Can't Get You Out of My Head: A Connectionist Model of Cyclic Rehearsal

Herbert Jaeger[1] and Douglas Eck[2]

[1] Jacobs University Bremen
h.jaeger@jacobs-university.de,
http://www.faculty.jacobs-university.de/hjaeger/
[2] University of Montreal, Department of Computer Science, Canada
douglas.eck@umontreal.ca,
http://www.iro.umontreal.ca/~eckdoug

Abstract. Humans are able to perform a large variety of periodic activities in different modes, for instance cyclic rehearsal of phone numbers, humming a melody sniplet over and over again. These performances are, to a certain degree, robust against perturbations, and it often suffices to present a new pattern a few times only until it can be "picked up". From an abstract mathematical perspective, this implies that the brain, as a dynamical system, (1) hosts a very large number of cyclic attractors, such that (2) if the system is driven by external input with a cyclic motif, it can entrain to a closely corresponding attractor in a very short time. This chapter proposes a simple recurrent neural network architecture which displays these dynamical phenomena. The model builds on echo state networks (ESNs), which have recently become popular in machine learning and computational neuroscience.

Keywords: neural dynamics, periodic attractors, music processing, Echo State Networks.

1 Introduction

One scientific metaphor for the brain is to see it as a nonlinear dynamical system. This view has become popular in cognitive (neuro)science since about a decade [34] [33] and is particularly inviting when it comes to rhythmic or periodic phenomena. However, one should be aware that dynamical systems modeling still has narrow limitations. The mathematical tools of nonlinear dynamcal systems have been developed – mainly by mathematicians and theoretical physicists – for systems which are low-dimensional or structurally or dynamically homogeneous. In contrast, the (human) brain is high-dimensional, and its neural state variables are coupled in complex network structures with few symmetries, and are governed by local dynamical laws which differ greatly between locations and neuron types. Mathematical tools to capture the behaviour of high-dimensional, heterogeneous dynamical systems are in their infancy. The contribution of this chapter is to expand the power of the nonlinear dynamcial systems metaphor a small but definite step further toward structural and dynamical complexity.

I. Wachsmuth and G. Knoblich (Eds.): Modeling Communication, LNAI 4930, pp. 310–335, 2008.

To set the stage, we remark that many everyday human behaviours are periodic, and to a certain degree stable against perturbation. Familiar examples include

- repeating a phone number in one's mind;
- picking up a beat or ostinato motif from a piece of music and keep on tapping or humming it even when the instruments take a pause or after the piece has finished;
- in some children's games, the players are challenged to repeat arbitrary gestures.

For the amusement of the reader, and as a further example, we have put a few audiofiles online[1], which start with some random tone sequence, then repeat a short motif two to four times, then stop. The reader will find it more or less easy, on first hearing, to identify the periodic motif and continue reproducing it (covertly or overtly, to varying degrees of muscial precision). Figure 1 gives a graphical impression.

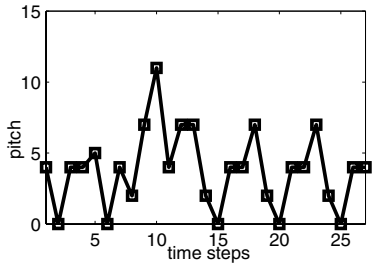

Fig. 1. A periodic pattern that sets in after a non-periodic preceding context can easily be recognized visually and acoustically (if one is instructed to watch out for repetitions). The figure depicts a little melody with discrete time steps and 12 half-tone pitches; its notes (marked by squares) are connected by lines.

Intriguingly, these phenomena combine dynamical and discrete-combinatorial aspects. From the dynamics angle, one will see periodic attractors at work, and would naturally investigate issues like stability, period length, and entrainment. From the combinatorial angle, one will remark that at least some of these examples can be seen as periodic symbol sequences (this is natural e.g. for the phone number rehearsal or muscial motifs), with an essentially arbitrary chaining of symbols from a finite "alphabet". In this perspective, one might find the descriptive tools from computer science and discrete mathematics appropriate, and might wish to investigate how many periodic sequences are combinatorially possible, what is the information of observing or generating one of them, etc.

[1] A zip file with ten demo files (.wav format) can be fetched from, `http://www.faculty.jacobs-university.de/hjaeger/pubs/melodyPickDemos.zip`

Here we present an entirely "dynamical" model, fleshed out as a recurrent neural network (RNN), which can simultaneously account for the dynamical sides of stable periodic pattern generation, as well as for the combinatorial side. Specifically, we describe an RNN architecture with the following properties:

1. the system can generate many periodic patterns which can be seen as symbol sequences;
2. the generation of such a pattern is robust against perturbations, i.e. it can be understood as a periodic attractor;
3. the number of different periodic attractors that the RNN can generate is exponential in the size of the RNN – this is the "combinatorial" aspect;
4. the RNN can be locked into each of its periodic attractors by driving it with a few (two to three) repeated external presentations of a corresponding cue pattern, from a random previous context (as in the demo examples illustrated in Fig. 1).

To preclude a likely misunderstanding, we emphasize that our system does not include an aspect of long-term storage, learning or recall. The periodic attractors hosted by the system have not been learnt, and they are not "stored". Our system realizes a purely dynamical, transient phenomenon, where a motif is picked up and kept "alive" in what one might call a dynamic short-term memory for a while – and that is all.

In order to facilitate the following discussion, we want to give a name to systems with the properties listed above, and call any such system a "cue-addressable periodic attractor system" (CAPAS).

CAPAS' have been variously considered in the cognitive and neurosciences. A well-known case is the cyclic rehearsal subsystem in Baddeley's influential model of acoustic/linguistic STM [1], where in the *phonological loop* cyclic rehearsal prevents memory traces from decaying. Baddeley's model of short-term memory includes a number of further modules and can explain intricate effects observed in human auditory short-term recall, which add important additional structure and functionality beyond go beyond the CAPAS phenomenon.

Instances of short-term imitation of a periodic pattern have been addressed in robotics (e.g., [38] [29]). As a basis for technical realization, coupled oscillator systems are typcially invoked, which become entrained to the external periodic cue signal (but compare [6] for an approach which rests on a chaotic neural network). The phenomena modeled in this kind of research are, on the one hand, simpler than CAPAS' insofar as either the external driving signal persists, reducing the phenomenon to pure entrainment (without the need for autonomous continuation); on the other hand they are more complex, because the imitation/entrainment often includes a mode or coordinate transformation of the driving signal, for instance from external visual 2-D input to motor control signals.

CAPAS are also related to some basic aspects of music cognition. Specifically, they relate to empirical and theoretical research on a listener's entrainment to the rhythmical patterns in a piece of music (overview in [11]). A particular challenge for modelling lies in the fact that the rhythmical patterns in real-life music

are complicated, non-stationary, and replete with exceptions. A seemingly easy recognition task such as detecting the dominant beat can become arbitrarily challenging for cognitive modeling. Apart from symbolic/rule-based approaches, which we will ignore here, there are two major types of "dynamical" approaches. The first builds on neural oscillators which become entrained by the music signal (e.g. [22] [7]). The second type of approach calls methods and mechanisms from linear signal processing (such as delay line memory, coincidence detectors, correlation detectors) which are used to build up a representation of the rhythmic patterns in terms of autocorrelation measures as the stimulus music evolves (e.g. [4] [8]). A general problem for oscillator-based explanations is that the time it takes for the oscillators to synchronize with the driving signal is longer than what is observed in humans (but see [29] where in a motion control domain oscillators become entrained very quickly by adjusting their time constants). A possible problem for autocorrelation-based models is that these do not lend themselves directly (as do oscillator-based models) to be run in a generative mode; however, this is typically outside the scope of investigation.

Periodic (and non-periodic) motor behaviour, linguistic STM and music understanding and generation are phenomenologically and neurally connected in many ways. They are subserved by numerous coupled subsystems in the human cerebrum (see e.g. [12] for fMRI studies of multi-modal brain responses or [31] for therapeutic exploits of these interactions). In ecologically plausible settings, different sensory modes are active simultaneously and interact, as do perception vs. production modes.

