JACOBS
UNIVERSITY

arXiv:1403.3369v1 [cs.NE] 13 Mar 2014

Herbert Jaeger

Controlling Recurrent Neural Networks by Conceptors

Technical Report No. 31
March 2014

School of Engineering and Science

Controlling Recurrent Neural Networks by Conceptors

Herbert Jaeger

Jacobs University Bremen
School of Engineering and Science
Campus Ring
28759 Bremen
Germany

E-Mail: h.jaeger@jacobs-university.de
http://minds.jacobs-university.de

Abstract

The human brain is a dynamical system whose extremely complex sensor-driven neural processes give rise to conceptual, logical cognition. Understanding the interplay between nonlinear neural dynamics and concept-level cognition remains a major scientific challenge. Here I propose a mechanism of neurodynamical organization, called *conceptors*, which unites nonlinear dynamics with basic principles of conceptual abstraction and logic. It becomes possible to learn, store, abstract, focus, morph, generalize, de-noise and recognize a large number of dynamical patterns within a single neural system; novel patterns can be added without interfering with previously acquired ones; neural noise is automatically filtered. Conceptors help explaining how conceptual-level information processing emerges naturally and robustly in neural systems, and remove a number of roadblocks in the theory and applications of recurrent neural networks.

Notes on the Structure of this Report. This report introduces several novel analytical concepts describing neural dynamics; develops the corresponding mathematical theory under aspects of linear algebra, dynamical systems theory, and formal logic; introduces a number of novel learning, adaptation and control algorithms for recurrent neural networks; demonstrates these in a number of case studies; proposes biologically (not too im-)plausible realizations of the dynamical mechanisms; and discusses relationships to other work. Said shortly, it's long. Not all parts will be of interest to all readers. In order to facilitate navigation through the text and selection of relevant components, I start with an overview section which gives an intuitive explanation of the novel concepts and informal sketches of the main results and demonstrations (Section 1). After this overview, the material is presented in detail, starting with an introduction (Section 2) which relates this contribution to other research. The main part is Section 3, where I systematically develop the theory and algorithms, interspersed with simulation demos. A graphical dependency map for this section is given at the beginning of Section 3. The technical documentation of the computer simulations is provided in Section 4, and mathematical proofs are collected in Section 5. The detailed presentation in Sections 2 – 5 is self-contained. Reading the overview in Section 1 may be helpful but is not necessary for reading these sections. For convenience some figures from the overview section are repeated in Section 3.

Acknowledgements. The work described in this report was partly funded through the European FP7 project AMARSi (www.amarsi-project.eu). The author is indebted to Dr. Mathieu Galtier and Dr. Manjunath Ghandi for careful proofreading (not an easy task).

Contents

1 Overview

Scientific context. Research on brains and cognition unfolds in two directions. *Top-down* oriented research starts from the "higher" levels of cognitive performance, like rational reasoning, conceptual knowledge representation, command of language. These phenomena are typically described in symbolic formalisms developed in mathematical logic, artificial intelligence (AI), computer science and linguistics. In the *bottom-up* direction, one departs from "low-level" sensor data processing and motor control, using the analytical tools offered by dynamical systems theory, signal processing and control theory, statistics and information theory. The human brain obviously has found a way to implement high-level logical reasoning on the basis of low-level neuro-dynamical processes. How this is possible, and how the top-down and bottom-up research directions can be united, has largely remained an open question despite long-standing efforts in neural networks research and computational neuroscience [80, 87, 33, 2, 36, 43], machine learning [35, 47], robotics [11, 81], artificial intelligence [83, 104, 8, 10], dynamical systems modeling of cognitive processes [94, 98, 105], cognitive science and linguistics [22, 96], or cognitive neuroscience [5, 26].

Summary of contribution. Here I establish a fresh view on the neuro-symbolic integration problem. I show how dynamical neural activation patterns can be characterized by certain neural filters which I call *conceptors*. Conceptors derive naturally from the following key observation. When a recurrent neural network (RNN) is actively generating, or is passively being driven by different dynamical patterns (say $a, b, c, \ldots$), its neural states populate different regions $R_a, R_b, R_c, \ldots$ of neural state space. These regions are characteristic of the respective patterns. For these regions, neural filters $C_a, C_b, C_c, \ldots$ (the conceptors) can be incrementally learnt. A conceptor C_x representing a pattern x can then be invoked after learning to constrain the neural dynamics to the state region R_x, and the network will select and re-generate pattern x. Learnt conceptors can be blended, combined by Boolean operations, specialized or abstracted in various ways, yielding novel patterns on the fly. Conceptors can be economically represented by single neurons (addressing patterns by neurons, leading to explicit command over pattern generation), or they may be constituted spontaneously upon the presentation of cue patterns (content-addressing, leading to pattern imitation). The logical operations on conceptors admit a rigorous semantical interpretation; conceptors can be arranged in conceptual hierarchies which are structured like semantic networks known from artificial intelligence. Conceptors can be economically implemented by single neurons (addressing patterns by neurons, leading to explicit command over pattern generation), or they may self-organize spontaneously and quickly upon the presentation of cue patterns (content-addressing, leading to pattern imitation). Conceptors can also be employed to "allocate free memory space" when new patterns are learnt and stored in long-term memory, enabling incremental

