Herbert Jaeger

Generating exponentially many periodic attractors with linearly growing Echo State Networks

Technical Report No. 3
August 2006

School of Engineering and Science

Generating exponentially many periodic attractors with linearly growing Echo State Networks

Herbert Jaeger

International University Bremen
School of Engineering and Science
Campus Ring 6
28759 Bremen
Germany

E-Mail: h.jaeger@iu-bremen.de
http://www.faculty.iu-bremen.de/hjaeger/

Abstract

When a human listens to a novel piece of music, wherein a reasonably short motif (likewise novel to the listener) is repeated twice, the human is able to (1) detect this fact, (2) continue to reproduce the motif periodically. A similar phenomenon is iterated phone number rehearsal to keep a phone number in short-term memory. Mathematically, these are cases where a dynamical system (the brain) hosts a huge number of periodic attractors, into which it can be driven by feeding the corresponding periodic time series as a cue input. Here we demonstrate how this phenomenon can be modelled using Echo State Networks featuring a spatial encoding of musical pitch. The network is trained as a pure delay-line memory. The number of periodic attractors that a given, trained ESN can produce without further parameter adjustment scales exponentially with the network size. A stability analysis of the attractors is provided.

1 Introduction

Humans are capable of detecting in acoustic time series a novel pattern that is repeated twice (or more often) in immediate succession. After detection, it

is immediately available in short term memory and can be stably reproduced periodically. An example is discerning a repeated motif in a piece of music upon first hearing, with subsequent overt, periodic reproduction of the motif by singing (or whistling or humming...). Likewise, when someone tells us a phone number, we can keep it active in short-term memory by mental rehearsal, that is, by repeating the number to ourselves. The reproduction of the motif or phone number is stable, that is, immune to disruption by noise. Because of the short time required for this phenomenon to unfold, synaptic weight changes cannot be involved. This implies that human brains, cast as dynamical systems, host a large number of periodic attractors which can be selected and activated by input signals that are similar to signal produced by the respective attractor.

Here we present a recurrent neural network architecture based on Echo State Networks (ESNs, [1] [4]) which exhibits an abstract version of these phenomena. The data that we use are discrete-time, one-dimensional, random time series that we might consider as "melodies". Standard ESNs are trained as delay-line memories, or "retrodictors", on such signals, that is, they have various output channels each of which corresponds to a discrete delay time k and should output the input from k time steps earlier. The "pitch" of the input and signals is coded spatially, by assigning different input (or output, respectively) units to respond optimally to different pitches. The trained network is augmented by a leaky retrodiction error integration mechanism which enables the system to detect periodicity in its input: if the input signal is periodic with period length k, the error on the retrodiction channel trained on the delay k will go toward zero. An additional voting mechanism monitors the integrated retrodiction errors and feeds back those network outputs to the input that correspond to low-error retrodictions, that is, to the appropriate period length. We show analytically that the periodic re-productions of cue signals is stable. The analysis reveals that the nonlinear nature of the network output units is crucial for stability. Furthermore, the spatial pitch coding, together with the short-term memory properties of ESNs [2], enable the proposed architecture to host a number of periodic attractors that grows exponentially with the network size.

Previous work on periodic attractors in neural networks concerned the training of a network to stably reproduce a periodic teacher signal – in the early days of neural network research this was a frequently used challenge to demonstrate the performance of learning algorithms. Querying Google on "neural network" + "figure eight", for instance, retrieves more than 200 hits, mostly pointing to papers where a 2-dimensional periodic time series whoses plots look like the figure 8 was to be learnt. A rather different venue of research with a rich literature concerns the mathematical analysis of the

(bifurcation) dynamics of small recurrent neural networks, where the typical finding is that even very small networks (2 neurons) exhibit a host of periodic and other attractors *across different weight settings*, see, for instance, [5]. Our present study aims at something quite different and has, as far as we can perceive, no precedent. Namely, we wish to set up (and analyze) a recurrent neural network that hosts a large (even huge) number of periodic attractors *with one given, fixed setting of weights*, such that the network can be driven into any of these attractors by a suitable cue input.

The report is organized as follows. In sections 2 and 3 we rehearse the basic ESN notation and describe the spatial pitch coding, as well as the data scaling required to run the ESN in an appropriate operating regime. The training of ESNs as delay-line memories is detailed out in section 4.3. In section 5 the complete architecture comprising the delay-line ESN and the error monitoring and voting scheme is explained. Its performance is highlighted in section 6. A stability analysis of the network attractors is provided in section 7. In the concluding discussion (section 8) an outlook on further research is made.

All computations were done with Matlab. The code is available online from the author's publication page at `http://www.faculty.iu-bremen.de/hjaeger/pubs.html`. The "EdNote" comments in the report below point to variable names in the code.

2 Basic ESN notation

For a comprehensive introduction to Echo State Networks the reader is referred to the literature [1] [3]. This short section serves only to fix our notation. We consider discrete-time ESN networks with K input units, N internal network units and L output units. Activations of input units at time step n are $\mathbf{u}(n) = (u_1(n), \ldots, u_K(n))$, of internal units are $\mathbf{x}(n) = (x_1(n), \ldots, x_N(n))$, and of output units $\mathbf{y}(n) = (y_1(n), \ldots, y_L(n))$. Real-valued connection weights are collected in a $N \times K$ weight matrix $\mathbf{W}^{\text{in}} = (w^{\text{in}}_{ij})$ for the input weights, in an $N \times N$ matrix $\mathbf{W} = (w_{ij})$ for the internal connections, and in an $L \times (K + N)$ matrix $\mathbf{W}^{\text{out}} = (w^{\text{out}}_{ij})$ for the connections to the output units. Here we do not use backprojections from the output units to the internal units or connections between output units. Note that connections directly from the input to the output units are allowed.

The activation of internal units is updated according to

$$\mathbf{x}(n+1) = \mathbf{f}(\mathbf{W}\mathbf{x}(n) + \mathbf{W}^{\text{in}}\mathbf{u}(n+1)), \tag{1}$$

where $\mathbf{f}$ are the internal unit's output functions – here we use the identity[1], applied element-wise to the network state vector. The ESN thus becomes a linear network, as far as the reservoir activations are concerned. The output is computed according to EdNote(1)

$$\mathbf{y}(n) = \mathbf{f}^{\text{out}}(\mathbf{W}^{\text{out}}(\mathbf{x}(n), \mathbf{u}(n)), \tag{2}$$

where $\mathbf{f}^{\text{out}}$ is the output activation function – here we use $1/2 + \tanh/2$[2], which is a sigmoid ranging strictly between 0 to 1 with a value of $1/2$ for a zero argument. Introducing a sigmoid-shaped nonlinearity here is crucial for achieving a *stable* reproduction of the motif. EdNote(2)

3 Melody coding

We study "melody" signals, that is, sequences $m(n)$[3], where $n = 1, 2, ...$ and $m(n)$ can take p equidistant values $i/(p-1)$ between 0 and 1 (where $i = 0, 1, ..., p-1$). In order to feed such a signal to the ESN, it is space-coded by mapping $m(n)$ on a p-dimensional[4] binary vector $\mathbf{b}(n)$[5]. Specifically, the space coding transforms $m(n) = i/(p-1)$ to a binary vector $\mathbf{b}(n)$ whose i-th component is one. EdNote(3) EdNote(4) EdNote(5)

Our networks will be trained to generate various time-delayed versions of such space-coding vectors $\mathbf{b}(n)$ as their output signals. Because we use a $(0, 1)$-ranging sigmoid $\mathbf{f}^{\text{out}}$ for the output activation function, this would make it impossible for the network to produce zero values in $\mathbf{b}(n)$ output vectors. To accomodate for this snag, we shift and scale[6] the vectors $\mathbf{b}(n)$ to vectors $\mathbf{u}(n)$ whose components range in $(0.1, 0.9)$: EdNote(6)

$$\mathbf{u}(n)[i] = 0.8\ \mathbf{b}(n)[i] + 0.1. \tag{3}$$

Such code vectors $\mathbf{u}(n)$ have components $\nu = 0.1$ and $\mu = 0.9$ where $\mathbf{b}(n)$ have 0 and 1. They provide the data format which the network finally receives

[1] EdNote: Matlab: esn.reservoirActivationFunction

[2] EdNote: Matlab: esn.dataActivationFunction

[3] EdNote: Matlab: rawSignal

[4] EdNote: Matlab: p

[5] EdNote: Matlab: sampleInput, sampleOutput, the mapping is done with the helper function rawToUnitCoded. Notice that originally the code was written for coding arbitrary analog values into a p-dimensional vector, which is what rawToUnitCoded does. For the purpose of this paper, I however used only binary coding vectors. Leaving the rawToUnitCoded parts in place for later experiments, the analog-equivalent space coding is reduced to binary coding by taking the max of the vectors created by rawToUnitCoded.