Furthermore, we want to point out that much previous work on periodic attractors in recurrent neural networks exists, but it is of a different nature than CAPAS. Previous work almost invariably 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. Another important venue of research 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, [28]. 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 which 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 network is not previously trained to any of these attractors in particular, but acts as a kind of "periodic attractor reservoir".

The concrete model which we are going to develop here is based on Echo State Networks (ESNs, [14] [18]). The reason for choosing ESNs as a core component is that we need a delay line memory with certain additional stability properties; these are offered by ESNs, and the delay line / dynamical STM properties of ESNs are rather well understood [15].

The chapter is organized as follows. First we provide a more detailed motivation for exploring periodic motif reproducing systems, by discussing related phenoma that occur in music processing (Section 2). We proceed through the technical part of our contribution, by explicating the naive CAPAS intuition in the terms of nonlinear dynamical systems (Section 3), giving a short introduction to ESNs (Section 4), and explaining how we encode temporal data (Section 5). After describing our connectionist architecture (Section 6), we report the findings from two simulation studies, which highlight two complementary aspects of the architecture: its behaviour under limitations of resources (the biologically and cognitively relevant situation), and its behaviour when virtually unlimited resources are available (interesting from a theoretical nonlinear dynamics perspective).

All computations were done with Matlab on a notebook PC. The basic code is available online at `http://www.faculty.iu-bremen.de/hjaeger/pubs/MelodyMatlab.zip`. This chapter is an adaptation of a technical report [17], where the theoretically inclined reader can find the mathematical analyses behind this chapter.

2 Repetition and Content-Addressability in Melody Generation

We have noted that our model is capable of hosting a large number of periodic attractors, which can be activated by driving the system with the desired periodic trajectory for a few repetitions. In this sense, the host of attractors can be called *content addressable*. Note that this behavior is very different from that of a traditional recurrent neural network, which is capable of learning to repeat only patterns similar to those on which it has been trained. This property of content-addressable repetition turns out to be crucial for modeling melody production and perception. The vast majority of Western music is metrical, that is, built on a temporal framework of hierarchically-nested periodicities [5]. For example, part of what makes a waltz stylistically identifiable is its metrical structure of three events per measure ("3/4"). Meter lends stability to music in the sense that it provides a temporal framework around which a piece of music is organized. The impact of meter is seen, for example, in how musical repetition is carried out in Western music: when long segments of music are repeated in a piece of music, the repeated segments almost always fit neatly at some level of the metrical hierarchy. For example, in the nursery rhyme "Mary Had a Little Lamb", the melody repeats after exactly four measures (16 quarter notes).

The framework of meter gives rise to the perhaps unexpected side effect of content-addressability in music. Once a temporal framework has been established via meter, *almost any* sequence of notes can fit the style provide that sequence (a) obeys local stylistic constraints, e.g. what key and what scale is used and (b) repeats at the appropriate times. Musically-experienced readers will realize that we are simplifying things greatly. Our goal here is to highlight the fact that almost any sequence of notes (even a random one!) becomes melodic as

soon as it is repeated at appropriate intervals. This suggests that a model able to discern melodies from non-melodies must at least be able to recognize that such repetition of arbitrary patterns is taking place. This amounts to recognizing that sequence $S = s_0, s_1, ..., s_k$ has been repeated, regardless of the actual combination of symbols s_k (here representing musical notes). We invite the listener to compare randomly-generated music to random music repeated at metrically-aligned intervals. To our ears, the random music sounds drifting and meaningless while the repeated music is strongly melodic[2].

Given that music has this content-addressable quality it is perhaps not surprising that previous attempts to learn melodies using dynamical systems (in all cases mentioned here, recurrent neural networks) have not worked very well. Mozer [26] introduced a model called "CONCERT" which used a recurrent neural network trained with Back-Propagation Through Time (BPTT) [37]) to learn a musical style by listening to examples (a task very similar to PPMM). Mozer trained the model on melodic sequences and then sampled from the trained model to see whether model-generated melodies were stylistically similar. Though CONCERT regularly outperformed (in terms of likelihood) third-order transition table approaches, it failed in all cases to find global musical structure and was unable to generate new waltzes when trained on examples from Strauss[3]. Similar results were had in other attempts early attempts to use neural networks to learn melody [32,30]. In short: local interactions between events are easy to learn; global structure is hard.

One reason that dynamical neural networks have difficulty with musical melody is that long-range dependencies are difficult to learn using gradient descent. Specifically, for a wide class of dynamical systems, error gradients flowing "backwards in time" either decay exponentially or explode, yielding the so-called "vanishing gradient problem"[2]. In previous work by the second author [10] this problem was addressed (to some extent at least) using the Long Short-Term Memory (LSTM) hybrid recurrent network [13]. Because LSTM can learn to bridge appropriate long-timescale lags it was able to learn to improvise blues music. However it was *not* able to respond effectively or generate music that was much different from what it had heard. That is, LSTM was able to bridge long time-lags and thus learn the global regularities that make up a musical style, but it was unable to deal with content addressability, having learned a set of recurrent weights specific to the examples on which it was trained.

Thus we argue that two qualities are valuable for addressing CAPAS with a dynamical system. First the system must discover repeated sequences (content-addressability; periodicity). Second the system must have a short-term memory capable of bridging long timespans. In following sections we will show that these qualities are met by an ESN-based CAPAS architecture.

[2] For the review versions of this manuscript, reviewers may find these audio files at `http://www.iro.umontreal.ca/~eckdoug/repetition/`

[3] In his interesting and well-argued paper, Mozer cannot be faulted for making inflated claims. He described the output of CONCERT as "music only its mother could love".

3 CAPAS Spelled Out as Dynamical Systems

In this section we give a more formal account of the particular version of CAPAS which will subsequently be realized as an RNN, and introduce some basic notation.

The CAPAS's that we consider operate in discrete time and generate (and are driven by) discrete-valued signals which assume only a finite (and small) number of different values. To aid intuition, we employ a music-inspired terminology and refer to the possible different values as "pitches". We assume that there are p different pitches which are coded as equidistant values $i/(p-1)$ between 0 and 1 (where $i = 0, 1, ..., p-1$).

We furthermore assume that a CAPAS is initially driven by an external *cue* signal $m(n)$ (think of it as a "melody"), which itself has two phases. The first phase establishes a "distractor" context through a random sequence of length M_0 of pitch values, which is followed by r repetitions of a periodic motif of length k. All in all, the cue signals will have length $M_0 + rk$.

Our CAPAS's are driven by the cue $m(n)$ for times $n = 1, \ldots, N_0 + rk$; after which point the cue signal stops. The CAPAS's generate an open-ended output $y(n)$ for times $n = 1, 2, \ldots$.

Working in discrete time, we will thus be designing a model of the general type $\mathbf{x}(\mathbf{n}+\mathbf{1}) = f(\mathbf{x}(\mathbf{n}), m(n))$, $y(n+1) = h(\mathbf{x}(\mathbf{n}+\mathbf{1}))$, where $\mathbf{x}(\mathbf{n}+\mathbf{1})$ is the system state, f, h are some nonlinear functions. This scheme only captures the cueing phase. When it ends, the system gets the input signal $m(n)$ no longer. It is then replaced by the output signal $y(n)$ to yield a system of type $\mathbf{x}(\mathbf{n}+\mathbf{1}) = f(\mathbf{x}(\mathbf{n}), y(n))$, $y(n+1) = h(\mathbf{x}(\mathbf{n}+\mathbf{1}))$. In short, the driving input to the system can either be $m(n))$ or the fed-back own productions $y(n)$:

$$\mathbf{x}(\mathbf{n}+\mathbf{1}) = f(\mathbf{x}(\mathbf{n}), \{m(n)|y(n)\}) \tag{1}$$

$$y(n+1) = h(\mathbf{x}(\mathbf{n}+\mathbf{1})) \tag{2}$$

Having decided on the basic mathematical format, we can now express the basic dynamical properties desired from our system:

1. After the system has been driven with a two-phase cue signal, it should continue to run autonomously after time $M_0 + rk$ (with its own output fed back to its input channels). The output generated should be periodic of period length k, and this periodic pattern should be stable against perturbations. In other words, after the cueing the system should be locked into a periodic attractor. The output signal of this attractor should be close to the periodic motif of the cue.
2. The system (1) must be able to lock into a combinatorially large number of different attractors. By "combinatorially large" we mean that each attractor (modulo cyclic shift) corresponds to a sequence of length k, where each element may be one out of p possible elements. There exist in the order of p^k such sequences (due to shift symmetries the exact number is smaller and hard to calculate, but still at least exponential in a factor of k).

