticeably broader when using reservoir optimization. However, it is interesting that there is a small range at a scale of around 10^{-4} where quite good networks are found, followed by a range at around 10^{-5} in which no good network can be generated, until finally the scale is small enough such that from then on good networks are reliably found. This phenomenon also appears with regular ESNs and seems not immediately compensable due to reservoir optimization. It was argued that lowly scaled networks are linear, while the highly scaled networks could provide non-linearity. The former observation can also be confirmed due to the fact that the performance of the linear architecture is nearly identical to the one as achieved with lowly scaled networks. The performance of the linear network is almost independent from the output feedback scale, since it achieved roughly the same performance for all tested scale values.

Based on these findings we chose three different networks for the remaining experiments. The first network (RNN_1) has a feedback scale of 10^{-4}, whereas the second network (RNN_2) has a feedback scale of 10^{-12}, like the networks in [5]. For both networks a hyperbolic tangent is used as hidden activation. The third network (RNN_3) has a feedback scale of 1 and is explicitly linear. Each network is the best one (smallest prediction error rate within the test phase after training) found within 100 runs of training over 100 generations on MSO_{12}. The error rates are 2.68×10^{-4} for RNN_1, 8.06×10^{-8} for RNN_2, and 7.81×10^{-8} for RNN_3.

The mentioned results underline the importance of the feedback scale for the MSO benchmark. Hereby, the interplay of the output and the internal dynamics seem to be a fine tuned system that might be sensitive to outer influences. To investigate this in more detail we did an experiment, where a trained network is taken and a regular washout was performed. Within the washout phase the network is driven towards a stable linear attractor. After washout, however, we slightly modulated the scale of the output (the output scale factor is multiplied by another factor) of the network in each time step before it was fed back into the network. Fig. 6 reveals how the error of the reference networks changes for different modulation rates. An exemplary plot that illustrates the behavior under a moderate output feedback modulation is given in Fig. 7.

It can be observed that all networks behave relatively sensitive to even very small modulation rates. Thereby, it makes no difference whether the modulation rate is amplifying (>1) or

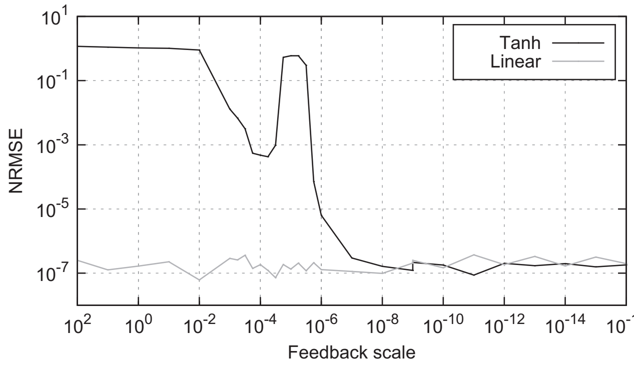

Fig. 5. Effect of feedback scale on network performance comparing RNNs with hyperbolic tangent activation and RNNs with linear activation on the MSO_{12} dynamics. The curves show the NRMSE measured in the test phase.

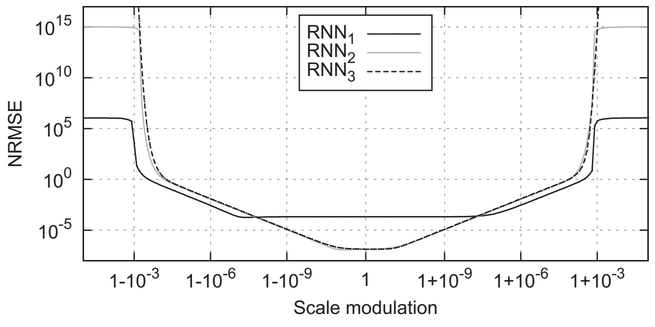

Fig. 6. Error rates of the three reference networks measured within 600 time steps after washout. During the evaluations the output feedback was modulated by a certain factor. The curve of RNN_2 almost covered by the curve of RNN_3.

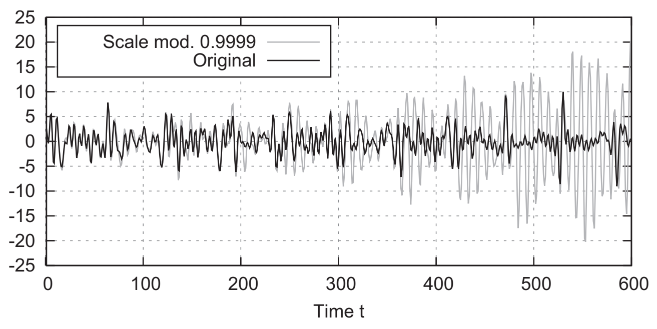

Fig. 7. Exemplary demonstration of the sensitivity regarding the output feedback scale modulation of RNN_2. Only 100–200 time steps after washout the dynamic destabilizes and the output signal explodes.

attenuating (<1) – the error increases equally. However, RNN$_1$ tolerates a broader range of modulation rates until its error increases. RNN$_2$ and RNN$_3$ behave similarly, except for higher modulation rates, where for RNN$_2$ the saturation of the non-linear activation function is reached such that the error explosion is limited.

It is particularly interesting that an attenuating modulation rate (as shown in Fig. 7) also leads to an exploding output signal, since, due to the loss of amplitude, one would expect a vanishing output signal. Nonetheless, the modulation destabilizes the finely tuned internal balance of the oscillating reservoir.