[6] EdNote: Matlab: done via esn.inputShift, esn.inputScaling, esn.teacherShift, and esn.teacherScaling

as inputs, and which it should produce as outputs.

4 Training ESNs as delay-line memories

4.1 Data preparation

We train ESNs to work as delay-line memories. The goal is that the trained network, on an input sequence $\mathbf{u}(n)$, outputs d vectors[7] $[\mathbf{u}(n-d)\mathbf{u}(n-d+1)\dots\mathbf{u}(n-1)]$. Technically, we arrange the vectors $\mathbf{u}(n-d), \mathbf{u}(n-d+1), \dots, \mathbf{u}(n-1)$ into a $p \times d$ array $\mathbf{y}(n)$[8]. The training data are thus generated as follows[9]: EdNote(7) EdNote(8) EdNote(9)

1. Produce a random sequence $m(n)$ of length N and transform it into a space-coded, scaled/shifted sequence $\mathbf{u}(n)$.

2. For each $n > d$, assemble $\mathbf{y}(n)$ from the previous d instances of $\mathbf{u}(n)$. For the first $n \leq d$, use dummies (initial transients of the network will be discarded anyway).

3. The training data consist of $n_{\max}$ input – teacher output pairs $(\mathbf{u}(n), \mathbf{y}(n))$.

4. Create a similar pair sequence for testing purposes.

In our study, we use $p = 10$ pitch bins, $d = 10$ delays and $n_{\max} = 1500$ for training data and $n_{\max} = 500$ for testing.

Notice that we train the network on a delay memory task – that is, the network outputs are trained to "retrodict" the input. However, when later used in a task where the objective is to detect and repeat a k-periodic motic, the semantics of the outputs change from retrodiction to prediction. When the input is k-periodic, the *current* output trained on a delay of $k-1$ will *predict* the *next* input; or equivalently, the *previous* output trained on a delay of $k-1$ will predict the *current* input.

4.2 Network setup

We use an ESN with a reservoir of $N = 100$ units[10]. The internal weights $\mathbf{W}$ EdNote(10)

[7] EdNote: Matlab: the number d of delays is the variable esn.d

[8] EdNote: Matlab: the concatenation order is to sort the longest-delay outputs to the left in this array. The teacher signal thus becomes an array of size signal-duration $\times p \times d$, and the network outputs are collected in a $p \times d$ array netOut (used in run_melody_8).

[9] EdNote: Matlab: in demo_8, this is done in the code block 1.2

[10] EdNote: Matlab: the creation of the network is done in generate_esn_8.

are drawn from a uniform distribution over $[-1, 1]$, with approximately 90% of the connection weights becoming nulled, resulting in an average connectivity of 10%. The sparse weight matrix is then rescaled to yield a spectral radius of 0.8.

The input weights $\mathbf{W}^{\text{in}}$ are drawn from a uniform distribution over $[-1, 1]$[11]. EdNote(11)

For each delay i, we assign a separate output weight matrix $\mathbf{W}_i^{\text{in}}$ of size $p \times (K + N)$

4.3 Training and testing

The ESN was trained on the delay-line memory task in the standard fashion described, for instance, in [2]. The result of the training are d output weight matrices $\mathbf{W}_k^{\text{in}}$. In order to stay away from overfitting, a regularizer is implemented in the form of uniform noise[12] from $[-0.0005, 0.0005]$ added to the network states harvested while the ESN is driven by the training input. Figure 1 shows the basic setup used in training. EdNote(12)

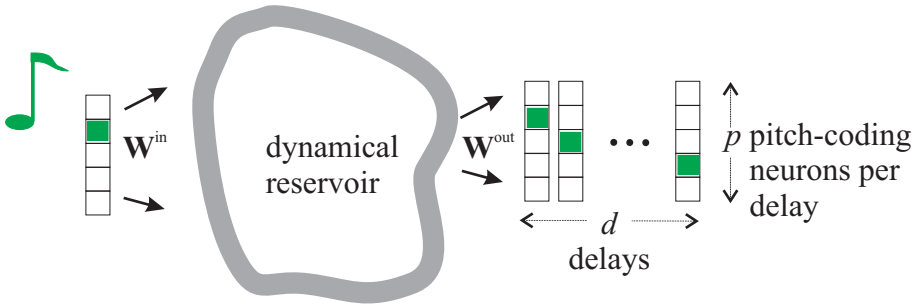

Figure 1: The basic ESN setup during training.

Notice that in the light of the concepts and findings reported in [2], an ESN capable of perfect performance on this memory task would need a memory capacity of $p\,d = 100$, the theoretical maximum for a 100-unit ESN. Because this theoretical maximum will not be reached (some of the network's theoretical memory capacity would be associated with signals not used here), we can expect suboptimal learning performance especially on longer delays. In fact, a plot[13] of the normalized mean root square errors (NRMSE) on training data shows that the NRMSE of recalling earlier inputs is rather poor (see EdNote(13)

[11] EdNote: Matlab: by setting esn.inputWeightScaling to 1.0

[12] EdNote: Matlab: trainNoiseLevel in demo_8

[13] EdNote: Matlab: the diagnostics for the learning/testing performance is displayed in Matlab figure windows 1 – 5

figure 2). The difficulties the ESN has with larger delays are also reflected in the large output weights earned on larger delays (figure 2, right panel). A detrimental consequence of large output weights is high sensitivity to noise. We can expect that if the trained network is used and noise is added to its operation, the produced outputs on the larger-delay channels will strongly deteriorate.

While we could have easily achieved a more accurate and more noise-resistant performance for the larger delays by using a larger reservoir, we did not do so in order to investigate how the picking-up of a motif deteriorates when the length of the motif ranges into the limits of the underlying short-term memory.

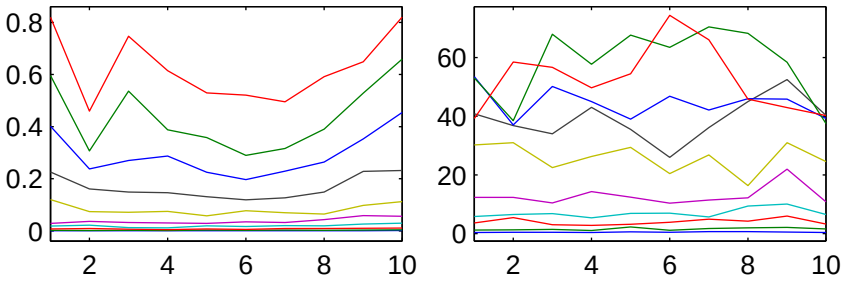

Figure 2: Left: Test error (NRMSE) for the ten pitches (x-axis). Each plot line corresponds to one delay, larger delays give higher test error. Note that a NRMSE of 1 would correspond to completely uncorrelated test/teacher signals; thus the NRMSE of about 0.7 for the largest delay characterizes a quite poor accuracy. Right: average absolute output weights for the various pitches (x-axis) and delays.

5 Picking up a motif and repeating it

5.1 Basic ideas

Once trained, the network is ready to be used in a setting where it receives a cue signal which starts with a random "melody", then presents a "motif" twice, then stops. After the cue signal has terminated, the network should continue to produce output autonomously, such that the output has two properties:

1. The output is a periodic signal consisting of approximate repetitions of the motif which was presented twice at the end of the cue.