3. The system (1), when driven by an eventually periodic cue $m(n)$, should become entrained to this driving signal quickly. Concretely, we desire that the locking occurs within 2 to 4 repetitions.

We are not aware of a previous connectionist model or abstract nonlinear dynamical system which could satisfy these requirements. Specifically, the combination of noise robustness with fast entrainment appears hard to realize. As pointed out in Section 1, existing models for related periodic entrainment phenomena typically build on coupled oscillators – which do not synchronize with the driving cue quickly enough – or on linear operators (synchrony detection, delay line memory) which lack the desired stability in the free production phase.

4 Basic ESN Concepts and Notation

One component of the model proposed in this chapter is a delay-line memory, that is a subsystem whose input signal $u(n)$ is available on the output side in delayed versions $u(n-1), u(n-2), ..., u(n-k)$. Such delay lines are elementary filters in linear signal processing, but a neurally plausible realization is not to be immediately found. In the context of precise short-term timing tasks, a number of short-term neural timing mechanisms have been proposed [25]. One of these models [3] explains cerebellar timing by assuming that the input signal elicits transient nonlinear responses in an essentially randomly connected neural subsystem (the granule cells); from this complex network state the timing outputs are obtained by adaptive readout-neurons (Purkinje cells). The basis of temporal processing, in the view of this model, is a combination of the temporal evolution of a high-dimensional nonlinear input response in a large random network, with task-specific readout neurons that compute the desired output from the information implicit in this apparently random state.

This idea is also constitutive for the twin approaches of *Echo State Networks (ESNs)* [14] [16] [18] and *Liquid State Machines (LSMs)* [24] [19] [23]. Developed simultaneously and independently, these approaches devise a generic neuro-computational architecture which is based on two components. First, a *dynamical reservoir* (or *liquid*). This is a large, randomly connected, recurrent neural network which is dynamically excited by input signals entering the reservoir through, again, essentially random input connections. Second, adaptive *readout* mechanisms compute the desired overall system response signals from the dynamical reservoir states. This happens typically by a linear combination which can be learned by solving a linear regression problem (for which the Widrow-Hoff learning rule [36] provides a biologically realizable implementation). While this basic idea is shared between ESNs and LSMs, they differ in their background motivation and the type of neurons that are typically used. ESNs originated from an engineering perspective of nonlinear signal processing and use simple sigmoid or leaky integrator neurons for computational efficiency and precision; LSMs arose in computational neuroscience, aim at the biological modeling of cortical microcircuits and typically use spiking neuron models.

Here we use ESNs because we are not claiming neurophysiological adequacy, and because ESNs are easier to analyze. Specifically, we can benefit from the analysis of their dynamic short-term memory properties given in [15].

For a comprehensive introduction to Echo State Networks the reader is referred to the cited literature. The remainder of this section only serves to fix the notation. We consider discrete-time ESN networks with K input units, N internal network units and L output units. The activations of input units at time step n are $\mathbf{u}(n) = (u_1(n), \ldots, u_K(n))^\mathsf{T}$, of internal units are $\mathbf{x}(n) = (x_1(n), \ldots, x_N(n))^\mathsf{T}$, and of output units $\mathbf{y}(n) = (y_1(n), \ldots, y_L(n))^\mathsf{T}$, where $\cdot^\mathsf{T}$ denotes transpose. 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

$$\mathbf{x}(n+1) = \mathbf{f}(\mathbf{W}\mathbf{x}(n) + \mathbf{W}^{\text{in}}\mathbf{u}(n+1)), \tag{3}$$

where $\mathbf{f}$ are the internal unit's output functions – here we use the identity, 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

$$\mathbf{y}(n) = \mathbf{f}^{\text{out}}(\mathbf{W}^{\text{out}}(\mathbf{x}(n); \mathbf{u}(n)), \tag{4}$$

where $(\mathbf{x}(n); \mathbf{u}(n))$ is the concatenation of vectors $\mathbf{x}(n)$ and $\mathbf{u}(n)$, and $\mathbf{f}^{\text{out}}$ is the output activation function – here we use $1/2+\tanh/2$, 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.

5 Signal Coding

To recall, the input signals which we will use are sequences $m(n)$, 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 binary vector $\mathbf{b}(n)$. Specifically, the space coding transforms $m(n) = i/(p-1)$ to a binary vector $\mathbf{b}(n)$ whose i-th component is one.

The ESN will be trained to function as a delay line memory, generating various time-delayed versions of such space-coding vectors $\mathbf{b}(n)$ as its 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 or unit values in

$\mathbf{b}(n)$ output vectors. To accommodate for this circumstance, we shift and scale the vectors $\mathbf{b}(n)$ to vectors $\mathbf{u}(n)$ whose components range in $(\nu, \mu) = (0.1, 0.9)$:

$$\mathbf{u}(n)[i] = (\mu - \nu)\ \mathbf{b}(n)[i] + \nu = 0.8\ \mathbf{b}(n)[i] + 0.1. \tag{5}$$

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 as inputs, and which it should produce as outputs.

6 Architecture

The core of the CAPAS proposed here is a delay line memory. We realized this as a nonlinear recurrent neural network of the "Echo State Network" type. A number of reasons invited us to use such a network to implement the delay line, and not a linear tapped delay line made from concatenated unit delay filters, which would appear as the most standard implementation:

- We wish to create *stable* periodic dynamics, i.e., periodic *attractors*. Linear systems cannot generate stable perdiodic patterns. Nonlinear dynamics are needed, and ESNs have proven in the past to sustain periodic attractors [15].
- Theoretical results on the delay-line memory capacity of ESNs are available [15] [35].
- Ultimately we aim at connectionist models of cognitive and neural phenomena. The central mechanism of ESNs, an excitable random nonlinear dynamical medium (the "reservoir"), has been suggested as a neurobiological explanation for cerebellar timing [25], which adds further to our motivation to use these models.

The ESN delay line memory is the only trainable part of our CAPAS architecture. We first describe how this subsystem is set up and trained, and then proceed to describe the surrounding control architecture, likewise implemented in a connectionist fashion, which provides the CAPAS functionality.

The delay line subsystem is a standard ESN, as specified in eqns. (3), (4), with p input units, a linear reservoir of size N, and $p\,d$ sigmoid output units which for clarity are arranged in a $p \times d$ grid (see figure 2). For each delay j, we assign a separate output weight matrix $\mathbf{W}^{\text{out}}_j$ of size $p \times (K + N)$, whose elements are determined by the training procedure.

The delay line memory task solved by this network is to replicate, in the jth column of output units, the p-dimensional input signal $\mathbf{u}(n)$ with a delay of j, that is, the signal vector $\mathbf{u}(n - j)$. The ESN is trained on the delay-line memory task in the standard fashion described, for instance, in [15]. Details are reported in the appendices.

This (trained and operative) delay line memory is embedded in a larger feedback cycle which is outlined in figure 3. The working principle can be outlined, as follows.