In sum, the results show that while the optimized networks are very well-tuned to the dynamics they were trained on, they are not very robust against signal distortions. However, note that they also were not trained on developing robustness against distortions.

4.4. Importance of washout length

The next investigation concerns the influence of the washout length. We examined how trained networks behave regarding the resulting output error, when the washout phase is shortened and we studied the effect that a shortened washout phase during training has on the resulting performance. For each network the error rate of 600 time steps after washout with different washout lengths was measured. The measurements were done independently, with each particular washout length due to resetting the network state beforehand. The results are depicted in Fig. 8.

Let us first consider the results for regularly trained RNNs. As can be seen, the networks use the full washout length of 100 time steps to reach their final error rates. However, beyond the 100 time steps there is no further improvement. This holds true for all three networks. RNN$_2$ and RNN$_3$ show a similar behavior, while RNN$_2$ seems to work slightly better for almost every washout length.

When looking at the three other curves (denoted with sw) we can observe that a shorted washout length during training leads to a faster adaption to the target signal. However, the final performance is markedly inferior in comparison with the regularly trained networks – the length of the washout phase thus imposes a limit on the final performance.

4.5. Robustness to noise

Previously, we have seen that an output feedback modulation, even with a small rate, causes the internal dynamics of the networks to crash. The experiment in this section addresses the robustness to noise. Our investigation therefore includes two

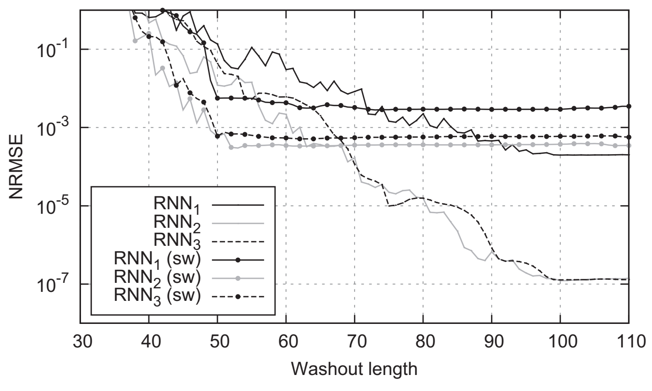

Fig. 8. Output error of 600 time steps after washout with varying length produced by the three reference networks as well as by three networks (denoted with sw) explicitly trained using short washout phase of 50 time steps.

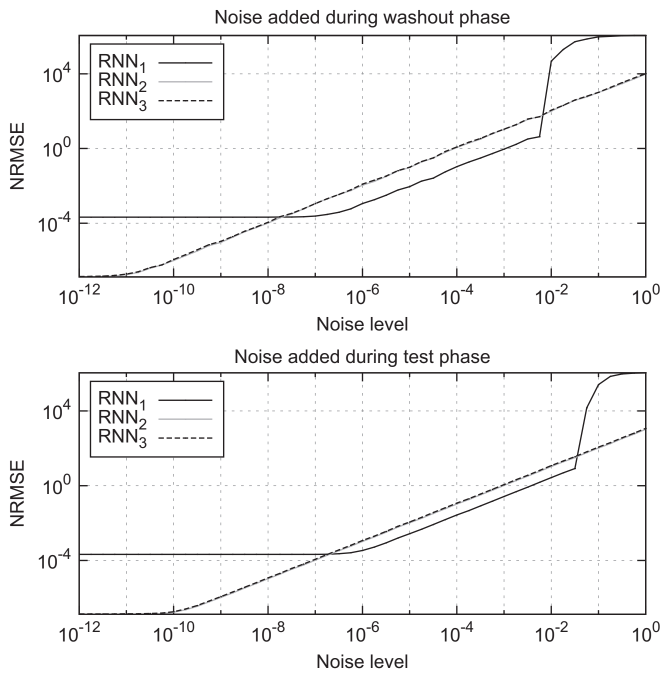

Fig. 9. Illustration of the effect of noise added to the feedback signal. The curves show the error rates over 600 time steps after washout for different noise levels. For the upper diagram noise was added only during the washout phase, whereas for the lower diagram noise was added during the evaluation phase after a noise-free washout. Note that the gray curves of RNN$_2$ overlay with the curves of RNN$_3$, since they behave equally in both cases.

procedures. First, noise is added only during the washout phase and we looked at whether the networks were still able to adapt to the input signal. Second, the washout is performed as usual without noise, and noise is then added to the output feedback during the evaluation phase. For this experiment the error rates of the subsequent 600 time steps after the washout were measured. Fig. 9 summarizes the achieved results.

The upper diagram in the figure shows the evaluated error of the three RNNs when noise is added in the washout phase only. RNN$_2$ and RNN$_3$ behave nearly identically here, hence the curves are not distinguishable. As can be seen, their performance decreases relatively early, even for small noise levels. The error grows linearly with the noise level. RNN$_1$, as previously reported in other experiments, shows a higher stability. It tolerates a significantly higher amount of noise than the other networks until its performance also decreases. It is noteworthy that for the noise range of roughly 10^{-8}–10^{-2} the measured error is smaller than the error of the two other networks. However, at a noise level of approximately 10^{-2} and beyond, RNN$_1$ breaks out completely from its learned attractor and the output signal explodes until saturation of the activation function is reached.

When the washout phase is performed regularly noise-free, but noise is added to the feedback signal during the evaluation, the behavior for all networks looks similar in principle, as can be observed in the lower diagram of Fig. 9. Nevertheless, it appears that all RNNs are more robust to noise, once they are completely tuned to the target signal.