life-long learning without the danger of freshly learnt patterns disrupting already acquired ones. Conceptors are robust against neural noise and parameter variations. The basic mechanisms are generic and can be realized in any kind of dynamical neural network. All taken together, conceptors offer a principled, transparent, and computationally efficient account of how neural dynamics can self-organize in conceptual structures.

Going bottom-up: from neural dynamics to conceptors. The neural model system in this report are standard recurrent neural networks (RNNs, Figure 1 **A**) whose dynamics is mathematically described be the state update equations

$$\begin{aligned} x(n+1) &= \tanh(W^* x(n) + W^{\text{in}} p(n)), \\ y(n) &= W^{\text{out}} x(n). \end{aligned}$$

Time here progresses in unit steps $n = 1, 2, \ldots$. The network consists of N neurons (typically in the order of a hundred in this report), whose activations $x_1(n), \ldots, x_N(n)$ at time n are collected in an N-dimensional *state vector* $x(n)$. The neurons are linked by random synaptic connections, whose strengths are collected in a *weight matrix* W^* of size $N \times N$. An input signal $p(n)$ is fed to the network through synaptic input connections assembled in the *input weight* matrix W^{in}. The "S-shaped" function tanh squashes the neuronal activation values into a range between -1 and 1. The second equation specifies that an *ouput signal* $y(n)$ can be read from the network activation state $x(n)$ by means of *output weights* W^{out}. These weights are pre-computed such that the output signal $y(n)$ just repeats the input signal $p(n)$. The output signal plays no functional role in what follows; it merely serves as a convenient 1-dimensional observer of the high-dimensional network dynamics.

The network-internal neuron-to-neuron connections W^* are created at random. This will lead to the existence of cyclic ("recurrent") connection pathways inside the network. Neural activation can reverberate inside the network along these cyclic pathways. The network therefore can autonomously generate complex neurodynamical patterns even when it receives no input. Following the terminology of the *reservoir computing* [56, 6], I refer to such randomly connected neural networks as *reservoirs*.

For the sake of introducing conceptors by way of an example, consider a reservoir with $N = 100$ neurons. I drive this system with a simple sinewave input $p(n)$ (first panel in first row in Fig. 1 **B**). The reservoir becomes entrained to this input, each neuron showing individual variations thereof (Fig. 1 **B** second panel). The resulting reservoir state sequence $x(1), x(2), \ldots$ can be represented as a cloud of points in the 100-dimensional reservoir state space. The dots in the first panel of Fig. 1 **C** show a 2-dimensional projection of this point cloud. By a statistical method known as principal component analysis, the shape of this point cloud can be captured by an N-dimensional ellipsoid whose main axes point in the main