2. The output is *stable*, that is, the network has settled into an *attractor* reflecting the motif.

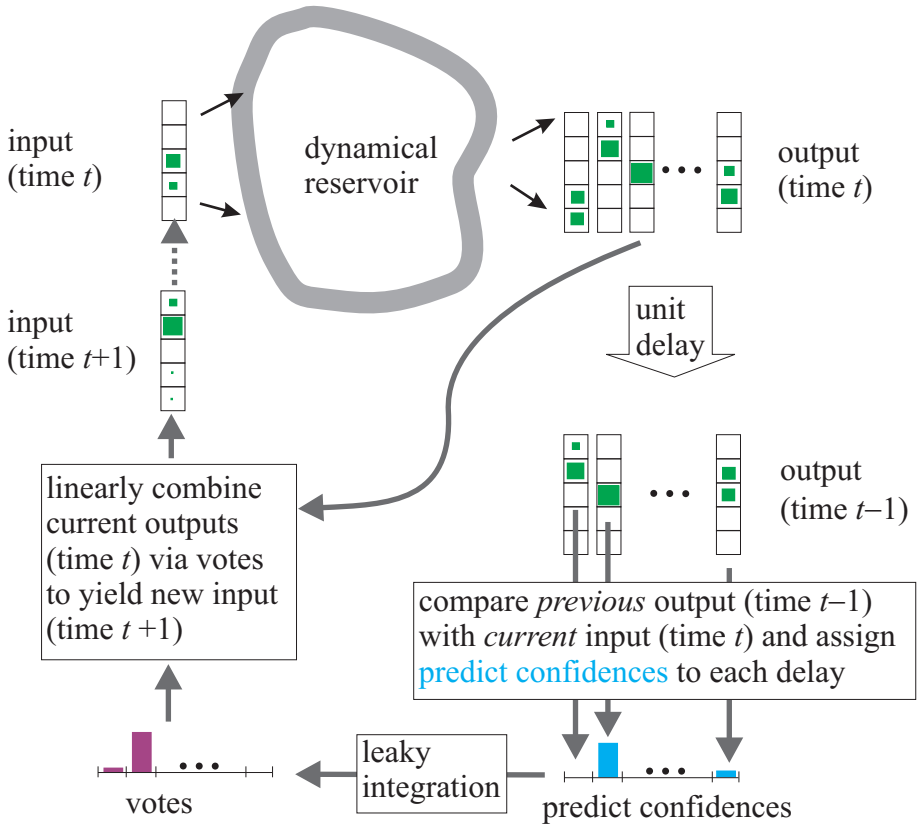

Figure 3: ESN setup during autonomous operation.

The basic idea to realize such an operation is the following (see figure 3).

- The network's output (that is, d delayed versions of the current input) are monitored for how well they predict the current input. In the presence of an input which is periodic with period k[14], the k-delay block of outputs should show consistently good agreement with the input. The prediction agreement across the d output blocks is integrated over time and the "winner" output block is allowed to feed its signal back into the input channel. EdNote(14)

[14] EdNote: Matlab: k is periodN in demo_8

- Technically, this requires (i) measuring and integrating the prediction error for the d output blocks, (ii) using the accumulated error as a basis for a competitive vote among the d outputs to determine one (or several) winners, (iii) feed the winning ouput(s) back into the input.
- The network is operated in two distinct phases. First, in the cueing phase, an external cue input (consisting in a transient random initial melody followed by two repetitions of a motif) is used as input. After that, the external input is replaced by the fed-back output(s) which are linearly combined according to the strength of their respective votes.

5.2 Error integration and voting

The measuring and integration of prediction error is done by a leaky integration scheme[15]. For each delay i ($i = 1, \ldots, d$), in each update cycle n the average $\mathrm{MSE}_i(n)$ across the p components of the ith pitch coding vector is computed[16]: EdNote(15) EdNote(16)

$$\mathrm{MSE}_i(n) = \|\mathbf{y}_i(n-1) - \mathbf{u}(n)\|^2/p, \tag{4}$$

where $\mathbf{y}_i(n-1)$ is the previous network output for delay i (a vector of size p) and $\mathbf{u}(n)$ is the current input to the ESN. These MSE_i are leaky-integrated according to

$$\text{MSE-int}_i(n) = \tanh((1-\gamma_1)\ \text{MSE-int}_i(n-1) + \alpha_1\ \mathrm{MSE}_i(n)), \tag{5}$$

where γ_1 is the forgetting rate and α_1 is an accumulation weight[17]. In the simulations reported below, we used $\gamma_1 = .4, \alpha_1 = 4$. The tanh wrapper makes the integrated prediction error saturate at 1. EdNote(17)

The d integrated error signals thus obtained will evolve towards zero on all delays that are a multiple of the period length k for a k-periodic input. Likewise, for delays which are not multiples of k, the integrated errors will evolve towards 1.

An obvious mechanism to "vote" for outputs which should be fed back into the input channel would be to weigh ouputs by "predict confidences" $C_i(n) = 1 - \text{MSE-int}_i(n)$, average across the weighted outputs, and feed the

[15] EdNote: Matlab: the first part of the "voting and competing" code block in run_melody_8

[16] EdNote: Matlab: in run_melody_8, this is errors_per_delay(d, 1)

[17] EdNote: Matlab: γ_1 is errDecay and α_1 is errGrowthConstant in run_melody_8. The development of the MSE-int$_i$'s is plotted in Matlab output figure 7

resulting mixture back to the input. In this way, only those ouputs whose delays are multiples of k – i.e., replicas of the periodic target – would be chosen.

However, the integrated errors in the delay channels not commensurate with k will not in general evolve towards 1, because within the periodic input motif there may be time points with spurious good matches between the current motif signal and incommensurable delays – this is bound to occur when the motif contains notes with similar pitch. Thus, before becoming useful for guiding the combination of outputs into new fed-back inputs, some further cleaning of the integrated errors is necessary. We do this by two operations. First, the predict confidences[18] $C_i(n) = 1 - \text{MSE-int}_i(n)$ are thresholded at their lower end such that confidences falling below a threshold ε[19] will be zeroed, that is, instead of $C_i(n) = 1 - \text{MSE-int}_i(n)$ we use EdNote(18) EdNote(19)

$$C_i(n) = s(1 - \text{MSE-int}_i(n)), \tag{6}$$

where $s : \mathbb{R} \to [0, 1]$ maps any number smaller than ε to zero, any number greater or equal to 1 to 1, and linearly interpolates in between. In the simulations reported below, we use a threshold of $\varepsilon = 0.3$. Finally, a further leaky integration and subsequent normalization smoothes $C_i(n)$ to obtain the final votes V_i:

$$\begin{aligned} \tilde{V}_i(n) &= (1 - \gamma_2)\, V_i(n-1) + \alpha_2\, C_i(n) \\ V_i(n) &= \tilde{V}_i(n) / \sum_{j=1,\ldots,d} \tilde{V}_j(n) \end{aligned} \tag{7}$$

where again γ_2, α_2 are forgetting / accumulation factors[20]. In the simulations reported below, we used $\gamma_2 = .2, \alpha_2 = 4$. EdNote(20)

The details of how we realized an error-monitoring-and-voting mechanism here are not important. We could have designed a more efficient mechanism, for instance, by setting the maximum vote to 1 and the others to 0 at each time step – this would result in a sharper, faster detection of input periodicity. The reason why we relied on two cascaded leaky integrations to clean our votes is an admittedly feeble attempt to remain biologically plausible.