4.6. Varying phase-shifts and amplitudes

One central question of this research is how the networks behave when fed with unknown signals. We know that the optimized reservoirs achieve very small error rates even on difficult MSOs, but it was not clear, whether they were able to perform on varied MSO instances, and if so, whether they could then also yield comparably small error rates. In order to investigate this issue, we performed two kinds of experiments. In the first experiment, which is presented in this section, we varied phase-shifts φ_i and amplitudes a_i of the sinusoidal waves, and we also dropped some of them randomly. The second experiment, in which the frequencies f_i are varied, is discussed in the subsequent section.

In the experimental procedure we tested each of the pre-trained networks on randomly generated MSOs with a certain number of frequencies (1–12). The amplitudes were drawn from the interval $[0.1, 1]$ and the phase-shifts were drawn on the interval $[0, 2\pi]$. The NRMSE for the optimized networks was calculated for the 600 time steps after the washout phase of 100 time steps. Only during the washout phase, the true signal was presented to the optimized network. The error rates of each single experiment given in Table 2 are average values over ten repetitions.

As can be seen, all three networks behave similarly in comparison to their achieved performance on the original MSO_{12}. Interestingly, the networks were able to predict the modified sinusoidal signals with nearly the same precision achieved on the trained dynamics. The error on less than twelve frequencies is even slightly smaller.

To verify these results visually, we recorded the outputs of our reference networks on three concatenated exemplary signals (the original MSO_{12}, and two random MSOs with nine or four waves, respectively), which are shown in Fig. 10. We altered between washout (signal presentation) and prediction every 100 time steps. The output of the networks is produced continuously and the activations are not set to zero upon a signal change. It can be seen that the networks are able to adapt very quickly to the novel signal and can predict it perfectly after washout. Varied amplitudes, phase-shifts, and entirely dropped frequencies can thus be compensated without any performance loss. It should be mentioned that the predictions are still highly accurate even after 10,000 or more time steps. When looking at the washout phase more closely, RNN_2 appears to adapt a bit faster than the two other networks (cf. Fig. 10). Remember that we did not reset the internal activations of the networks, so during the washout phase the previous dynamics have to be overwritten in the reservoir, which is apparently accomplished sufficiently well in the allotted 100 time steps.

The results thus indicate that the internal RNN dynamics were optimized towards a linear attractor, which approximates the rotations implied by the sinusoidal target function (the MSO) very well. Interestingly, the individual rotations for the individual sinusoidal functions do not seem to be extractable directly from the internal reservoir dynamics – they are fully intertwined (as already shown in Fig. 3). Nonetheless, it is fascinating to see how the one-dimensional feedback signal during the washout phase is able to very quickly tune the individual MSO components to their current signal strength. This is even more remarkable when reconsidering the sensitivity of the networks to just a minimal modulation of output feedback.

Table 2
NRMSE after washout for MSOs with random amplitudes and phase-shifts.

No. frequencies	RNN_1	RNN_2	RNN_3
12[a]	$2.11 \cdot 10^{-4}$	$1.32 \cdot 10^{-7}$	$1.39 \cdot 10^{-7}$
1	$5.63 \cdot 10^{-5}$	$3.63 \cdot 10^{-8}$	$5.51 \cdot 10^{-8}$
2	$1.30 \cdot 10^{-4}$	$4.11 \cdot 10^{-8}$	$4.79 \cdot 10^{-8}$
3	$1.59 \cdot 10^{-4}$	$5.41 \cdot 10^{-8}$	$6.53 \cdot 10^{-8}$
4	$1.50 \cdot 10^{-4}$	$1.14 \cdot 10^{-7}$	$7.26 \cdot 10^{-8}$
5	$2.21 \cdot 10^{-4}$	$1.26 \cdot 10^{-7}$	$7.68 \cdot 10^{-8}$
6	$4.44 \cdot 10^{-4}$	$1.58 \cdot 10^{-7}$	$1.01 \cdot 10^{-7}$
7	$3.70 \cdot 10^{-4}$	$1.32 \cdot 10^{-7}$	$1.02 \cdot 10^{-7}$
8	$4.43 \cdot 10^{-4}$	$1.38 \cdot 10^{-7}$	$1.27 \cdot 10^{-7}$
9	$4.87 \cdot 10^{-4}$	$1.37 \cdot 10^{-7}$	$1.31 \cdot 10^{-7}$
10	$4.65 \cdot 10^{-4}$	$1.35 \cdot 10^{-7}$	$1.19 \cdot 10^{-7}$
11	$4.19 \cdot 10^{-4}$	$1.31 \cdot 10^{-7}$	$1.30 \cdot 10^{-7}$
12	$3.91 \cdot 10^{-4}$	$1.15 \cdot 10^{-7}$	$1.20 \cdot 10^{-7}$

[a] Original MSO12 without variations.