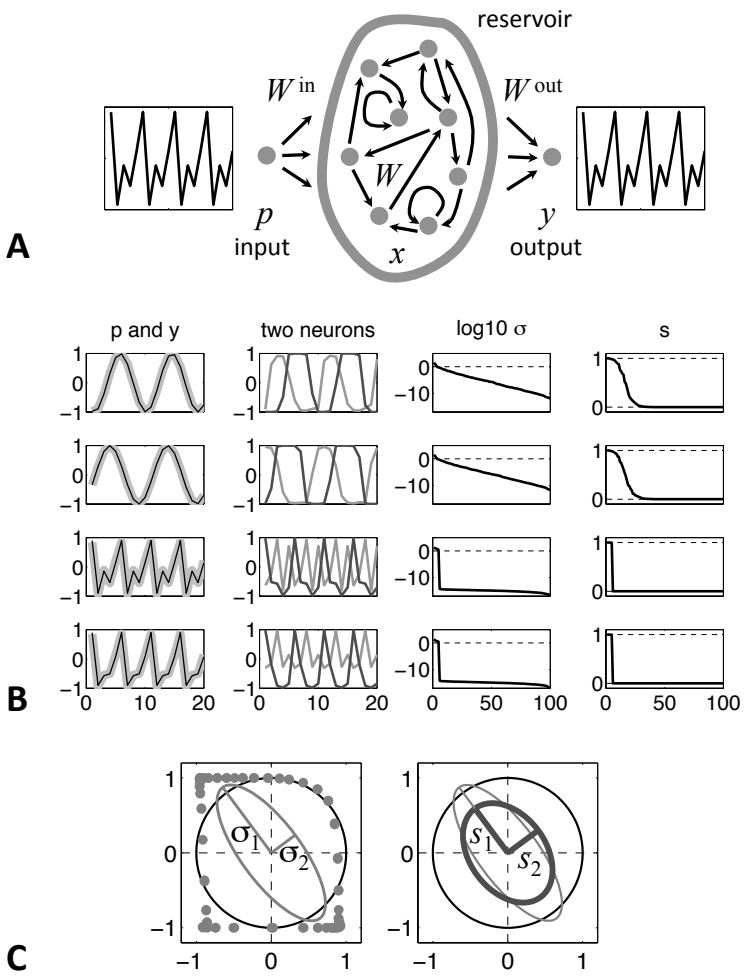

Figure 1: Deriving conceptors from network dynamics. **A.** Network layout. Arrows indicate synaptic links. **B.** Driving the reservoir with four different input patterns. Left panels: 20 timesteps of input pattern $p(n)$ (black thin line) and conceptor-controlled output $y(n)$ (bold light gray). Second column: 20 timesteps of traces $x_i(n), x_j(n)$ of two randomly picked reservoir neurons. Third column: the singular values σ_i of the reservoir state correlation matrix R in logarithmic scale. Last column: the singular values s_i of the conceptors C in linear plotting scale. **C.** From pattern to conceptor. Left: plots of value pairs $x_i(n), x_j(n)$ (dots) of the two neurons shown in first row of **B** and the resulting ellipse with axis lengths σ_1, σ_2. Right: from R (thin light gray) to conceptor C (bold dark gray) by normalizing axis lengths σ_1, σ_2 to s_1, s_2.

scattering directions of the point cloud. This ellipsoid is a geometrical representation of the *correlation matrix* R of the state points. The lengths $\sigma_1, \ldots, \sigma_N$ of the ellipsoid axes are known as the *singular values* of R. The directions and lengths of these axes provide a succinct characterization of the geometry of the state point cloud. The $N = 100$ lengths σ_i resulting in this example are log-plotted in Fig. 1 **B**, third column, revealing an exponential fall-off in this case.

As a next step, these lengths σ_i are normalized to become $s_i = \sigma_i/(\sigma_i + \alpha^{-2})$, where $\alpha \geq 0$ is a design parameter that I call *aperture*. This normalization ensures that all s_i are not larger than 1 (last column in Fig. 1 **B**). A new ellipsoid is obtained (Fig. 1 **C** right) which is located inside the unit sphere. The normalized ellipsoid can be described by a N-dimensional matrix C, which I call a *conceptor* matrix. C can be directly expressed in terms of R by $C = R(R + \alpha^{-2}I)^{-1}$, where I is the identity matrix.

When a different driving pattern p is used, the shape of the state point cloud, and subsequently the conceptor matrix C, will be characteristically different. In the example, I drove the reservoir with four patterns $p^1 - p^4$ (rows in Fig. 1**B**). The first two patterns were sines of slightly different frequencies, the last two patterns were minor variations of a 5-periodic random pattern. The conceptors derived from the two sine patterns differ considerably from the conceptors induced by the two 5-periodic patterns (last column in Fig. 1**B**). Within each of these two pairs, the conceptor differences are too small to become visible in the plots.

There is an instructive alternative way to define conceptors. Given a sequence of reservoir states $x(1), \ldots, x(L)$, the conceptor C which characterizes this state point cloud is the unique matrix which minimizes the cost function $\sum_{n=1,\ldots,L} \|x(n) - Cx(n)\|^2/L + \alpha^{-2}\|C\|^2$, where $\|C\|^2$ is the sum of all squared matrix entries. The first term in this cost would become minimal if C were the identity map, the second term would become minimal if C would be the all-zero map. The aperture α strikes a balance between these two competing cost components. For increasing apertures, C will tend toward the identity matrix I; for shrinking apertures it will come out closer to the zero matrix. In the terminology of machine learning, C is hereby defined as a *regularized identity map*. The explicit solution to this minimization problem is again given by the formula $C = R\,(R + \alpha^{-2}I)^{-1}$.