[18] EdNote: Matlab: currentPredictConfidence

[19] EdNote: Matlab: ε is zeroPredictConfidenceThreshold, used in run_melody_8

[20] EdNote: Matlab: figure 6 displays the development of the clean vote signals $V_i(n)$

5.3 Feeding back the vote-combined output

In the cueing phase, after a "warmup" presentation of a random melody (which serves to wash out the ESN startup transient), the external input consists of two repetitions of a motif of period length k. During the presentation of the second of the motif repeats, the network outputs match with the k-step previous input and the vote $V_{k-1}(n)$ will grow toward 1, while the other votes move toward zero (it is not $V_k(n)$ that will grow toward 1 due to the unit delay in the feedback circle, see figure 3). When the cue period ends after the second motif repetition, the external input is switched off and the network receives instead an input $\mathbf{u}(n)$ which is essentially made from a weighted combination of the network outputs:

$$\tilde{\mathbf{u}}(n) = \sum_{i=1,\ldots,d} V_i(n-1)\ \mathbf{y}_i(n-1). \tag{8}$$

A little refinement is needed here, though. The network outputs $\mathbf{y}_i(n-1)$ contain some inaccuracies and will not perfectly correspond to pitch-coding vectors $\mathbf{p}(n-1)$. Specifically, if the scaling/shifting from equation 3 is undone to bring back $\mathbf{y}_i(n-1)$ into the original pitch coding format,

$$\tilde{\mathbf{y}}_i(n-1) = (\mathbf{y}_i(n-1) - [0.1...0.1]')/0.8 \tag{9}$$

(where ' denotes vector transpose), then the component sum of $\tilde{\mathbf{y}}_i(n-1)$ would in general slightly deviate from 1, the value required for perfect pitch coding vectors. In order to enforce clean pitch coding, we normalize $\tilde{\mathbf{u}}(n)$ by putting[21] EdNote(21)

$$\begin{aligned}
\tilde{\tilde{\mathbf{u}}}(n) &= (\tilde{\mathbf{u}}(n) - [0.1...0.1]')/0.8 \\
\tilde{\tilde{\tilde{\mathbf{u}}}}(n) &= \tilde{\tilde{\mathbf{u}}}(n)/\text{component sum of } \tilde{\tilde{\mathbf{u}}}(n) \\
\mathbf{u}(n) &= 0.8\ \tilde{\tilde{\tilde{\mathbf{u}}}}(n) + 0.1
\end{aligned} \tag{10}$$

Notice that we must carry out the normalization on a version of $\tilde{\mathbf{u}}$ where the scaling/shifting from equation 3 has been undone.

In order to check the stability of the periodic pattern reproduction, we added uniform noise $\nu(n)$ to the (fed-back) input $\mathbf{u}(n)$ in the simulations reported below[22]. EdNote(22)

[21] EdNote: Matlab: code block "restore perfect pitch coding" in run_melody_8

[22] EdNote: Matlab: noiseLevel argument in run_melody_8

6 Performance

We tested our trained ESN with motifs of different length k and under various amounts of noise $\nu(n)$. Depending on k and $\nu(n)$, interestingly different types of performance were observed.

Motifs of length 6 or 7. In this condition, when the noise is not too strong ($|\nu(n)| \leq 0.005$), the voting mechanism focusses on the appropriate value of k before the second cue motif has ended, that is, the vote of for the delay $k-1$ goes to 1 and the others to 0. A periodic pattern similar to the cue motif is stably reproduced. Figure 4 shows a typical run for $k = 7$, without and with noise. It is apparent that moderate noise does not disrupt the periodic motif reproduction. Figure 5 shows the development of the integrated errors MSE-int$_i(n)$ and the resulting votes $V_i(n)$. If the noise level is increased to a range around $|\nu(n)| \approx 0.01$, the system is driven out of its attractors after every few repetitions of the attractor's loop, whereafter it settles into another periodic attractor, usually of the same period as the previous but distinguished from it by settling on different values on some of the period's time points (figure 6); more rarely or when the noise is further increased, it may jump to different period lengthes or become non-periodic altogether.

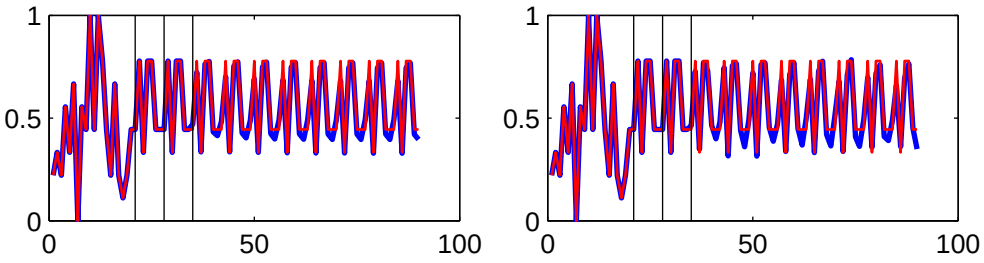

Figure 4: Network output (thick blue line, re-transformed from pitch coding vectors to one-dimensional "melody" signal) vs. network cycles (x-axis). The cue signal is rendered by a thin red line. Vertical black lines mark the two cue motifs. Notice that the ESN does not receive the cue signal after the second presentation of the motif. Left panel shows zero noise condition, right panel with noise $|\nu(n)| \leq 0.005$.

Motifs of length 8 – 10. If the length of the motif grows into the region where the delay line memory performance of the ESN is poor, the inaccuracies in recalling the input from $k-1$ steps earlier accumulate to

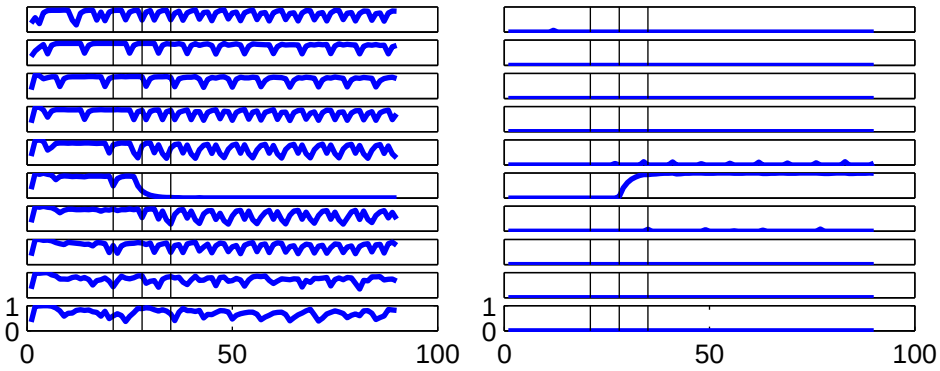

Figure 5: Development of integrated errors (left panel) and votes (right panel) in the run with added noise rendered in the right panel of figure 4. Each plot corresponds to one delay; top plot corresponds to shortest delay $k = 1$.

an extent that the motif is not stably retained over time, even when no noise is inserted into the dynamics. Figure 7 shows a typical development for a $k = 9$ cue. Although the cue motif is initially reproduced a few times, the reproduction accuracy is not good enough to retain the pattern. A complex transient dynamics unfold, which after a much longer time (1000 steps, not shown) would eventually settle in an attractor not related to the cue motif in shape or period.

Motifs of length 4 or 5. Here two mechanisms become superimposed (see figure 8 for an example where $k = 5$, zero noise condition). The voting mechanism correctly determines the period length k by the time the second cue motif ends, and the motif is initially correctly reproduced. However, because a k-periodic signal is also $2k$-periodic, the vote for delay $2k$ also rises soon after the system starts to produce the pattern autonomously, leading to a shared vote between k and $2k$. Because the $2k$-delay output channel has a poor reproduction performance, errors accumulate and the reproduced pattern wanders away from the original. The long-term behaviour is unpredictable; often the systems settles in a k-periodic attractor unrelated in its shape to the cue, or (more rarely) settles into an attractor with a different period. Notice that this behaviour could be remedied by implementing a winner-take-all mechanism between the votes which would prevent the $2k$ vote from rising. We would then obtain a stable reproduction of the cue motif.