4.7. Varying frequencies

When we first experimented with varying frequencies we discovered that shortly after the washout phase the network output digresses from the target-signal and the resulting error rates were as high as the error of a random signal with a signal

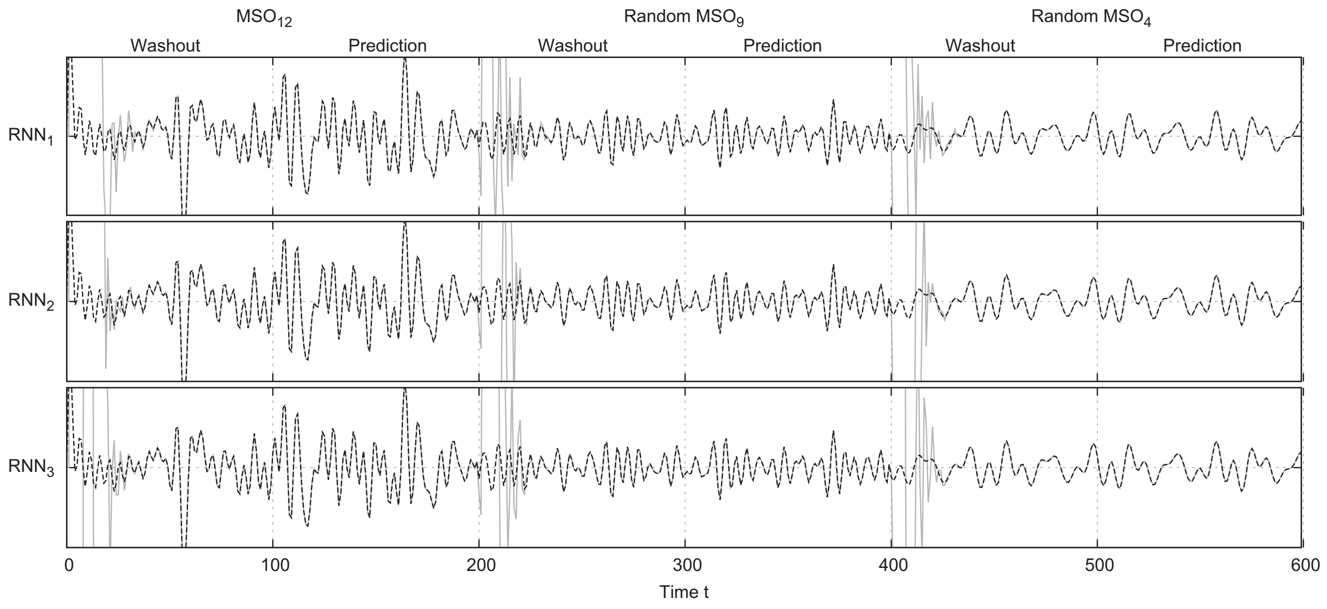

Fig. 10. The adaption of evolutionary optimized RNNs trained on the standard MSO_{12} to unknown varied MSO instances with random phase-shifts and amplitudes. The dashed black lines represent the ground-truth signals, which are injected during the washout phases. The gray lines show the particular network outputs. For all three networks both curves overlay after short period of time.

variance equal to the target signal (the related error rates are given in Table 3). The networks replayed something, which was not the target signal, but which at least seemed to be a stably generated signal. The output did not vanish or explode over time. Since it was difficult to interpret the outputs in the time domain, we looked into the frequency domain.

For the conducted experiments, we took the output measured on an MSO_{12} with randomly chosen frequencies from the interval $[0.2, 1.32]$. The amplitudes and phase-shifts were not varied. The overall sequence had a length of 600 times steps. The first 300 times steps were used for washout, to match the window length of 300 which we used to produce an acceptably accurate frequency spectrum using *Fast Fourier Transformation* (FFT). For each network we generated four spectra on the output signal with a shifted time window, namely 0–299, 100–399, 200–499, and 300–599. Thus, the first window covers the entire washout phase and the last window covers a pure prediction phase.

The spectra depicted in Fig. 11 reveal an interesting behavior. During the washout phase the frequencies of the target signal are clearly recognizable as peaks in the spectra, but as soon as the networks are driven by their own output they fall back to a signal clearly composed of frequencies of the originally learned dynamics. The three reference networks here differ slightly in how strongly the original frequencies were amplified. The spectra of RNN_1 and RNN_2 represent the "density" of the frequency distribution of the target-signal somewhat better, while the amplitudes of RNN_3 become more equally distributed. The outputs of RNN_1 and RNN_2 can thus be interpreted as better representations of the target signal.

Table 3
Effect of remapping output for varied frequencies.

Dynamics	RNN_1	RNN_2	RNN_3
Original	$2.65 \cdot 10^{-4}$	$8.06 \cdot 10^{-8}$	$7.81 \cdot 10^{-8}$
Varied	1.42	1.39	1.44
Varied rm. (mean)	0.14	$2.08 \cdot 10^{13}$	$1.14 \cdot 10^{17}$
Varied rm. (min)	$4.50 \cdot 10^{-3}$	$6.05 \cdot 10^{-4}$	$5.67 \cdot 10^{-4}$

To analyze the systematicity in such representations and its dependency on the RNN structure, we furthermore attempted to discover the output spectra related to a single wave with a certain frequency. We stimulated each of the three trained reference networks with frequencies covering the band $[0, 1.4]$ (step size of 2.5×10^{-3}, resulting in 561 frequencies). For each particular wave we computed the output frequency spectrum of 1000 time steps after washout (we used 1000 time steps to yield more accurate spectra). The results are depicted in Fig. 12.

The illustrations reveal a general effect that holds for all three networks almost equally. Whenever a stimulating frequency matches one of the previously learned frequencies all others are silenced, which is consistent with the observation that arbitrary frequencies can be dropped, as discussed in the previous section. However, when the stimulating frequency that lies within the frequency band of MSO_{12} does not exactly match one of the learned frequencies, it decomposes into all learned spectral components, whereby the amplitude of an activated spectral component decreases in proportion with the distance to the stimulating frequencies. Thus, these two components mostly appear in the output frequency spectrum that are closest to the stimulus. On the other hand, when the input frequency is outside of the trained frequency band it decomposes less systematically. With increasing distance from the familiar frequency it even causes the internal dynamics to collapse and the output signal to again explode. In this case RNN_1 seems to be the most stable one, while RNN_3 behaves the most sensitively.