Summing up: if a reservoir is driven by a pattern $p(n)$, a conceptor matrix C can be obtained from the driven reservoir states $x(n)$ as the regularized identity map on these states. C can be likewise seen as a normalized ellipsoid characterization of the shape of the $x(n)$ point cloud. I write $C(p, \alpha)$ to denote a conceptor derived from a pattern p using aperture α, or $C(R, \alpha)$ to denote that C was obtained from a state correlation matrix R.

Loading a reservoir. With the aid of conceptors a reservoir can re-generate a number of different patterns $p^1, \ldots, p^K$ that it has previously been driven with. For this to work, these patterns have to be learnt by the reservoir in a special

sense, which I call *loading* a reservoir with patterns. The loading procedure works as follows. First, drive the reservoir with the patterns $p^1, \ldots, p^K$ in turn, collecting reservoir states $x^j(n)$ (where $j = 1, \ldots, K$). Then, recompute the reservoir connection weights W^* into W such that W optimally balances between the following two goals. First, W should be such that $W\,x^j(n) \approx W^*\,x^j(n) + W^{\text{in}}p^j(n)$ for all times n and patterns j. That is, W should allow the reservoir to "simulate" the driving input in the absence of the same. Second, W should be such that the weights collected in this matrix become as small as possible. Technically this compromise-seeking learning task amounts to computing what is known as a regularized linear regression, a standard and simple computational task. This idea of "internalizing" a driven dynamics into a reservoir has been independently (re-)introduced under different names and for a variety of purposes (*self-prediction* [72], *equilibration* [55], *reservoir regularization* [90], *self-sensing networks* [100], *innate training* [61]) and appears to be a fundamental RNN adaptation principle.

Going top-down: from conceptors to neural dynamics. Assume that conceptors $C^j = C(p^j, \alpha)$ have been derived for patterns $p^1, \ldots, p^K$, and that these patterns have been loaded into the reservoir, replacing the original random weights W^* by W. Intuitively, the loaded reservoir, when it is run using $x(n+1) = tanh(W\,x(n))$ (no input!) should behave exactly as when it was driven with input earlier, because W has been trained such that $W\,x(n) \approx W^*\,x(n) + W^{\text{in}}p^j(n)$. In fact, if only a single pattern had been loaded, the loaded reservoir would readily re-generate it. But if more than one patter had been loaded, the autonomous (input-free) update $x(n+1) = tanh(W\,x(n))$ will lead to an entirely unpredictable dynamics: the network can't "decide" which of the loaded patterns it should re-generate! This is where conceptors come in. The reservoir dynamics is filtered through C^j. This is effected by using the augmented update rule $x(n+1) = C^j\,tanh(W\,x(n))$. By virtue of inserting C^j into the feedback loop, the reservoir states become clipped to fall within the ellipsoid associated with C^j. As a result, the pattern p^j will be re-generated: when the reservoir is observed through the previously trained output weights, one gets $y(n) = W^{\text{out}}\,x(n) \approx p^j(n)$. The first column of panels in Fig. 1 **B** shows an overlay of the four autonomously re-generated patterns $y(n)$ with the original drivers p^j used in that example. The recovery of the originals is quite accurate (mean square errors 3.3e-05, 1.4e-05, 0.0040, 0.0019 for the four loaded patterns). Note that the first two and the last two patterns are rather similar to each other. The filtering afforded by the respective conceptors is "sharp" enough to separate these twin pairs. I will later demonstrate that in this way a remarkably large number of patterns can be faithfully re-generated by a single reservoir.

Morphing and generalization. Given a reservoir loaded with K patterns p^j, the associated conceptors C^j can be linearly combined by creating mixture conceptors $M = \mu^1 C^1 + \ldots + \mu^K C^K$, where the mixing coefficients μ^j must sum to 1.

When the reservoir is run under the control of such a *morphed* conceptor M, the resulting generated pattern is a morph between the original "pure" patterns p^j. If all μ^j are non-negative, the morph can be considered an *interpolation* between the pure patterns; if some μ^j are negative, the morph *extrapolates* beyond the loaded pure patterns. I demonstrate this with the four patterns used in the example above, setting $\mu^1 = (1-a)b, \mu^2 = ab, \mu^3 = (1-a)(1-b), \mu^4 = a(1-b)$, and letting a, b vary from -0.5 to 1.5 in increments of 0.25. Fig. 2 shows plots of observer signals $y(n)$ obtained when the reservoir is generating patterns under the control of these morphed conceptors. The innermost 5 by 5 panels show interpolations between the four pure patterns, all other panels show extrapolations.