Motifs of length 2 or 3. If the motif is very short, the voting mechanism

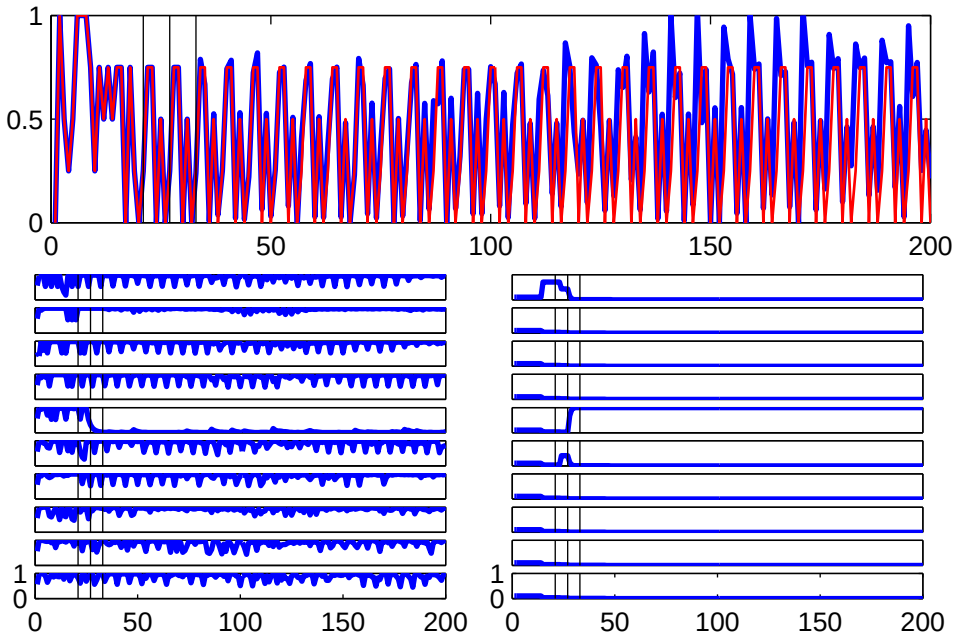

Figure 6: Similar conditions as in figures 4 and 5, but with medium noise (here of size 0.06), leading to "hopping" between attractor variants.

needs more time than is afforded by a double presentation of the cue to rise toward 1 for the correct period length k. In our simulations, three instead of two successive presentations were needed. Similar to the case of motif length 4 or 5, after the reproduction sets in, the votes for multiples of k rise, too. For example, if the motif has period 3 (this case is shown in figure 9), the votes for $2k$ and $3k$ subsequently share their saying with the vote for k. Unlike the case of motif length 4 or 5, however, here we have two voted channels (k and $2k$) with a sufficient accuracy of recall, which outweigh the detrimental influence from the inaccurate channel $3k$. In the end, a stable reproduction of the original motif is ensured even in the presence of noise.

7 Analysis

Our melody-pick-up ESN apparently hosts a huge number of different cyclic attractors, into which it can be individually driven through a presentation of corresponding cues. In this section we analyze this phenomenon mathematically.

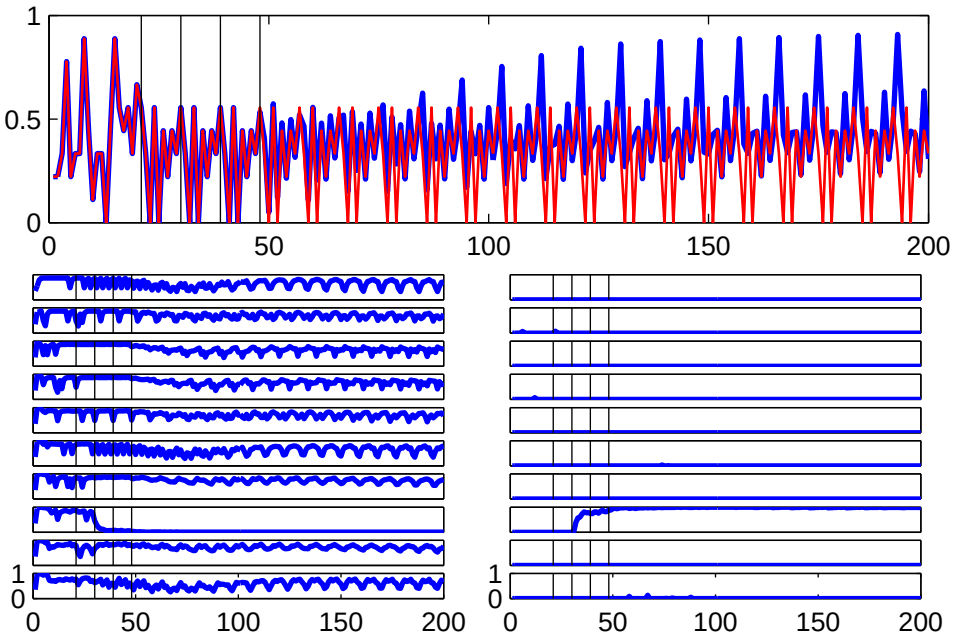

Figure 7: A run on a cue of length 9, under a zero noise condition.

7.1 Problem simplification

We first simplify our architecture, stripping away a number of features, until only the core phenomenon of a dynamical system hosting many cyclic attractors is left.

The first non-essential feature is all the error monitoring and voting mechanisms. If we had an oracle instead that after the presentation of the second cue motif simply sets the vote on the k-delay output to 1 and the others to 0, – or even simpler, if we knew the period length k from the beginning and would use an ESN with only the k-delay outputs installed at all –, then we would still have an architecture hosting a large number of periodic attractors (now all of period k), which requires an explanation.

The second non-essential feature is the ESN reservoir itself. At time n, the un-squashed k-delay output

$$(\mathbf{f}^{\text{out}})^{-1}\mathbf{y}_k(n) = \mathbf{W}_k^{\text{out}}(\mathbf{x}(n), \mathbf{u}(n)) \tag{11}$$

is extracted from the information in $(\mathbf{x}(n), \mathbf{u}(n))$ in a least-mean-square error sense via $\mathbf{W}_k^{\text{out}}$. The relevant component in $(\mathbf{x}(n), \mathbf{u}(n))$ which is read out by $\mathbf{W}_k^{\text{out}}$ is the "echo" in the reservoir state reflecting the input from k steps earlier, namely

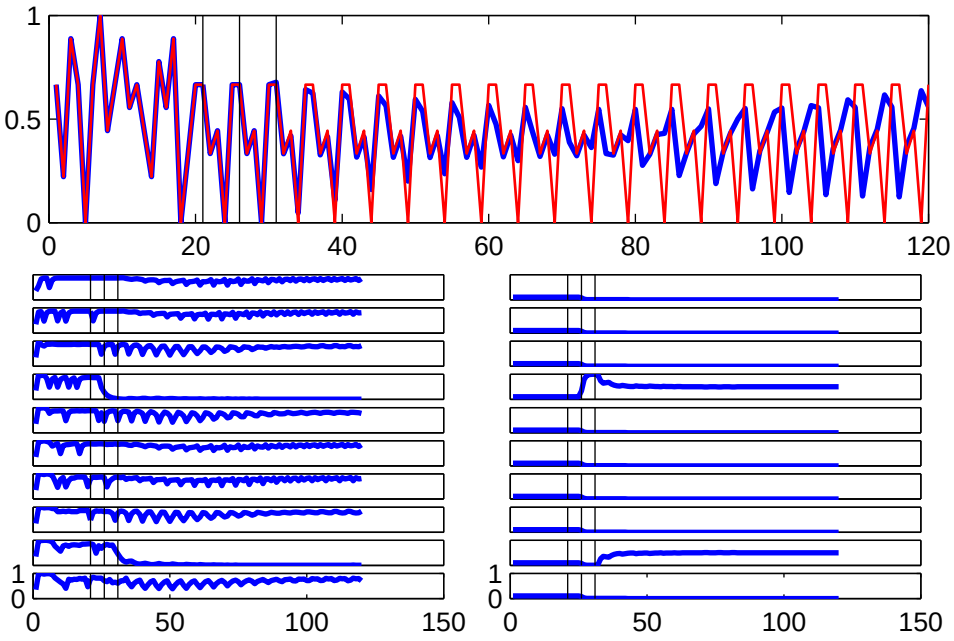

Figure 8: Performance on a cue motif with length $k = 5$, zero noise condition. Panels show network output, integrated error, and votes, respectively.

$$\mathbf{W}^{k-1}\mathbf{W}^{\text{in}}\mathbf{u}(n-k+1) = \mathbf{W}^{k-1}\mathbf{W}^{\text{in}}\mathbf{y}_k(n-k), \tag{12}$$

assuming that the output $\mathbf{y}_k(n-k)$ is fed back to become the input $\mathbf{u}(n-k+1)$. All other signal components in $(\mathbf{x}(n), \mathbf{u}(n))$, which result from linear superpositions of the fading echoes of other inputs, are filtered out by $\mathbf{W}_k^{\text{out}}$ as well as possible. Using a large reservoir is just a way to provide enough degrees of freedom such that a linear separation of all these superimposed echoes becomes possible with a reasonable accuracy. We would only improve the retrodiction performance if we did away with the reservoir and installed a simple delay-line memory, turning equation 11 into

$$(\mathbf{f}^{\text{out}})^{-1}\mathbf{y}_k(n) = \mathbf{W}^{\text{direct}}\mathbf{u}(n-k+1) = \mathbf{W}^{\text{direct}}\mathbf{y}_k(n-k). \tag{13}$$