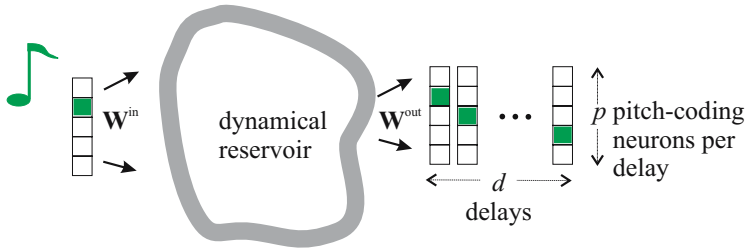

Fig. 2. The core ESN delay line memory setup

- 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, the k-delay block of outputs will be in consistent good agreement with the input. The prediction agreement for each of the d delayed output vectors is integrated over time and the "winner" output block is allowed to feed its signal back into the input channel.
- 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 output(s) back into the input.
- In our simulations we operate the systems 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.

The details of our implementation are documented in Appendix A.

7 Simulations Studies

We ran two suites of numerical experiments to demonstrate two complementary aspects of our CAPAS architecture.

The first series of simulations used a small ESN of $N = 100$ throughout and explored a number of dynamical phenomena, of potential relevance for cognitive modelling, that arise from resource limitations.

The second suite served to demonstrate that the proposed architecture is fit to host a number of periodic attractors which is exponential in network size. To this end, a series of CAPAS architectures of increasing size was shown to host periodic attractors whose number grew exponentially with the size.

7.1 Study 1: CAPAS with Limited Resources

All details of this study can be found in Appendix B. Here we present an intuitive overview of the setup and the findings.

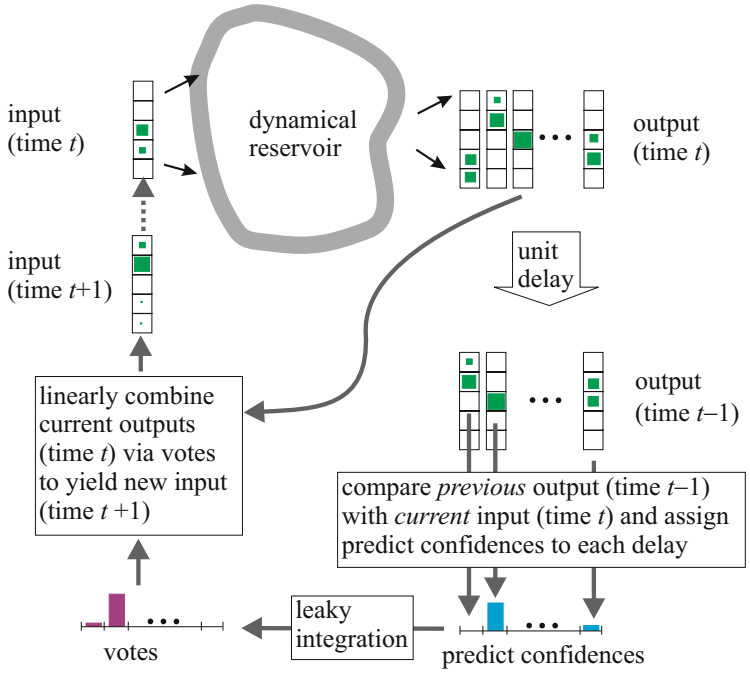

Fig. 3. The overall CAPAS architecture

An relatively small ($N = 100$) ESN was trained as a delay line memory in a setup with $p = 10$ input units and $d = 10$ delays. In the light of findings reported in [15], an ESN capable of perfect performance on this delay line 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, we can expect suboptimal learning performance especially on longer delays. In fact, a plot of the normalized mean root square errors (NRMSE) on training data shows that the NRMSE of recalling inputs with a longer delay is rather poor (see figure 4). The difficulties the ESN has with larger delays are also reflected in the large output weights earned on larger delays (figure 4, 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 (as we did in the second suite of experiments), we did not do so in order to investigate how the CAPAS performance deteriorates when the length of the motif ranges into the limits of the underlying short-term memory.

The trained ESN delay line memory was then employed as a module within the complete CAPAS architecture shown in figure 3, which was submitted to various tests involving motifs of different length k and amounts of noise added

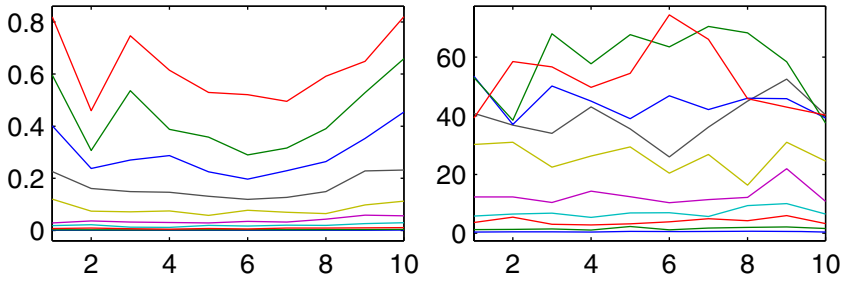

Fig. 4. Left: Delay learning test error (NRMSE) for the ten pitches (x-axis). Each plot line corresponds to one delay, larger delays give higher test error. Note that a NRMSE of 1 would correspond to completely uncorrelated test/teacher signals. Right: average absolute output weights for the various pitches (x-axis) and delays. Larger delays give larger weights.

to the input. In all conditions except for very short motifs, the motif was cued with $r = 2$ repetitions. Depending on k and the amount of noise, interestingly different types of performance were observed.

Motifs of length 6 or 7. In this condition, when the noise is not too strong (amplitude ≤ 0.005), the voting mechanism locks on the appropriate value of k before the second cue motif has ended, that is, the vote 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 5 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 6 shows the development of the leaky-integrated errors and the resulting votes. If the noise amplitude is increased to 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 7); more rarely or when the noise is further increased, it may jump to different period lengths or become non-periodic altogether.

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 an extent that the motif is not stably retained over time, even when no noise is inserted into the dynamics. Figure 8 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 unfolds, which after a much longer time (1000 steps, not shown) would eventually settle in a shorter attractor not related to the cue motif in shape or period.

Motifs of length 4 or 5. Here two mechanisms become superimposed (see figure 9 for an example where $k = 5$, zero noise condition). The voting

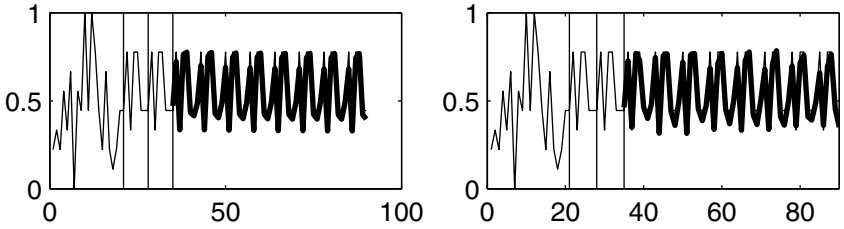

Fig. 5. Cue signal (thin line) and autonomously generated signal (thick line) vs. network cycles (x-axis). Vertical black lines mark the two cue motifs. The correct periodic cue continuation is indicated by the continued thin line (unknown to the system); system output is plotted as a bold line. The unit-coded system output was retransformed to the scalar signal $m(n)$ for plotting. Left panel shows zero noise condition, right panel with a noise amplitude of 0.005.

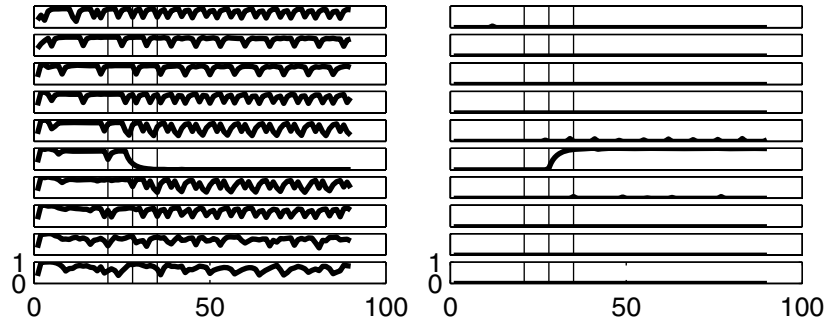

Fig. 6. Development of integrated errors (left panel) and votes (right panel) in the run with added noise rendered in the right panel of figure 5. Each plot corresponds to one delay; top plot corresponds to shortest delay $k = 1$.