In machine learning terms, both interpolation and extrapolation are cases of *generalization*. A standard opinion in the field states that generalization by interpolation is what one may expect from learning algorithms, while extrapolation beyond the training data is hard to achieve.

Morphing and generalizing dynamical patterns is a common but nontrivial task for training motor patterns in robots. It typically requires training demonstrations of numerous interpolating patterns [88, 17, 68]. Conceptor-based pattern morphing appears promising for flexible robot motor pattern learning from a very small number of demonstrations.

Aperture adaptation. Choosing the aperture α appropriately is crucial for regenerating patterns in a stable and accurate way. To demonstrate this, I loaded a 500-neuron reservoir with signals p^1 – p^4 derived from four classical chaotic attractors: the Lorenz, Rössler, Mackey-Glass, and Hénon attractors. Note that it used to be a challenging task to make an RNN learn any single of these attractors [56]; to my knowledge, training a single RNN to generate several different chaotic attractors has not been attempted before. After loading the reservoir, the regeneration was tested using conceptors $C(p^j, \alpha)$ where for each attractor pattern p^j a number of different values for α were tried. Fig. 3 **A** shows the resulting regenerated patterns for five apertures for the Lorenz attractor. When the aperture is too small, the reservoir-conceptor feedback loop becomes too constrained and the produced patterns de-differentiate. When the aperture is too large, the feedback loop becomes over-excited.

An optimal aperture can be found by experimentation, but this will not be an option in many engineering applications or in biological neural systems. An intrinsic criterion for optimizing α is afforded by a quantity that I call *attenuation*: the damping ratio which the conceptor imposes on the reservoir signal. Fig. 3 **C** plots the attenuation against the aperture for the four chaotic signals. The minimum of this curve marks a good aperture value: when the conceptor dampens out a minimal fraction of the reservoir signal, conceptor and reservoir are in good "resonance". The chaotic attractor re-generations shown in Fig. 3 **B** were obtained by using this minimum-attenuation criterion.

The aperture range which yields visibly good attractor re-generations in this

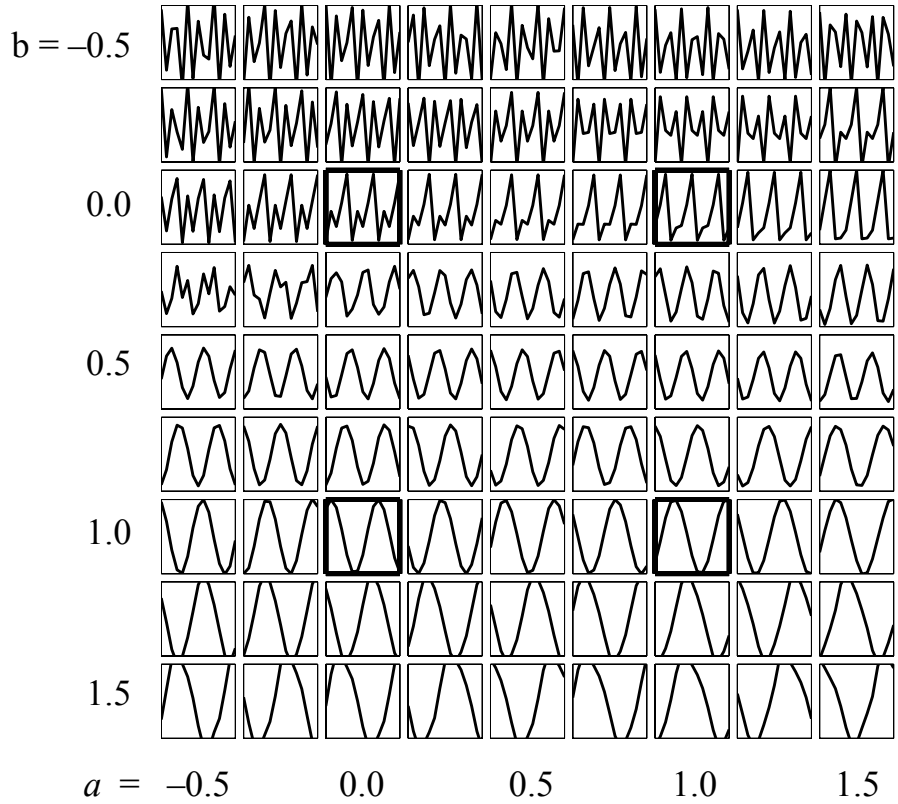

Figure 2: Morphing between, and generalizing beyond, four loaded patterns. Each panel shows a 15-step autonomously generated pattern (plot range between -1 and $+1$). Panels with bold frames: the four loaded prototype patterns (same patterns as in Fig. 1 **B**.)

demonstration spans about one order of magnitude. With further refinements (zeroing small singular values in conceptors is particularly effective), the viable aperture range can be expanded to about three orders of magnitude. While setting the aperture right is generally important, fine-tuning is unnecessary.