Due to the previous experiments we know that the trained networks are highly specialized to a concrete composition of frequencies. In the following we answer the question of whether it is possible to remap the trained reservoir to a new output dynamic such that a different spectral composition is stably produced. An MSO with twelve random frequencies was selected. With the regular training procedure the output weights of the reference networks were recomputed to optimally match the varied target signal. Then, after a new washout, the error was measured for the test phase.

The achieved error rates (mean values as well as the minimum error over ten repetitions) are listed in Table 3. In the first row the

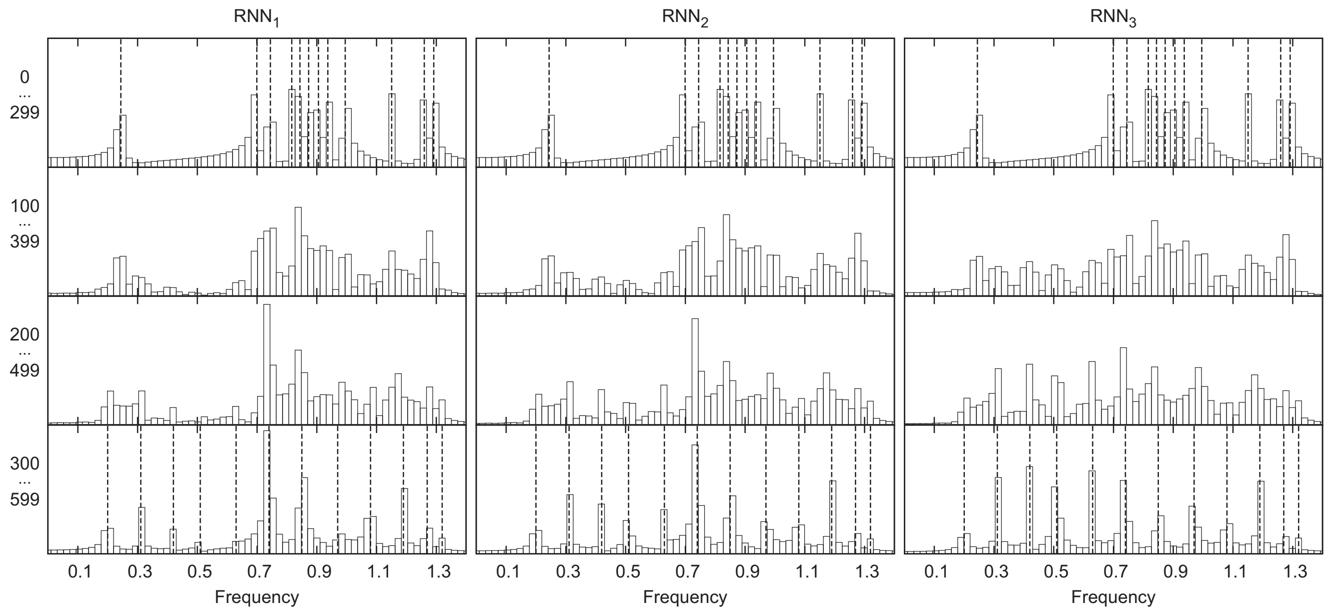

Fig. 11. Changing of the output frequency-spectrum for trained networks fed with an MSO signal with varied frequencies. The upper row covers the time window from 0 to 299 (washout only), the second row the time window from 100 to 399, the third row the time window from 200 to 499, and the last row the time window from 300 to 599 (prediction only). The dotted lines in the first row indicate the frequencies of the target signal, while the dotted lines in the last row indicate the frequencies of the original MSO_{12}.

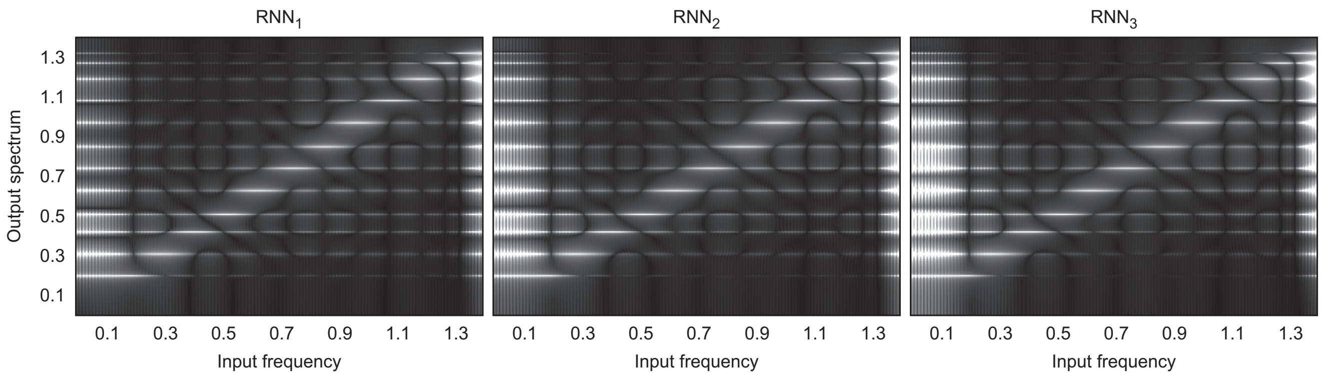

Fig. 12. Illustration of the output frequency spectra produced by the reference networks stimulated with single waves. Each spectrum (vertical axis) is computed based on 1000 time steps after washout of the network's response, whereby the networks have been stimulated with a certain frequency (horizontal axis) beforehand. This shows how each frequency of the input signal decomposes into the learned spectral components and exposes the systematicity behind the frequency-fallback discovered in Fig. 11.