The final item that we can dispense with is k and the motif's periodicity. Generally, if in a discrete-time dynamical system – that is, an iterated map F – we have a k-periodic attractor that loops through values $\mathbf{z}_1, \ldots, \mathbf{z}_k$, analysing its stability reduces to investigating the stability of the fixed point $\mathbf{z}_1$ under the map F^k. In our case, we must analyse the stability of fixed points of iterated maps of the kind

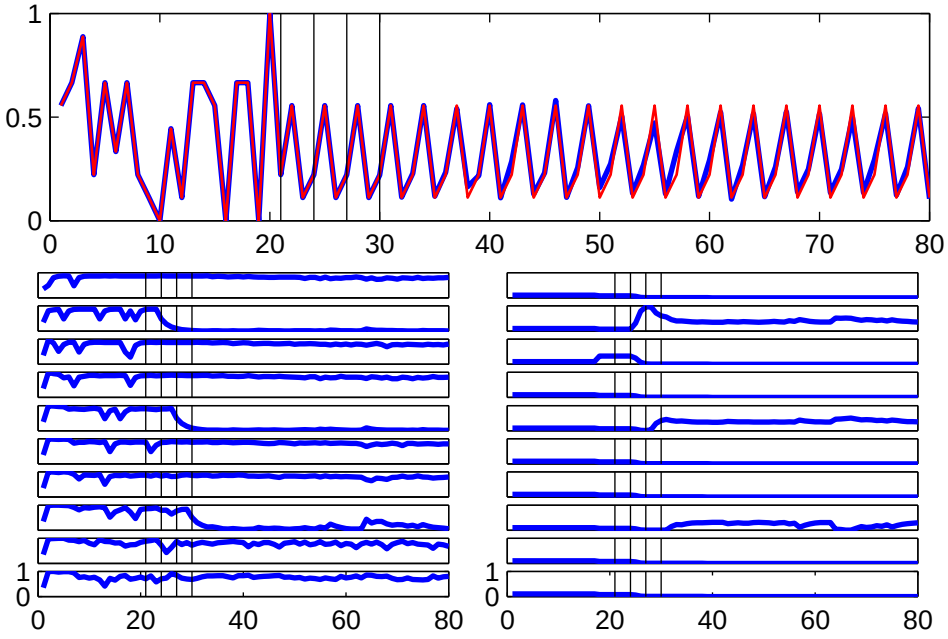

Figure 9: Performance on a cue motif with length $k = 3$. Noise size is $|\nu(n)| \leq 0.005$. Panels show again network output, integrated error, and votes, respectively.

$$\mathbf{y}(n+1) = \mathbf{f}^{\text{out}}\mathbf{W}^{\text{direct}}\mathbf{y}(n). \tag{14}$$

The goal, then, is to demonstrate that maps as in equation 14 have many stable fixed points – namely, as many as we find possible pitches coded per time step in our original setting. Concretely, every vector $\mathbf{y}(n)$ that has a single entry with a value of μ, while the other entries are $\nu = 1 - \mu$, should be stable fixed points. Figure 10 visualizes what we are after.

We will thus investigate the following question: if $\mathbf{W}^{\text{direct}}$ is such that on μ, ν-component vectors $\mathbf{y}$ the MSE $E[\|\mathbf{y} - \mathbf{f}^{\text{out}}\mathbf{W}^{\text{direct}}\mathbf{y}\|^2]$ is minimal, what point attractors does this map have? Specifically, are the p vectors $\mathbf{y}_i$ (where $i = 1, ..., p$ and $\mathbf{y}_i$ is μ in the ith component, elsewhere being ν) stable fixed points?

7.2 Stability of periodic patterns

We start with a simple observation: if $\mathbf{f}^{\text{out}}$ were the identity, that is, if $\mathbf{f}^{\text{out}}\mathbf{W}^{\text{direct}}$ were a linear map, then we could not obtain stable fixed points

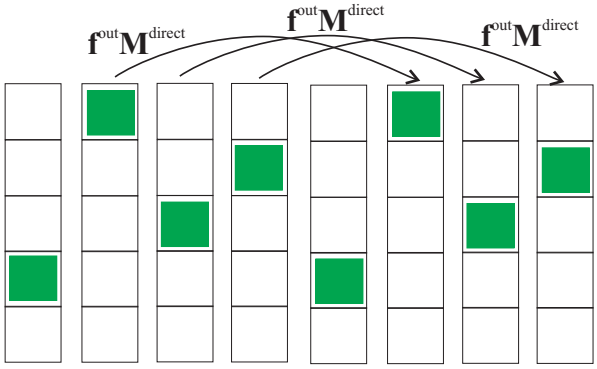

Figure 10: Periodic motif generation seen as a combination of stable fixed points of a map $\mathbf{f}^{\text{out}}\mathbf{W}^{\text{direct}}$. Here the case $k = 4, p = 5$ is shown.

other than zero (a generic property of linear maps). Thus, the sigmoid character of $\mathbf{f}^{\text{out}}$ is decisive.

Another simple observation is that if $\mathbf{W}^{\text{direct}}$ maps each $\mathbf{y}_i$ on $(\mathbf{f}^{\text{out}})^{-1}\mathbf{y}_i$, we obtain a zero MSE $E[\|\mathbf{y} - \mathbf{f}^{\text{out}}\mathbf{W}^{\text{direct}}\mathbf{y}\|^2]$. It is not difficult to see that such a matrix $\mathbf{W}^{\text{direct}}$ is of the form

$$\mathbf{W}^{\text{direct}} = \begin{pmatrix} \alpha & \beta & \cdots & \beta \\ \beta & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \beta \\ \beta & \cdots & \beta & \alpha \end{pmatrix}. \tag{15}$$

Let us consider the fixed point $\mathbf{z} = \mathbf{y}_1 = (\mu\nu \ldots \nu)^{\mathsf{T}}$ (where superscript T denotes transpose) and demonstrate that it is stable. If this is found true, then the other fixed points $\mathbf{y}_i$ are stable too by symmetry.

The fixed point $\mathbf{z}$ is stable if it is stable in its first and second component (the remaining components are identical to the second), that is, if

$$\mathbf{f}^{\text{out}\,\prime}((\mathbf{W}^{\text{direct}}\mathbf{z})[1])\ \|\mathbf{W}^{\text{direct}}\| < 1 \text{ and} \tag{16}$$
$$\mathbf{f}^{\text{out}\,\prime}((\mathbf{W}^{\text{direct}}\mathbf{z})[2])\ \|\mathbf{W}^{\text{direct}}\| < 1, \tag{17}$$

where $\mathbf{f}^{\text{out}\,\prime}$ is the derivative of $\mathbf{f}^{\text{out}}$, $(\mathbf{W}^{\text{direct}}\mathbf{z})[1]$ is the first component of $(\mathbf{W}^{\text{direct}}\mathbf{z})$, and $\|\mathbf{W}^{\text{direct}}\|$ is the norm (largest singular value) of $\mathbf{W}^{\text{direct}}$, which due to the symmetry of this matrix coincides with its spectral radius, i.e. its absolute greatest eigenvalue. We start our verification of equations 16, 17 by computing the spectral radius of $\mathbf{W}^{\text{direct}}$.