Boolean operations and conceptor abstraction. Assume that a reservoir is driven by a pattern r which consists of randomly alternating epochs of two patterns p and q. If one doesn't know which of the two patterns is active at a given time, all one can say is that the pattern r currently is p OR it is q. Let $C(R_p, 1), C(R_q, 1), C(R_r, 1)$ be conceptors derived from the two partial patterns p, q and the "OR" pattern r, respectively. Then it holds that $C(R_r, 1) = C((R_p + R_q)/2, 1)$. Dropping the division by 2, this motivates to define an OR (mathematical notation: $\vee$) operation on conceptors $C_1(R_1, 1), C_2(R_2, 1)$ by putting $C_1 \vee C_2 := (R_1 + R_2)(R_1 + R_2 + I)^{-1}$. The logical operations NOT ($\neg$) and AND ($\wedge$) can be defined along similar lines. Fig. 4 shows two-dimensional examples of applying the three operations.

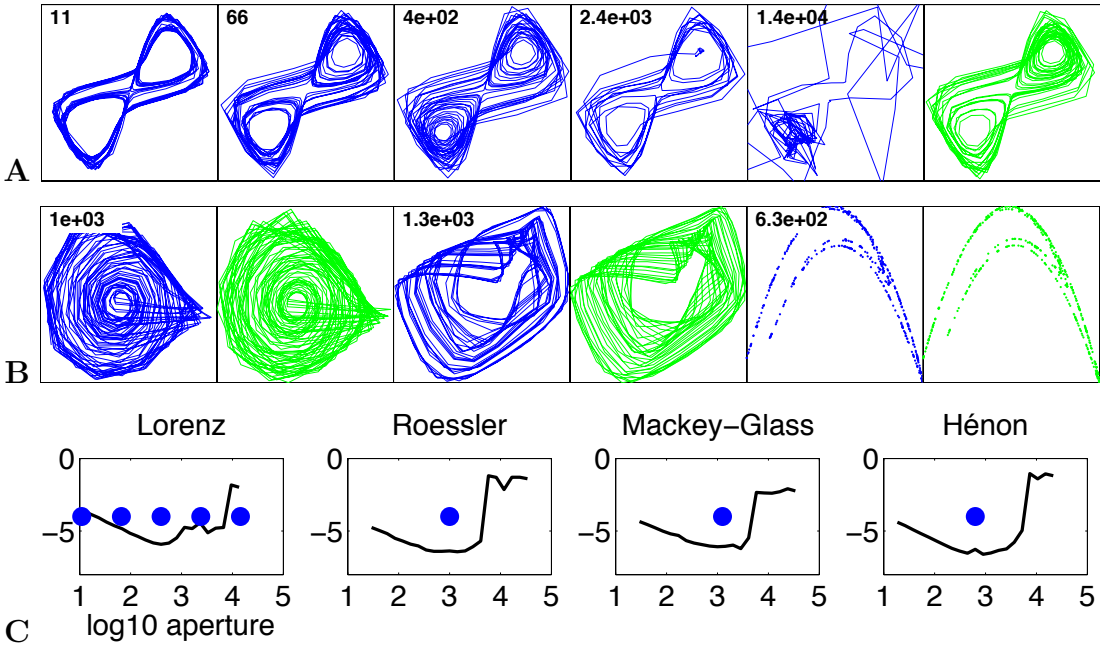

Figure 3: Aperture adaptation for re-generating four chaotic attractors. **A** Lorenz attractor. Five versions re-generated with different apertures (values inserted in panels) and original attractor (gray background). **B** Best re-generations of the other three attractors (from left to right: Rössler, Mackey-Glass, and Hénon, originals on gray background). **C** Log10 of the attenuation criterion plotted against the log10 of aperture. Dots mark the apertures used for plots in **A** and **B**.