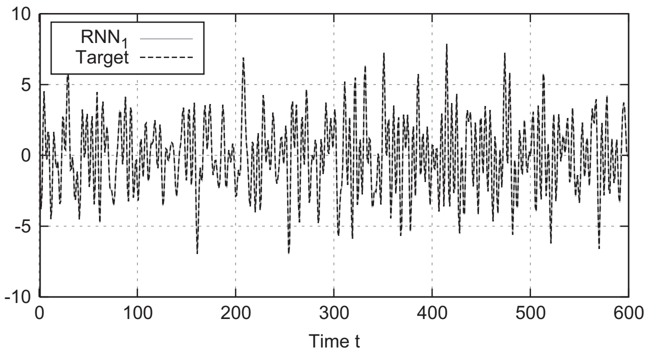

Fig. 13. Exemplary output over 600 time steps after washout of RNN_1 remapped to a new frequency composition.

error rates regarding the original dynamics are given. The second row shows the error rates on varied frequencies (mean) with no remapping. The last two rows contain the error rates after remapping (mean error and minimum error respectively). The results show that, at least for RNN_2 and RNN_3, remapping to varied frequency compositions seems to be on average more difficult, while it seems to be easier for RNN_1. Nonetheless, in some cases the remapping works for all three networks, as can be seen from the minimum error achieved. Obviously, there are frequency compositions which are better suited to being remapped by the networks than others. For instance, in RNN_1 it was possible to remap its internal dynamics to a signal consisting of the frequencies (0.21, 0.308, 0.353, 0.494, 0.599, 0.615, 0.880, 1.078, 1.151, 1.271, 1.273, 1.284) with a test error of $4.50 \cdot 10^{-3}$.

For visual verification of a successful remapping, Fig. 13 depicts the output of RNN_1 after washout. As can be seen, the output matches the target sequence quite well. Interestingly, RNN_2 could not be remapped to the same dynamics as accurately, since a test error of only 0.014 was achieved. Conversely, the situation is similar. Using the dynamics for which RNN_2 produced its minimum error of $6.05 \cdot 10^{-4}$, RNN_1 achieved an error of only 0.017. However, the results indicate that the reservoirs (including the input weights or the output feedback weights respectively) have been tuned by the DE optimization in such a way that they are highly specialized, replaying exactly one particular frequency composition and remapping disturbs the sensitive internal balance.

5. Beyond linear dynamics

The major part of this paper investigated the performance of DE-optimized RNNs on the MSO benchmark, showing that

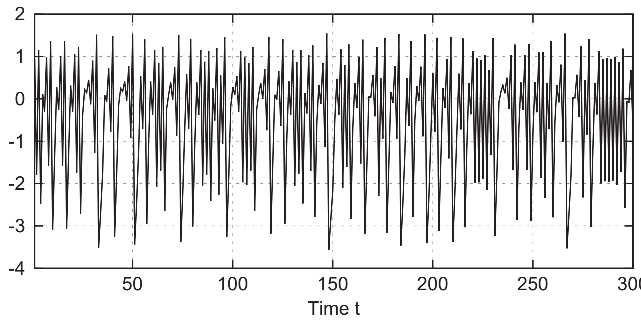

Fig. 14. The first 300 time steps of the non-linear Hénon attractor that exhibits chaotic behavior.

excellent linear attractors evolved. Dynamic real-world problems, however, are often highly non-linear and are thus potentially more difficult to learn. Therefore, the remainder of this work addresses learning non-linear dynamics. As a first investigation, we evaluated the performance of our method on learning the well-studied *Hénon attractor* [18]. The Hénon attractor, or Hénon map, is a discrete-time non-linear dynamic system, which generates chaotic behavior. Thus, a recurrent neural system needs to produce unique state – next state correlations and thus effective, independent component frequencies.

In our experiments we used the one-dimensional formulation of the Hénon attractor with delay coordinates as given in [19] by the equation

$$y(t) = 1 - 1.4y(t-1)^2 + 0.3y(t-2). \tag{7}$$

Here $y(t)$ represents the system output at time t and we assume $y(t) = 0$ for $t < 1$. An example trajectory of the Hénon attractor is depicted in Fig. 14.

To produce comparable results we used the same configuration as in [20]. The learning task is to predict the next value $y(t+1)$ of the time series at any time step t based on the previously given values $\tilde{y}(t) = y(t) + \xi(t)$, where $\xi(t)$ is normally distributed white noise with standard deviation 0.05. Note that the original signal $y(t)$ is additionally shifted by -0.5 and scaled with 2 according to [20]. The first 2000 time steps are used for training, which in our case means computing the optimal output mapping using Eq. (1). The next 3000 time steps are used for validation. In this phase, the fitness signal for the optimizer is generated. Finally, the subsequent 3000 time steps are the test-phase. The first 200 time steps of each phase are used for washout. To enable performance comparison with Rodan et al. [16], we use the Normalized Mean Square Error (NMSE) as the

Table 4
Prediction error (NMSE) for the Hénon attractor.