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 needs more time than is afforded by a double presentation of the cue to

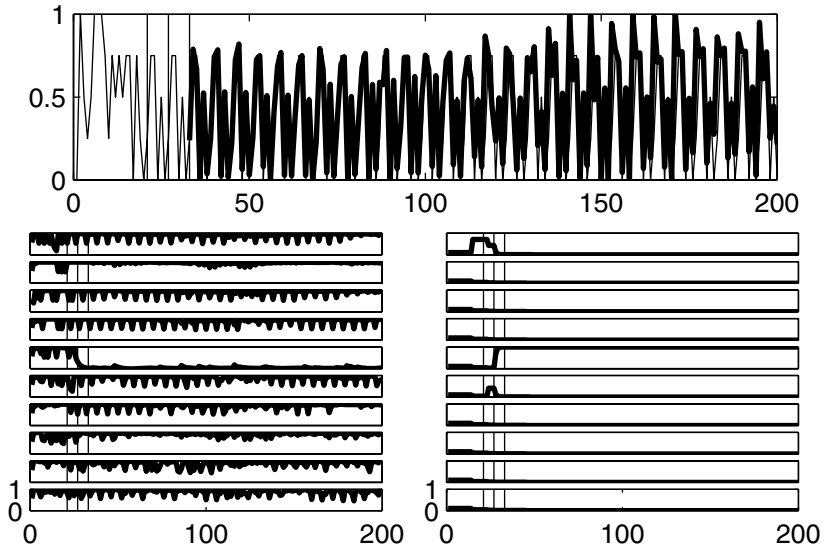

Fig. 7. Similar conditions as in figures 5 and 6, but with medium noise (here of size 0.06), leading to "hopping" between attractor variants.

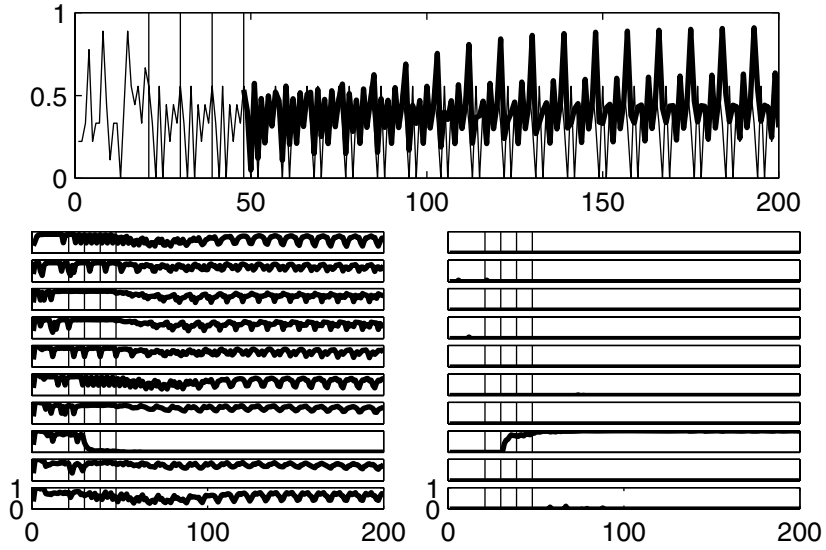

Fig. 8. A run on a cue of length 9, under a zero noise condition

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

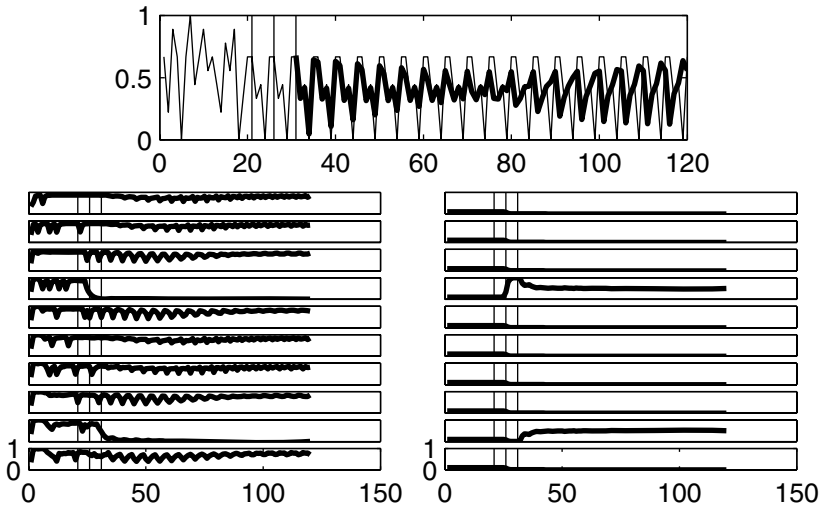

Fig. 9. Performance on a cue motif with length $k = 5$, zero noise condition. Panels show network output, integrated error, and votes, respectively.

k rise, too. For example, if the motif has period 3 (this case is shown in figure 10), 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.

7.2 Study 2: Hosting Exponentially Many Periodic Attractors

Again, all details of this study can be found in Appendix C. Here we present an intuitive overview of the setup and the findings.

We created CAPAS architectures using ESN reservoirs whose size was increased from $N = 800$ in increments of 800 to $N = 4000$ (which is about the largest size that could be accomodated by the available 2 GB main memory computer that we used). Each of these was first trained as a delay line memory in a setup with $p = 5$ input units and $d = 0.6N$ delays, that is, with delays ranging from 30 in increments of 30 to 150. For each size N, ten ESNs were randomly created and trained. Figure 11 shows the delay line recall accuracies achieved.

After this training step, each ESN delay line module was planted into an architecture like in figure 3 and submitted to the CAPAS task. Specifically, a network of size N would be tested for periodic motifs of length $k = N/40$, that is, $k = 20, 40, \ldots, 100$ for the models of increasing size. The test setup was similar to the first suite of experiments. The cues consisted of an initial random "melody" sequence of length $M_0 = 20+2k$, followed by three repetitions of a random motif

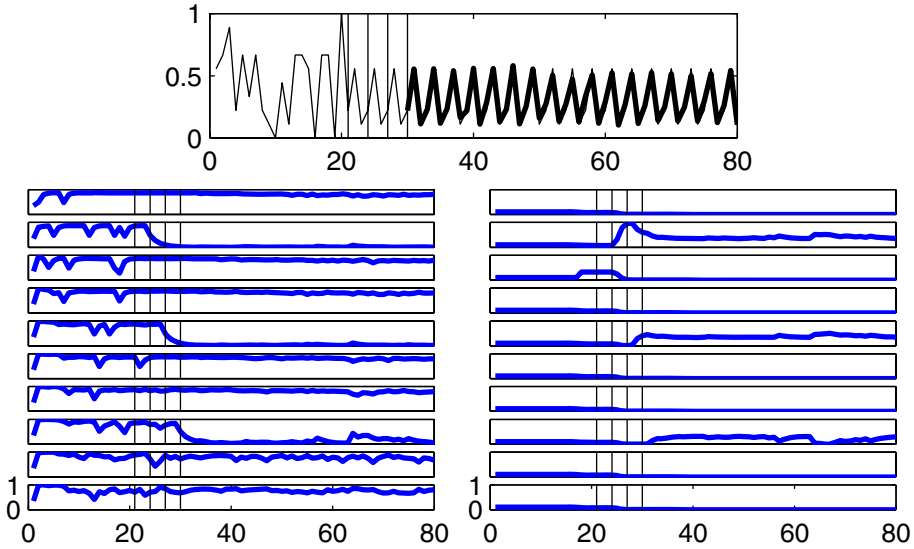

Fig. 10. Performance on a cue motif with length $k = 3$. Noise size is 0.005. Panels show again network output, integrated error, and votes, respectively.

of length k. In order to check whether the system stably locked into the required periodic pattern, it was ran after the end of the cue for 25 repetitions of the period, while noise was added to the fed-back input to challenge stability. After the 25th repetition, the noise was turned off and the system run for another 5 period cycles, of which the last was used for error measurements. The feedback noise was uniform of amplitude $0.01 \cdot 2^{-k/10}$, that is, the noise was scaled inverse exponentially with the the period length (which means inversely proportional to the number of attractors). The control parameters for the leaky integration were optimized by hand for the objective to yield a stable motif detection and reproduction (see Appendix C). Each trained system was tested on ten randomly generated periodic motifs, yielding 100 tests altogether for each network size. Figure 12 summarizes the findings.