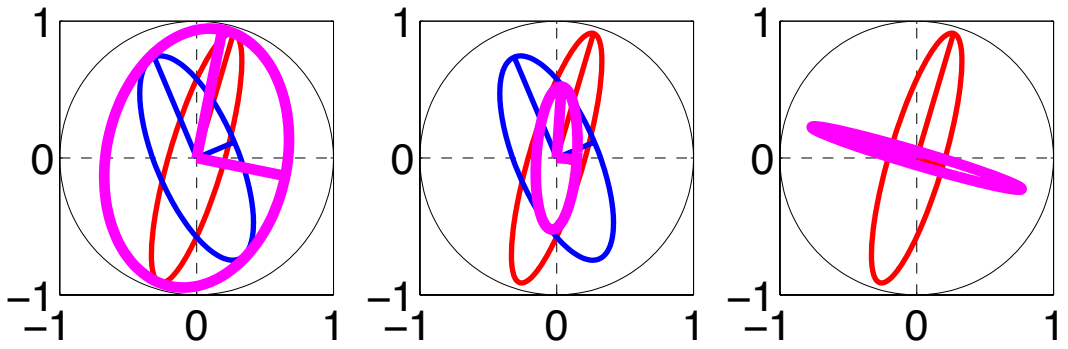

Figure 4: Boolean operations on conceptors. Red/blue (thin) ellipses represent source conceptors C_1, C_2. Magenta (thick) ellipses show $C_1 \vee C_2$, $C_1 \wedge C_2$, $\neg C_1$ (from left to right).

Boolean logic is the mathematical theory of $\vee, \wedge, \neg$. Many laws of Boolean logic also hold for the $\vee, \wedge, \neg$ operations on conceptors: the laws of associativity, commutativity, double negation, de Morgan's rules, some absorption rules. Furthermore, numerous simple laws connect aperture adaptation to Boolean operations. Last but not least, by defining $C_1 \leq C_2$ if and only if there exists a

conceptor B such that $C_2 = C_1 \vee B$, an *abstraction ordering* is created on the set of all conceptors of dimension N.

Neural memory management. Boolean conceptor operations afford unprecedented flexibility of organizing and controlling the nonlinear dynamics of recurrent neural networks. Here I demonstrate how a sequence of patterns $p^1, p^2, \ldots$ can be *incrementally* loaded into a reservoir, such that (i) loading a new pattern p^{j+1} does not interfere with previously loaded $p^1, \ldots, p^j$; (ii) if a new pattern p^{j+1} is similar to already loaded ones, the redundancies are automatically detected and exploited, saving memory capacity; (iii) the amount of still "free" memory space can be logged.

Let C^j be the conceptor associated with pattern p^j. Three ideas are combined to implement the memory management scheme. First, keep track of the "already used" memory space by maintaining a conceptor $A^j = C^1 \vee \ldots \vee C^j$. The sum of all singular values of A^j, divided by the reservoir size, gives a number that ranges between 0 and 1. It is an indicator of the portion of reservoir "space" which has been used up by loading $C^1, \ldots, C^j$, and I call it the *quota* claimed by $C^1, \ldots, C^j$. Second, characterize what is "new" about C^{j+1} (not being already represented by previously loaded patterns) by considering the conceptor $N^{j+1} = C^{j+1} \setminus A^j$. The *logical difference* operator $\setminus$ can be re-written as $A \setminus B = A \wedge \neg B$. Third, load only that which is new about C^{j+1} into the still unclaimed reservoir space, that is, into $\neg A^j$. These three ideas can be straightforwardly turned into a modification of the basic pattern loading algorithm.

For a demonstration, I created a series of periodic patterns $p^1, p^2, \ldots$ whose integer period lengths were picked randomly between 3 and 15, some of these patterns being sines, others random patterns. These patterns were incrementally loaded in a 100-neuron reservoir, one by one. Fig. 5 shows the result. The "used space" panels monitor the successive filling-up of reservoir space. Since patterns $j = 6, 7, 8$ were identical replicas of patterns $j = 1, 2, 3$, no additional space was consumed when these patterns were (re-)loaded. The "driver and y" panels document the accuracy of autonomously re-generating patterns using conceptors C^j. Accuracy was measured by the *normalized root mean square error* (NRMSE), a standard criterion for comparing the similarity between two signals. The NRMSE jumps from very small values to a high value when the last pattern is loaded; the quota of 0.98 at this point indicates that the reservoir is "full". The re-generation testing and NRMSE computation was done after all patterns had been loaded. An attempt to load further patterns would be unsuccessful, but it also would not harm the re-generation quality of the already loaded ones.