Reservoir size	This paper	Rodan et al. [20]
50	$4.23 \cdot 10^{-3}$	$9.75 \cdot 10^{-3}$
100	$4.31 \cdot 10^{-3}$	$8.94 \cdot 10^{-3}$
150	$5.53 \cdot 10^{-3}$	$8.71 \cdot 10^{-3}$
200	$7.96 \cdot 10^{-3}$	$8.68 \cdot 10^{-3}$

error metric, which is given by

$$NMSE(\mathbf{x},\mathbf{z}) = \frac{1}{T\sigma_z^2}\left[\sum_{1 \le t \le T} (z^t - x^t)^2\right]. \tag{8}$$

The DE setup was similar to that of the MSO benchmark. We used $CR=0.6$ and $F=0.4$, since these parameters worked better here (with the previous parameter the error did not converge below 0.01 frequently). In each single experiment the training was performed over 1000 generations. The achieved results (again, the average error rates are over 10 repetitions) are listed in Table 4. In [20] different ESNs, i.e., those with special topologies were investigated. However, on the Hénon map classic ESNs with forced echo state property were reported to achieve the best results.

As can be seen, the error rates are significantly smaller than those presented in [20] for classic ESNs. It should be mentioned that without optimization our reservoirs yielded test NMSE rates of $\approx 3.60 \cdot 10^{-2}$, which are, in contrast, considerably higher than the results in [20]. This again demonstrates the advantage of the optimization and that even non-linear dynamics, such as the Hénon attractor, can be learned effectively with our method by simple fully-connected reservoir networks without any manual fine-tuning.

6. Conclusion

In this paper we comprehensively discussed an approach that enables standard Recurrent Neural Networks (RNNs) without any specialized structural optimization or topologically dependent fine-tuning to learn dynamics very efficiently and precisely using a variant of Differential Evolution (DE) equipped with a modified mutation scheme. An interesting aspect of the method is that the DE optimizer requires very few individuals, namely five, throughout all of our experiments.

In our system, training the hidden weights is done via DE, whereas the output weights are determined using a least squares solution. Using these methods in combination, we achieved the best known results so far for various instances of the common Multiple Superimposed Oscillator (MSO) benchmark with up to twelve waves (more waves are also possible). In fact, our results are order of magnitude better than previously reported values.

Moreover, we presented several novel research findings regarding the proposed method and the optimized reservoirs networks. This covers investigating the influence of the reservoir size and output feedback scale on the quality of the networks that can be found with the optimization, and the behavior of trained networks on varied MSO instances, namely, randomly chosen amplitudes, phase shifts, and frequencies. We showed that the networks are very sensitive regarding a modulation of the feedback, causing the networks to very quickly lose their learned attractor. We studied the influence of the length of the washout phase and how robust the trained RNNs are to noise. We learned that when stimulating an optimized RNN with a single wave of a certain frequency, which does not match one of the learned spectral components, the signal decomposes mainly into the closest frequencies of the learned spectrum. It was also shown that recomputing the output weights does not allow the handling of varied frequencies with optimized, linearly behaving networks.

Finally, our approach was applied to learn non-linear dynamics. In an experiment with the chaotic Hénon Attractor, we showed that optimized reservoir networks are able to predict the signal of the non-linear target dynamics even more accurately than reported for several ESN variants in the literature. However, analyzing the networks on this and other non-linear benchmarks is an important next research investigation.

As a last remark, it should be mentioned that our approach of independently tuning the reservoir weights with DE can most likely be used in a similar manner to improve *Extreme Learning Machines* (ELMs) [21] as well, since they are the feed-forward counterpart of ESNs and, thus, should profit comparably.

References

[1] S. Dasgupta, D. Goldschmidt, F. Wrgtter, P. Manoonpong, Distributed recurrent neural forward models with synaptic adaptation and CPG-based control for complex behaviors of walking robots, Front. Neurorobot. (2015) 10, http://dx.doi.org/10.3389/fnbot.2015.00010.
[2] B.G. Woolley, K.O. Stanley, Evolving a single scalable controller for an octopus arm with a variable number of segments, In: Proceedings of the 11th International Conference on Parallel Problem Solving from Nature: Part II, PPSN'10, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 270–279.
[3] V. Mnih, N. Heess, A. Graves, K. Kavukcuoglu, Recurrent models of visual attention, In: Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, K.Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 27, Curran Associates, Inc., Montreal, Canada, 2014, pp. 2204–2212.
[4] A. Graves, Generating sequences with recurrent neural networks, arXiv:1308.0850 [cs].
[5] S. Otte, F. Becker, M.V. Butz, M. Liwicki, A. Zell, Learning recurrent dynamics with differential evolution, In: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), Bruges, Belgium, 2015.
[6] R. Storn, K. Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359.
[7] H. Jaeger, The "echo state" approach to analysing and training recurrent neural networks, Technical Report GMD Report, 148, Fraunhofer Institute for Analysis and Information Systems AIS, Sankt Augustin, Germany, 2001.
[8] J. Schmidhuber, D. Wierstra, M. Gagliolo, F. Gomez, Training recurrent neural networks by Evolino, Neural Comput. 19 (2007) 757–779.
[9] D. Koryakin, J. Lohmann, M.V. Butz, Balanced echo state networks, Neural Netw. 36 (2012) 35–45, http://dx.doi.org/10.1016/j.neunet.2012.08.008.
[10] B. Roeschies, C. Igel, Structure optimization of reservoir networks, Logic J. IGPL 18 (5) (2010) 635–669, http://dx.doi.org/10.1093/jigpal/jzp043.
[11] G. Holzmann, H. Hauser, Echo state networks with filter neurons and a delay & sum readout, Neural Netw. 23 (2) (2010) 244–256, http://dx.doi.org/10.1016/j.neunet.2009.07.004.
[12] Y. Xue, L. Yang, S. Haykin, Decoupled echo state networks with lateral inhibition, Neural Netw. 20 (3) (2007) 365–376, http://dx.doi.org/10.1016/j.neunet.2007.04.014.
[13] I.B. Yildiz, H. Jaeger, S.J. Kiebel, Re-visiting the echo state property, Neural Netw. Off. J. Int. Neural Netw. Soc. 35 (2012) 1–9, http://dx.doi.org/10.1016/j.neunet.2012.07.005.
[14] J. Ilonen, J.-K. Kamarainen, J. Lampinen, Differential evolution training algorithm for feed-forward neural networks, Neural Process. Lett. 17 (1) (2003) 93–105, http://dx.doi.org/10.1023/A:1022995128597.
[15] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Trans. Evol. Comput. 15 (1) (2011) 4–31, http://dx.doi.org/10.1109/TEVC.2010.2059031.
[16] S. Otte, D. Krechel, M. Liwicki, JANNLab neural network framework for Java, In: Poster Proceedings Conference MLDM 2013, ibai-publishing, New York, USA, 2013, pp. 39–46.
[17] D. Koryakin, M.V. Butz, Reservoir sizes and feedback weights interact non-linearly in echo state networks, In: Artificial Neural Networks and Machine Learning - ICANN 2012, vol. 7552, 2012, pp. 499–506. http://dx.doi.org/10.1007/978-3-642-33269-2_63.
[18] M. Hénon, A two-dimensional mapping with a strange attractor, Commun. Math. Phys. 50 (1) (1976) 69–77.