The gist of these experiments is that

- the proposed architecture with the hand-tuned control parameters indeed was fit to produce systems hosting in the order of p^k periodic attractors, where $p = 5$ and $k = 20, \ldots, 100$;
- however, the stability and the accuracy of the attractors degraded with period length k, and the attractor length 100 marks about the longest that appeared feasible with networks sized $40\,k$ and the chosen noise levels.

The source of the decline in stability and accuracy of the periodic attractors is the decrease in accuracy of the underlying delay line memory (figure 11, left) and the simultaneous reduction of noise resistance in the delay line memory (see figure 11 right; larger output weights imply increased susceptibility to noise). The

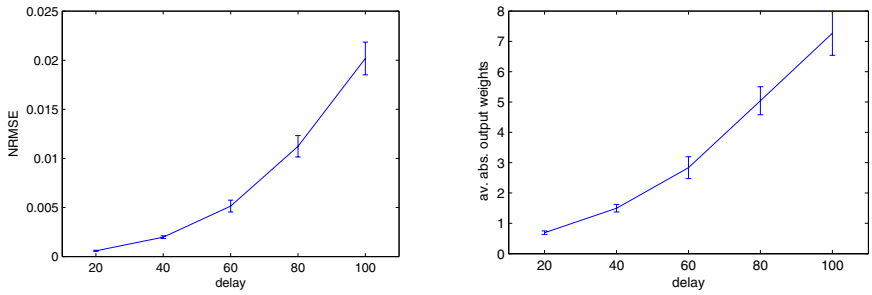

Fig. 11. Left: test NRMSEs of ESNs sized $N = 800, 1200, \ldots, 4000$ in the delay line memory of $p = 5$-dimensional data for delays $d = 20, 40, \ldots, 100$. Each plot point gives an average over ten trained networks. Right: corresponding average absolute output weights.

question of whether it could be possible, by an improved design, to host periodic attractors in numbers exponential in network size, *for unlimited network sizes*, amounts to the question of finding a connectionist model of a delay line memory whose memory span grows linearly with the network size, without bounds. This question has a rather mathematical-theoretical flavour; given the finiteness of biological neural systems it is in our opinion not crucial for purposes of cognitive or neural modelling.

We would like to conclude by pointing out that interesting types of breakdowns of performance can be induced when the global control parameters of this CAPAS architecture are changed. Seen reversely, such findings indicate that certain control parameters should be actively adapted if these breakdowns are to be prevented. One example concerns the control of "garden path" periods. When we did the 20-to-100 period survey experiment with faster leaky integration settings (higher leaking rates implying less temporal smoothing; details in Appendix C), we found that about ten percent of the test patterns were not correctly locked into. A closer inspection revealed that these patterns featured repetitions of subpatterns at intevals shorter than the total period length, which raised votes for delays corresponding to the distance of these "garden path" subpattern repetitions. Figure 13 shows one such example.

8 Discussion

We have presented a connectionist architecture for a cue-addressable periodic attractor system (CAPAS). It is capable of hosting periodic attractors ("motifs") which are made from arbitrary discrete "pitch" sequences. Within the experimental range accessible by the available computing resources, we demonstrated that 5^k attractors can be embedded in the dynamics of a system whose core is a recurrent neural network of $40\,k$ units (tested for $k \leq 100$; beyond this size, there are indications of accumulating inaccuracies and instabilities). The

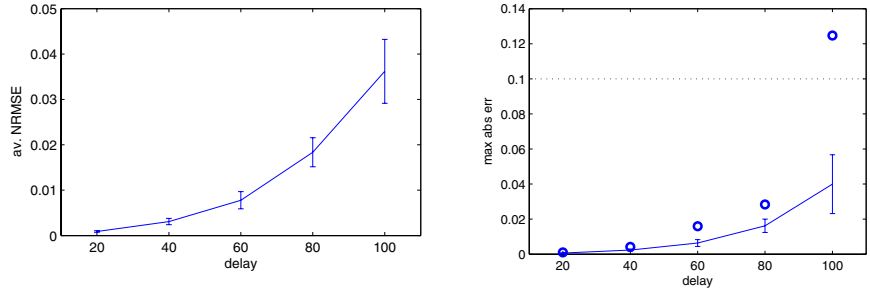

Fig. 12. Left: average test NRMSEs of ESNs sized $N = 800, 1200, \ldots, 4000$ of 100 trials per size N (10 networks tested with 10 random motifs each). The NRMSEs were computed from the last of 30 autonomous repetitions of the motif. Right: maximum absolute difference between CAPAS-produced 30th repetition and the correct target motif. For each trial, the maximum over the k points of the period was taken; these values were then averaged across the 100 trials per network size (solid line). The circles show the maximal maxima. One trial out of 100 in the longest period condition ($k = 100$) brought a maximal deviation greater than 0.1 – note that with $p = 5$ pitches and a pitch range from 0 to 1, an accuracy of better than 0.1 is the critical size needed to uniquely identify a pattern from a degraded replica.

attractors are addressable by cue presentations consisting of two or three repetitions of the periodic target pattern. To our knowledge, this system is the first connectionist architecture capable of being able to lock into any of a combinatorially large number of periodic cues quickly and stably.

We do not claim a close match between our model and biological neural circuits. There are two reasons why such a match cannot be expected. On the one hand, the error-monitoring and voting mechanisms in our model are clearly ad hoc. On the other hand, biological brains need the functionality to pick up periodic motifs in various contexts and modes, and it should be expected that differently structured neural circuitry is used in different instantiations.

The potential interest that our system might have for congnitive and neural modelling resides on a more basic and abstract level than our concrete design. A

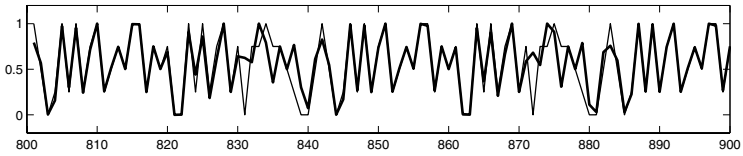

Fig. 13. A section from the last few test cycles of a period-40 test pattern with a "garden path" period of length. Thin line: target pattern, bold line: system output. The target pattern has a length-7 subpattern which repeats at a distance of 18 steps. The CAPAS system response at this time (30th repetition after end of cue) has not fully stabilized; one could say that it still "tries" to find a compromise between the conflicting periods of 40 and 18.

mathematical stability analysis (carried out in [17]) revealed that three aspects of our model are crucial for its capabilities:

1. a delay line memory with a nonlinearity (here: the output unit's sigmoid squashing function) for dynamical stability,
2. the space coding of discrete sequence values by neurons, for yielding a combinatorial mechanism to host very large numbers of attractors, and
3. an error integration and voting mechanism.

This mixture of ingredients mediates between previous models which, when based on attractor entrainment, were too slow to entrain to periodic cues within only very few presentations, or when based on (linear) delay lines, were not dynamically stable.