7.2.1 The spectral radius of $\mathbf{W}^{\mathbf{direct}}$

It is clear from the structure of $\mathbf{W}^{\text{direct}}$ that $e_0 = (1 \cdots 1)^\mathsf{T}$ is an eigenvector (to an eigenvalue λ_0), and furthermore $e_1 = (1s \cdots s)^\mathsf{T}, e_2 = (s1s \cdots s)^\mathsf{T}, \ldots, e_p = (s \cdots s1)^\mathsf{T}$, where $s \neq 1$, are eigenvectors, these latter all to the same eigenvalue λ_1. $e_1, \ldots, e_p$ span a linear subspace of dimension at least $p-1$ and at most p. It is p iff $\lambda_0 = \lambda_1$. In either case, there cannot be more eigenvalues other than λ_0 or λ_1. Elementary algebra (for details see Appendix **??**) reveals that

$$\begin{aligned} \lambda_0 &= \alpha - \beta + p\,\beta && (18) \\ \lambda_1 &= \alpha - \beta, && (19) \end{aligned}$$

or in terms of μ and ν (see Appendix **??**),

$$\begin{aligned} \lambda_0 &= \frac{(\mathbf{f}^{\text{out}})^{-1}\mu - (\mathbf{f}^{\text{out}})^{-1}\nu}{\mu - \nu} + p\,\frac{\mu\,(\mathbf{f}^{\text{out}})^{-1}\nu - \nu\,(\mathbf{f}^{\text{out}})^{-1}\mu}{((p-1)\nu + \mu)(\mu - \nu)} && (20) \\ \lambda_1 &= \frac{(\mathbf{f}^{\text{out}})^{-1}\mu - (\mathbf{f}^{\text{out}})^{-1}\nu}{\mu - \nu} && (21) \end{aligned}$$

of which the second is positive real and has the greater absolute value of the two (Appendix B) and hence is the spectral radius of $\mathbf{W}^{\text{direct}}$.

7.2.2 Computing $\mathbf{f}^{\mathbf{out}\,\prime}((\mathbf{W}^{\mathbf{direct}}\mathbf{y}_1)[i])$

The other item in equations 16, 17 that we need to determine is the derivative $\mathbf{f}^{\text{out}\,\prime}$ at the first and second component of $\mathbf{W}^{\text{direct}}\mathbf{y}_1$ (the remaining components are equal to the second). We consider the first component first. The derivative of $\mathbf{f}^{\text{out}}$ is

$$\mathbf{f}^{\text{out}\,\prime}(x) = (\frac{1}{2}\,\tanh x + \frac{1}{2})' = \frac{2}{(e^x + e^{-x})^2}. \qquad (22)$$

Using the obvious $(\mathbf{W}^{\text{direct}}\mathbf{y}_1)[1] = (\mathbf{f}^{\text{out}})^{-1}\,\mu$, $(\mathbf{f}^{\text{out}})^{-1}x = \frac{1}{2}log\frac{x}{1-x}$ and equation 22, elementary transformations yield

$$\mathbf{f}^{\text{out}\,\prime}((\mathbf{W}^{\text{direct}}\mathbf{y}_1)[1]) = 2\,(1-\mu)\,\mu. \qquad (23)$$

7.2.3 Concluding the proof of stability

In order to verify equation 16, and using equations 21 and 23, we have to show that for $1/2 < \mu < 1$ it holds that

$$2\,(1-\mu)\,\mu\,\frac{(\mathbf{f}^{\text{out}})^{-1}\mu-(\mathbf{f}^{\text{out}})^{-1}\nu}{\mu-\nu}<1. \tag{24}$$

Utilizing $(\mathbf{f}^{\text{out}})^{-1}x=\frac{1}{2}log\frac{x}{1-x}$ and $\mu=1-\nu$ yields the following equivalent version of equation 24:

$$2\,(1-\mu)\,\mu\,\frac{\log\frac{\mu}{1-\mu}}{2\mu-1}<1. \tag{25}$$

We substitute μ by $1/2+\varrho$, which yields another equivalent version of equation 24:

$$\frac{\log\frac{1/2+\varrho}{1/2-\varrho}\,(1/2-2\,\varrho^2)}{2\,\varrho}<1 \quad \text{for } 0<\varrho<1/2. \tag{26}$$

Figure 11 gives a plot of this function. Notice that this function is not defined in $\varrho=0$; however, it can be smoothly (with respect to the first derivative) closed at this point with a value of 1. Appendix C proves this and also that this function is properly smaller than 1 for $0<|\varrho|<1/2$.

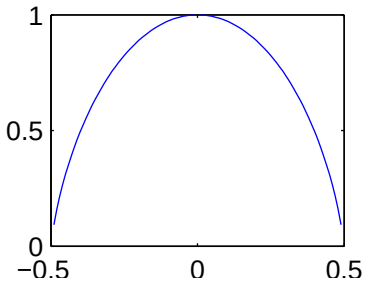

Figure 11: Plot of the function from equation 26 for $-1/2<\varrho<1/2$.

It appears from figure 11 that the closer we move $|\varrho|$ to $1/2$ (that is, μ to 1), the more stable should our attractors become. While this is certainly the case for our simplified model, the situation is not so clear for the original ESN + voting system. The reason is that moving μ closer to 1 interacts with the output weights computed in the training of the ESN as a delay line. The closer to 1 we set μ, the larger the output weights become. This earns us an increase in noise sensitivity which to a certain extent cancels the improvement of stability we would expect from the phenomenon in figure 11. We did not systematically investigate this matter but can only report a spurious finding that if μ is set to 0.975 instead to 0.9, the stability of reproducing $k=7$ patterns appeared to improve slightly.

7.2.4 The exponential number of periodic attractors

Assuming that our simplifications are admissible, we may interpret the workings or our ESN-based dynamical system as follows. For a delay k that is learnt with sufficient accuracy, and assuming that the voting mechanism singles out the appropriate period length k, the system implements k interleaved maps $\mathbf{f}^{\text{out}}\mathbf{W}^{\text{direct}}$, as sketched in figure 10. Given that this map has p point attractors, the entire system hosts p^k periodic attractors of period k. Due to cyclic shift and other symmetries, the actual number of "really" different attractors is less than this – and not easy to calculate precisely – but is certainly $O(p^k)$. If we would restrict ourselves to delays $k = 1, ..., d_{\max}$ that are learnt with sufficient accuracy (in our concrete simulations: $d_{\max} = 7$), this means that our network architecture hosts $O(p^{d_{\max}})$ periodic attractors.

Considering some fixed p, how does $d_{\max}$ scale with network size, that is, the number N of reservoir units plus the number of input and output units? Here we call upon the main finding in [2], which in a condensed version states that the short-term memory capacity of a linear-reservoir ESN is N. Roughly speaking, this means that in order to augment an ESN that is capable of recalling the dth previous input p-vector with a certain accuracy, to a network that also recalls the $d + 1$th previous input with a similar accuracy, we have to grant another p reservoir neurons. Together with the extra p-output-vector needed, we have to add another $2p$ neurons if we wish to enhance our motif-pickupper from some $d_{\max}$ to $d'_{\max} = d_{\max} + 1$. But this is only a linear increase of network size in $d_{\max}$. Thus, pending a more detailed analysis, it is fair to say that the number of periodic motifs that can be picked up and reproduced via our architecture grows exponentially with the size of the network.

8 Discussion

We have provided an ESN-based architecture that is capable of homing in on any from an exponential number of periodic attractors if cued by a similar motif presentend in two (or three, for short motifs) consecutive instances in the input to the network. The ESN was trained as a delay-line memory, and later works in conjunction with an external error integration and voting mechanism.

The essential ingredients for this to function in a stable fashion turned out to be (i) the output nonlinearity of the ESN, and (ii) the spatial pitch coding. The error integration and voting mechanism that we used is, by contrast, accidental; mechanisms of this sort other than the concrete instantiation used

here would work as well (or better) provided they are capable of detecting the correct period length sufficiently fast.

Simulation experiments reveal that a certain accuracy of the ESN's short-term memory recall is needed to ensure a stable reproduction of the cue motif. We arranged things in a way such the network ran into its accuracy limits for long delays, resulting in a breakdown of the performance for long motifs. On motifs of intermediate length the architecture worked as desired. With short motifs we encountered difficulties stemming from the fact that a k-periodic signal is also nk-periodic for $m = 2, 3, \ldots$, which activated feedback channels corresponding to these multiple delays. If such extra feedback channels with a poor recall accuracy become involved, the performance may be impaired. This could be prevented, however, by adding a competitive mechanism (winner-take-all) between the vote signals.

This report is a first investigation into ESNs picking up periodic motifs. Future research will elaborate the connections of this model with empirical findings from music understanding and reproduction by humans, and explore the question of whether similar architectures can be designed for continuous-time and/or continuous-valued periodic patterns.

Acknowledgement. The investigations reported here were inspired and triggered by lively conversations with Douglas Eck, who introduced me to this beautiful modelling challenge. The paper owes much to careful proofreading by Mathias Bode.