[19] M.W. Slutzky, P. Cvitanovic, D.J. Mogul, Manipulating epileptiform bursting in the rat hippocampus using chaos control and adaptive techniques, IEEE Trans. Biomed. Eng. 50 (5) (2003) 559–570, http://dx.doi.org/10.1109/TBME.2003.810701.
[20] A. Rodan, P. Tino, Minimum complexity echo state network, IEEE Trans. Neural Netw. 22 (1) (2011) 131–144, http://dx.doi.org/10.1109/TNN.2010.2089641.
[21] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, In: 2004 IEEE International Joint Conference on Neural Networks, 2004. Proceedings, vol. 2, 2004, pp. 985–999. http://dx.doi.org/10.1109/IJCNN.2004.1380068.

Sebastian Otte has received his M.Sc. in Computer Science at the RheinMain University of Applied Sciences in Wiesbaden, Germany, in 2012. His master thesis addressed recurrent neural networks and supervised sequence classification. Since 2013 he is a Ph.D. researcher at the Cognitive Systems Group related to the Wilhelm Schickard Institute for Computer Science at the University of Tübingen, Germany. His research focuses on collective and decentralized optimization, recurrent neural networks and sequential pattern recognition.

Martin V. Butz has been a professor in computer science and psychology at the University of Tübingen since 2011. Between 2002 and 2004 he studied at the University of Illinois at Urbana-Champaign, where he received his doctorate degree in computer science. Following his studies, Dr. Butz obtained a post-doc position at the University of Würzburg, where, in 2007, he received an Emmy-Noether grant (a young investigators grant) from the Deutsche Forschungsgemeinschaft (DFG). Dr. Butz has published two monographs, six edited volumes, and more than fifty journal articles. His current research is interdisciplinary, combining aspects from cognitive psychology, cognitive modeling, artificial intelligence, and machine learning. His main research focus lies in neuro-computational modeling of cognitive development, including self-perception, action understanding, social cognition, compositional concept structures, and language.

Danil Koryakin is a Ph.D. student at the department of Cognitive Modeling at the University of Tübingen. He received the Diploma of Engineering in System Analysis and Control from the Siberian Aerospace Academy, Krasnoyarsk, Russia, in 1999. His research won an award from the Frankfurt-Main Airport Foundation. In 2001 he received his Master of Science in Communications Technology at the University of Ulm, Germany. His current research interests focus on artificial neural networks for modeling time series, their modularity and neuro-evolution.

Fabian Becker has received an M.Sc. in Computer Science at the Technische Hochschule Mittelhessen, University of Applied Sciences in Gießen, Germany, in 2012. In 2012 he joined the Cognitive Systems Group at the Wilhelm Schickard Institute for Computer Science at the University of Tübingen, Germany as a Ph.D. researcher. His research focuses on meta optimization with evolutionary algorithms.

Marcus Liwicki received his M.S. degree in Computer Science from the Free University of Berlin, Germany, in 2004, and his Ph.D. degree from the University of Bern, Switzerland, in 2007. Subsequently, he successfully finished his Habiliation and received the postdoctoral lecture qualification from the University of Kaiserslautern, Germany, in 2011, and an associate professorship there in 2014. Currently he is senior assistant in the University of Fribourg (Switzerland) and Associate Professor at the University of Kaiserslautern. His research interests include on-line and off-line handwriting recognition, document analysis, especially for historical documents, knowledge management, semantic desktop and electronic pen-input devices. From October 2009 to March 2010 he visited Kyushu University (Fukuoka, Japan) as a research fellow, supported by the Japanese Society for the Promotion of Science.

Andreas Zell received a university diploma in Computer Science from the University of Kaiserslautern, Germany, in 1986 and a M.S. from Stanford University, USA, in 1987. He obtained his Ph.D. in 1989 and his habilitation in Computer Science in 1994 at the University of Stuttgart, Germany. Since 1995 he is full processor at the Wilhelm Schickard Institute for Computer Science at the University of Tübingen, Germany. He is head of the Cognitive Systems group and also the founding director of the Centre for Bioinformatics Tübingen (ZBIT).