We conclude by emphasizing that, at least at higher cognitive levels of processing such as music understanding, the dynamics of human periodic pattern processing is certainly more intricate and richer than our simple model. Taking again music processing as an example, human-level processing has at least the following two characteristics which the presented model does not capture:

1. Real-life temporal patterns are often structured in time (multiple timescales, rhythmic structures), and they are modulated in more than just one dimension (of pitch) - for instance, music input would be modulated in timbre, loudness, etc. In sum, real-life dynamical patterns can be much more complex than symbol sequences.
2. Human processing of temporal information certainly often involves accessing long-term memory. In music processing, for instance, it is certainly easier to pick up a periodic pattern that was known beforehand, than to lock into a novel pattern.

We conclude with remarks on music modelling. The current model does not take advantage of the hierarchical nature of musical temporal structure. Music is periodic and repetitive, two qualities accounted for by the current model. But music, at least most Wester music, is also governed by the hierarchical structure seen in meter. Furthermore this structure is generally quite simple, usually a tree having mostly binary divisions with a single beat-level division of some integer ratio (e.g. 3 for a waltz). This hierarchy has a profound impact on how music is performed [27] and perceived [21]. One future direction is to design an explicitly hierarchical ESN-based model which is able to take advantage of such metrical structure. In fact we have taken a first step toward such a model; see [9].

Finally, the way in which the model represents pitch could be expanded. By using simple spatial pitch coding, the model treats all pitches as having equal similarity. In reality (e.g.) a C is more similar to a G than it is to an F# in the key of C. That is, in almost all cases it would sound better to substitute a C with a G than it would to substitute a C with an F#. Having such low-level similarity encoded in the representation is both more human realistic and also more efficient. Given that these effects are relatively well understood [20],

it should be relatively straight-forward to represent pitch for the ESN such that these similarities are easy for the network to discover.

Acknowledgement. The authors would like to thank two anonymous reviewers whose acute observations helped to reshape and improve this contribution quite substantially.

References

1. Baddeley, A.: Working memory: looking back and looking forward. Nature Reviews: Neuroscience 4(10), 829–839 (2003)
2. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks 5(2), 157–166 (1994)
3. Buonomano, D.V.: A learning rule for the emergence of stable dynamics and timing in recurrent networks. Journal of Neurophysiology 94, 2275–2283 (2005), http://www.neurobio.ucla.edu/~dbuono/BuonoJNphy05.pdf
4. Cariani, P.: Temporal codes, timing nets, and music perception. Journal of New Music Research 30(2), 107–135 (2001)
5. Cooper, G., Meyer, L.B.: The Rhythmic Structure of Music. The Univ. of Chicago Press, Chicago (1960)
6. Daucé, E., Quoy, M., Doyon, B.: Resonant spatiotemporal learning in large random recurrent networks. Bological Cybernetics 87, 185–198 (2002)
7. Eck, D.: Finding downbeats with a relaxation oscillator. Psychological Research 66(1), 18–25 (2002), http://www.iro.umontreal.ca/~eckdoug/papers/2002_psyres.pdf
8. Eck, D.: Finding long-timescale musical structure with an autocorrelation phase matrix. Music Perception 24(2), 167–176 (2006)
9. Eck, D.: Generating music sequences with an echo state network. In: NIPS 2006 Workshop on Echo State Networks and Liquid State Machines, Whistler, British Columbia (2006)
10. Eck, D., Schmidhuber, J.: Finding temporal structure in music: Blues improvisation with LSTM recurrent networks. In: Bourlard, H. (ed.) Neural Networks for Signal Processing XII, Proceedings of the 2002 IEEE Workshop, pp. 747–756. IEEE, New York (2002)
11. Gouyon, F.: A computational approach to rhythm detection. Phd thesis, Dpt. of Technology of the University Pompeu Fabra, Barcelona (2005), http://www.iua.upf.edu/mtg/publications/9d0455-PhD-Gouyon.pdf
12. Hickok, G., Buchsbaum, B., Humphries, C., Muftuler, T.: Auditory-motor interaction revealed by fMRI: Speech, music, and working memory in area spt. Journal of Cognitive Neuroscience 15(5), 673–682 (2003)
13. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. Neural Computation 9(8), 1735–1780 (1997)
14. Jaeger, H.: 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.jacobs-university.de/hjaeger/pubs/EchoStatesTechRep.pdf

15. Jaeger, H.: Short term memory in echo state networks. GMD-Report 152, GMD - German National Research Institute for Computer Science (2002), http://www.faculty.jacobs-university.de/hjaeger/pubs/STMEchoStatesTechRep.pdf
16. Jaeger, H.: Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the echo state network approach. GMD Report 159, Fraunhofer Institute AIS (2002), http://www.faculty.jacobs-university.de/hjaeger/pubs/ESNTutorial.pdf
17. Jaeger, H.: Generating exponentially many periodic attractors with linearly growing echo state networks. IUB Technical Report 3, International University Bremen (2006), http://www.faculty.jacobs-university.de/hjaeger/pubs/techrep3.pdf
18. Jaeger, H., Haas, H.: Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. Science 304, 78–80 (2004), http://www.faculty.jacobs-university.de/hjaeger/pubs/ESNScience04.pdf
19. Kaske, A., Maass, W.: A model for the interaction of oscillations and pattern generation with real-time computing in generic microcircuit models. Neural Networks 19(5), 600–609 (2006)
20. Krumhansl, C.: Cognitive Foundations of Muiscal Pitch. Oxford University Press, Oxford (1990)
21. Large, E., Palmer, C.: Perceiving temporal regularity in music. Cognitive Science 26, 1–37 (2002)
22. Large, E., Kolen, J.: Resonance and the perception of musical meter. Connection Science 6(2/3), 177–208 (1994)
23. Maass, W., Joshi, P., Sontag, E.: Computational aspects of feedback in neural circuits. PLOS Computational Biology 3(1), 1–20 (2007)
24. Maass, W., Natschläger, T., Markram, H.: Real-time computing without stable states: A new framework for neural computation based on perturbations. Neural Computation 14(11), 2531–2560 (2002), http://www.lsm.tugraz.at/papers/lsm-nc-130.pdf
25. Mauk, M., Buonomano, D.: The neural basis of temporal processing. Annu. Rev. Neurosci. 27, 307–340 (2004)
26. Mozer, M.C.: Neural network composition by prediction: Exploring the benefits of psychophysical constraints and multiscale processing. Cognitive Science 6, 247–280 (1994)
27. Palmer, C., Pfordresher, P.: Incremental planning in sequence production. Psychological Review 110, 683–712 (2003)
28. Pasemann, F.: Characterization of periodic attractors in neural ring networks. Neural Networks 8(3), 421–429 (1995)
29. Pongas, D., Billard, A., Schaal, S.: Rapid synchronization and accurate phase-locking of rhythmic motor primitives. In: Proc. of 2005 IEEE Conf. on Intelligent Robotics and Systems (IROS 2005), pp. 2911–2916 (2005)
30. Stevens, C., Wiles, J.: Representations of tonal music: A case study in the development of temporal relationship. In: Mozer, M., Smolensky, P., Touretsky, D., Elman, J., Weigend, A.S. (eds.) Proceedings of the 1993 Connectionist Models Summer School, pp. 228–235. Erlbaum, Hillsdale (1994)
31. Thaut, M., Kenyon, G., Schauer, M., McIntosh, G.: The connection between rhythmicity and brain function. IEEE Engineering in Medicine and Biology Magazine 18(2), 101–108 (1999)
32. Todd, P.M.: A connectionist approach to algorithmic composition. Computer Music Journal 13(4), 27–43 (1989)