A The eigenvalues of $\mathbf{W}^{\mathbf{direct}}$

We consider the eigenvector $e_1 = (1 s \cdots s)^\mathsf{T}$ in the eigenvalue equation $\mathbf{W}^{\mathrm{direct}} e_1 = \lambda e_1$. Writing this equation for the first two vector components gives us

$$\begin{aligned} \lambda &= \alpha + (p-1)\,\beta\, s && (27)\\ \lambda\, s &= \alpha\, s + (p-2)\,\beta\, s + \beta && (28) \end{aligned}$$

Solving these equations for λ by eliminating s leads to a quadratic equation in λ whose roots are $\alpha - \beta + p\,\beta$ and $\alpha - \beta$. Since we know that $\mathbf{W}^{\mathrm{direct}}$ has exactly two eigenvalues, these must be the two eigenvalues. Checking these roots in the equations $\mathbf{W}^{\mathrm{direct}} e_0 = \lambda\, e_0$ and $\mathbf{W}^{\mathrm{direct}} e_1 = \lambda\, e_1$ yields that

$$\begin{aligned} \lambda_0 &= \alpha - \beta + p\,\beta && (29)\\ \lambda_1 &= \alpha - \beta && (30) \end{aligned}$$

In order to obtain eigenvalue expressions in terms of μ and ν, we exploit that $(\mu\nu\cdots\nu)^{\mathsf{T}} = \mathbf{f}^{\text{out}}\mathbf{W}^{\text{direct}}(\mu\nu\cdots\nu)^{\mathsf{T}}$, which gives

$$\begin{aligned} \mu &= \mathbf{f}^{\text{out}}(\alpha\,\mu + (p-1)\,\beta\,\nu) && (31)\\ \nu &= \mathbf{f}^{\text{out}}(\alpha\,\nu + (p-2)\,\beta\,\nu + \beta\,\mu), && (32) \end{aligned}$$

which in turn solve to

$$\begin{aligned} \alpha &= \frac{((p-2)\nu+\mu)\,(\mathbf{f}^{\text{out}})^{-1}\mu - (p-1)\,\nu\,(\mathbf{f}^{\text{out}})^{-1}\nu}{((p-1)\nu+\mu)(\mu-\nu)} && (33)\\ \beta &= \frac{\mu\,(\mathbf{f}^{\text{out}})^{-1}\nu - \nu\,(\mathbf{f}^{\text{out}})^{-1}\mu}{((p-1)\nu+\mu)(\mu-\nu)}. && (34) \end{aligned}$$

Inserting these into eqns. 29 and 30 yields equations 20 and 21.

B The spectral radius of $\mathbf{W}^{\text{direct}}$

We have to decide which of the two eigenvalues λ_0, λ_1 has the greater absolute value. We will show that λ_1 has the greater absolute value.

Because we may assume that $\mu > \nu$, an inspection of equation 21 reveals that $\lambda_1 = \frac{(\mathbf{f}^{\text{out}})^{-1}\mu - (\mathbf{f}^{\text{out}})^{-1}\nu}{\mu-\nu} =: t_1 > 0$. The term $\mu\,(\mathbf{f}^{\text{out}})^{-1}\nu - \nu\,(\mathbf{f}^{\text{out}})^{-1}\mu$ in equation 20 is negative, as can be verified easily when we exploit that $(\mathbf{f}^{\text{out}})^{-1}x = \frac{1}{2}log\frac{x}{1-x}$ and that $\nu = 1-\mu$. This renders the term $p\beta = p\,\frac{\mu\,(\mathbf{f}^{\text{out}})^{-1}\nu - \nu\,(\mathbf{f}^{\text{out}})^{-1}\mu}{((p-1)\nu+\mu)(\mu-\nu)} =: t_0$ in equation 20 negative. Thus in order to demonstrate that $|\lambda_1| > |\lambda_0|$, we must show that $0 < 2\,t_1 + t_0$. Writing out $2\,t_1 + t_0$ yields, after some elementary transformations

$$2\,t_1 + t_0 = \frac{(\mathbf{f}^{\text{out}})^{-1}\mu(2(p-1)\nu + 2\mu - p\nu) - (\mathbf{f}^{\text{out}})^{-1}\nu(2(p-1)\nu + 2\mu - p\nu)}{((p-1)\nu+\mu)(\mu-\nu)}. \quad (35)$$

Since the denominator is clearly positive, in order to show that $2\,t_1 + t_0 > 0$ we have to verify that the term $2(p-1)\nu + 2\mu - p\nu$ is positive, which is easily ascertained under the assumption that $0 < \nu < 1/2$. Thus, the spectral radius of $\mathbf{W}^{\text{direct}}$ is $\lambda_1 = \alpha - \beta = \frac{(\mathbf{f}^{\text{out}})^{-1}\mu - (\mathbf{f}^{\text{out}})^{-1}\nu}{\mu-\nu}$.

C Properties of the function from figure 11

We first show that

$$\lim_{\varrho\to 0} F(\varrho) = \lim_{\varrho\to 0} \frac{\log \frac{1/2+\varrho}{1/2-\varrho}\,(1/2-2\,\varrho^2)}{2\,\varrho} = \lim_{\varrho\to 0} \frac{\log \frac{1/2+\varrho}{1/2-\varrho}}{4\,\varrho} = 1. \tag{36}$$

This amounts to showing that

$$\left(\log \frac{1/2+\varrho}{1/2-\varrho}\right)'(0) = 4, \tag{37}$$

which can be mechanically verified. Similarly, it amounts to a mechanical verification that

$$\lim_{\varrho\to 0} \left(\frac{\log \frac{1/2+\varrho}{1/2-\varrho}\,(1/2-2\,\varrho^2)}{2\,\varrho}\right)' = \lim_{\varrho\to 0} \left(\frac{\log \frac{1/2+\varrho}{1/2-\varrho}}{4\,\varrho}\right)' = 0, \tag{38}$$

thereby ascertaining that F can be smoothly closed (with respect to the first derivative) at $\varrho = 0$ by putting $F(0) = 1$. It remains to show that

$$\frac{\log \frac{1/2+\varrho}{1/2-\varrho}\,(1/2-2\,\varrho^2)}{2\,\varrho} =< 1 \quad \text{for } 0 < |\varrho| < 1/2. \tag{39}$$

This is equivalent to

$$\log \frac{1/2+\varrho}{1/2-\varrho} < \frac{\varrho}{(1/2+\varrho)\,(1/2-\varrho)} \quad \text{for } 0 < |\varrho| < 1/2. \tag{40}$$

Knowing already that $F(0) = 1$ and $F'(0) = 0$, and taking into account that F is clearly symmetric, equation 40 is proven once we have shown that for the derivatives of the two sides of equation 40,

$$A(\varrho) = \left(\log \frac{1/2+\varrho}{1/2-\varrho}\right)' = \frac{1}{(1/2+\varrho)\,(1/2-\varrho)}, \tag{41}$$

$$B(\varrho) = \left(\frac{\varrho}{(1/2+\varrho)\,(1/2-\varrho)}\right)' = \frac{2\varrho^2 + (1/2+\varrho)\,(1/2-\varrho)}{(1/2+\varrho)^2\,(1/2-\varrho)^2}, \tag{42}$$

it holds that

$$1 > A(\varrho)/B(\varrho) = \frac{1/4-\varrho^2}{2\varrho^2 + (1/4-\varrho^2)}, \tag{43}$$

which clearly is the case for $0 < |\varrho| < 1/2$.

References

[1] H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks. GMD Report 148, GMD - German National Research Institute for Computer Science, 2001. http://www.faculty.iu-bremen.de/hjaeger/pubs/EchoStatesTechRep.pdf.

[2] H. Jaeger. Short term memory in echo state networks. GMD-Report 152, GMD - German National Research Institute for Computer Science, 2002. http://www.faculty.iu-bremen.de/hjaeger/pubs/STMEchoStatesTechRep.pdf.

[3] H. Jaeger. Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the echo state network approach. GMD Report 159, Fraunhofer Institute AIS, http://www.faculty.iu-bremen.de/hjaeger/pubs/ESNTutorial.pdf, 2002.

[4] H. Jaeger and H. Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304:78–80, 2004.

[5] F. Pasemann. Characterization of periodic attractors in neural ring networks. *Neural Networks*, 8(3):421–429, 1995.