33. van Gelder, T.: The dynamical hypothesis in cognitive science. Behavioural and Brain Sciences 21(5), 615–628 (1998)
34. van Gelder, T., Port, R. (eds.): Mind as Motion: Explorations in the Dynamics of Cognition. Bradford/MIT Press (1995)
35. White, O.L., Lee, D.D., Sompolinsky, H.S.: Short-term memory in orthogonal neural networks. Phys. Rev. Lett. 92(14), 102–148 (2004)
36. Widrow, B., Lehr, M.A.: Perceptrons, adalines, and backpropagation. In: Arbib, M. (ed.) The Handbook of Brain Theory and Neural Networks, pp. 719–724. MIT Press/Bradford Books (1995)
37. Williams, R.J., Zipser, D.: Gradient-based learning algorithms for recurrent networks and their computational complexity. In: Chauvin, Y., Rumelhart, D.E. (eds.) Back-propagation: Theory, Architectures and Applications, pp. 433–486. Erlbaum, Hillsdale (1992)
38. Yang, W., Chong, N.Y.: Goal-directed imitation with self-adjusting adaptor based on a neural oscillator network. In: Proceedings, 12th International Conference on Advanced Robotics, ICAR 2005, pp. 404–410 (2005)

Appendix A: Details of the CAPAS Architecture

Error Integration and Voting

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

$$\mathrm{MSE}_j(n) = \|\mathbf{y}_i(n-1) - \mathbf{u}(n)\|^2/p, \tag{6}$$

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

$$\text{MSE-int}_j(n) = \tanh((1-\gamma_1)\ \text{MSE-int}_j(n-1) + \alpha_1\ \mathrm{MSE}_j(n)), \tag{7}$$

where γ_1 is a leaking rate and α_1 is an accumulation weight. The tanh wrapper makes the integrated prediction error saturate at 1.

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 grow away from 0 and toward 1.

An obvious mechanism to "vote" for outputs which should be fed back into the input channel would be to weigh outputs by prediction confidences $C_j(n) = 1 - \text{MSE-int}_j(n)$, average across the weighted outputs, and feed the 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 (e.g., if a 10-periodic cue is made up from

three identical repetitions of a length-3 pattern followed by a singleton point, the delay-3 outputs will have zero error during 6 out of 10 times per 10-period, and the integrated error for the 3-delay outputs will be lower than for other delays). 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 $C_j(n) = 1 - \text{MSE-int}_j(n)$ are thresholded at their lower end such that confidences falling below a threshold ε will be zeroed, that is, instead of $C_j(n) = 1 - \text{MSE-int}_j(n)$ we use

$$C_j(n) = s(1 - \text{MSE-int}_j(n)), \tag{8}$$

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. Finally, a further leaky integration and subsequent normalization smoothes $C_j(n)$ to obtain the final votes V_j:

$$\begin{aligned} \tilde{V}_j(n) &= (1 - \gamma_2)\, V_j(n-1) + \alpha_2\, C_j(n) \\ V_j(n) &= \tilde{V}_j(n) / \sum_{j=1,\ldots,d} \tilde{V}_j(n) \end{aligned} \tag{9}$$

where again γ_2, α_2 are forgetting / accumulation factors.

Feeding Back the Vote-Combined Output

The second phase of the cueing input consists of a small number of repetitions of a motif of period length k. During the presentation of the second and all subsequent 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 or third motif repetition, the external input is switched off and the network receives instead an input which is essentially made from a weighted combination of the network outputs:

$$\tilde{\mathbf{u}}(n) = \sum_{j=1,\ldots,d} V_j(n-1)\, \mathbf{y}_j(n-1). \tag{10}$$

A normalization is needed to help stabilizing this feedback loop. Specifically, we ensure that the space coding vectors $\mathbf{b}(n)$ (if they were to be recomputed from the scaled versions $\mathbf{u}(n)$) sum to 1. To effect this, we first retransform $\tilde{\mathbf{u}}(n)$ to the binary format of $\mathbf{b}(n)$, then normalize to unit component sum, and then scale/shift back to the format generated by the ESN:

$$\begin{aligned} \tilde{\tilde{\mathbf{u}}}(n) &= (\tilde{\mathbf{u}}(n) - [0.1 \ldots 0.1]^{\mathsf{T}})/0.8 \\ \tilde{\tilde{\tilde{\mathbf{u}}}}(n) &= \tilde{\tilde{\mathbf{u}}}(n)/\text{component sum of } \tilde{\tilde{\mathbf{u}}}(n) \\ \mathbf{u}(n) &= 0.8\, \tilde{\tilde{\tilde{\mathbf{u}}}}(n) + 0.1 \end{aligned} \tag{11}$$

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

Appendix B: Details of the First Simulation Study

The learning task is that the trained network, on an input sequence $\mathbf{u}(n)$, outputs d vectors $[\mathbf{u}(n-d)\mathbf{u}(n-d+1)\dots\mathbf{u}(n-1)]$. For convencience we arrange the target vectors $\mathbf{u}(n-d), \mathbf{u}(n-d+1), \dots, \mathbf{u}(n-1)$ into a $p \times d$ array $\mathbf{y}(n)$. The training data are thus generated as follows:

1. To make the input data, produce a random space-coded "melody" sequence $\mathbf{u}(n)$ of length $n_{\max}$.
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 the purpose of testing the delay memory performance.

We use an ESN with a reservoir of $N = 100$ units, with $p = 10$ input units and $d = 10$ delays, resulting in 100 output units. The internal weights $\mathbf{W}$ 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 $[-1, 1]$.

The ESN was trained on the delay-line memory task in the standard fashion described, for instance, in [15]. The length of the training sequence was 1000, from which the first 200 points were discarded to wash out initial transients. The result of the training are d output weight matrices $\mathbf{W}^{\text{in}}_j$. To prevent overfitting, a regularizer was implemented in the form of uniform noise from $[-0.0005, 0.0005]$ added to the network states harvested while the ESN is driven by the training input.

The various control parameters for the voting dynamics were set to $\gamma_1 = .4$, $\alpha_1 = 4, \gamma_2 = .2, \alpha_2 = 4, \varepsilon = 0.3$.

Appendix C: Details of the Second Simulation Study

In the second study, for training a network of size $N = 800, 1600, \dots, 4000$ we used training sequences of length $n_{\max} = 2.25 \cdot N$ and testing sequences of length $1.5 \cdot N$.

The reservoir weight matrices $\mathbf{W}$ were sparse with a connectivity that made one neuron connect to ten on average. The nonzero weights were sampled from a uniform distribution centered at 0, and the resulting matrix was rescaled such that the resulting spectral radius was 0.995 (spectral radii close to unity are beneficial for long short-term memories, see [15]). Each input unit was connected to all reservoir units with weights sampled from a uniform distribution over $[0, 1]$.

The ESN was trained using the standard procedure which is variously documented in the literature (e.g., [15]). Data from the first N time steps were discarded to account for initial transients. The linear regression was computed

using the Wiener-Hopf equation, which was regularized with a Tikhonov (also known as ridge regression) regularizer of size $\alpha = 0.0001$.

Using Matlab for the computations, we encountered spurious problems when the inverses of the Tikhonov-regularized reservoir state correlation matrices were computed for the Wiener-Hopf solution of the linear regression. The correlation matrices were generally not very well conditioned (Matlab's inverse condition indicator rcond was about 1.0e-14) but still suffienct for a reliable inversion. However, especially for the larger matrices (of sizes 2400×2400 to 4000×4000), Matlab sometimes issued a warning that the matrix was ill-conditioned, and reported rcond's of much smaller size (dozens of orders of magnitude smaller). Given that these matrices were Tikhonov-regularized with $\alpha = 0.0001$, we could not explain this phenomenon, and must assume some instability in the implementation of matrix inverses. When this situation was encountered, the trial was re-started with a freshly created ESN.

The control parameters for the voting dynamics were set to $\gamma_1 = 0.05, \alpha_1 = 2, \gamma_2 = 0.1, \alpha_2 = 2, \varepsilon = 0.2$ for all networks. In the "garden path" example, the settings were $\gamma_1 = 0.2, \alpha_1 = 4, \gamma_2 = 0.1, \alpha_2 = 4, \varepsilon = 0.2$.