This ability to load patterns incrementally solves a notorious problem in neural network training, known as *catastrophic forgetting*, which manifests itself in a disruption of previously learnt functionality when learning new functionality. Although a number of proposals have been made which partially alleviate the problem in special circumstances [32, 42], catastrophic forgetting was still listed

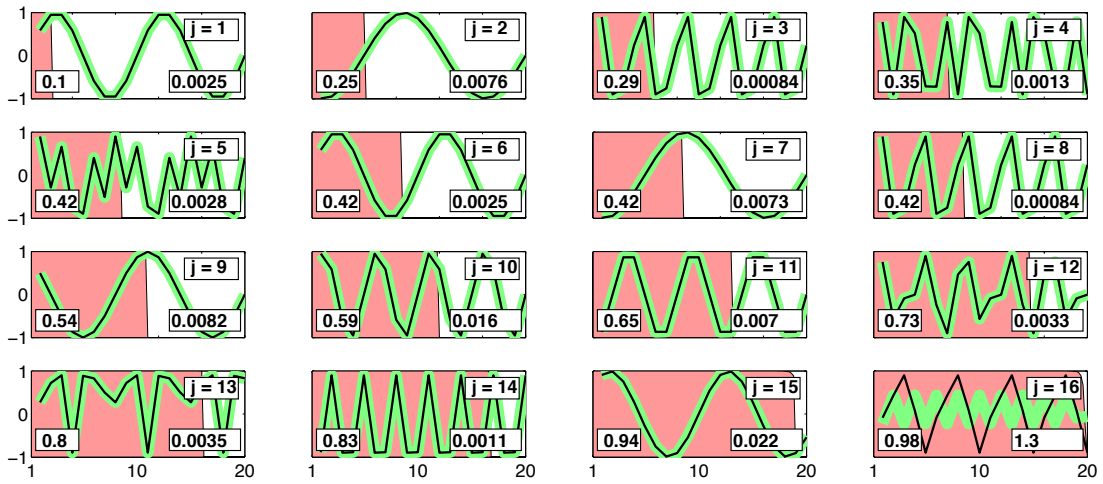

Figure 5: Incremental pattern storing in a neural memory. Each panel shows a 20-timestep sample of the correct training pattern p^j (black line) overlaid on its reproduction (green line). The memory fraction used up until pattern j is indicated by the panel fraction filled in red; the quota value is printed in the left bottom corner of each panel.

as an open challenge in an expert's report solicited by the NSF in 2007 [21] which collected the main future challenges in learning theory.

Recognizing dynamical patterns. Boolean conceptor operations enable the combination of positive and negative evidence in a neural architecture for dynamical pattern recognition. For a demonstration I use a common benchmark, the *Japanese vowel* recognition task [60]. The data of this benchmark consist in preprocessed audiorecordings of nine male native speakers pronouncing the Japanese di-vowel /ae/. The training data consist of 30 recordings per speaker, the test data consist of altogether 370 recordings, and the task is to train a recognizer which has to recognize the speakers of the test recordings. This kind of data differs from the periodic or chaotic patterns that I have been using so far, in that the patterns are non-stationary (changing in their structure from beginning to end), multi-dimensional (each recording consisting of 12 frequency band signals), stochastic, and of finite duration. This example thus also demonstrates that conceptors can be put to work with data other than single-channel stationary patterns.

A small (10 neurons) reservoir was created. It was driven with all training recordings from each speaker j in turn ($j = 1, \ldots, 9$), collecting reservoir response signals, from which a conceptor C^j characteristic of speaker j was computed. In addition, for each speaker j, a conceptor $N^j = \neg\,(C^1 \vee \ldots \vee C^{j-1} \vee C^{j+1} \vee \ldots C^9)$ was computed. N^j characterizes the condition "this speaker is not any of the other eight speakers". Patterns need not to be loaded into the reservoir for this application, because they need not be re-generated.

In testing, a recording p from the test set was fed to the reservoir, collecting a reservoir response signal x. For each of the conceptors, a *positive evidence* $E^+(p,j) = x'C^jx$ was computed. $E^+(p,j)$ is a non-negative number indicating how well the signal x fits into the ellipsoid of C^j. Likewise, the *negative evidence* $E^-(p,j) = x'N^jx$ that the sample p was not uttered by any of the eight speakers other than speaker j was computed. Finally, the *combined evidence* $E(p,j) = E^+(p,i) + E^-(p,i)$ was computed. This gave nine combined evidences $E(p,1), \ldots, E(p,9)$. The pattern p was then classified as speaker j by choosing the speaker index j whose combined evidence $E(p,j)$ was the greatest among the nine collected evidences.

In order to check for the impact of the random selection of the underlying reservoir, this whole procedure was repeated 50 times, using a freshly created random reservoir in each trial. Averaged over these 50 trials, the number of test misclassifications was 3.4. If the classification would have been based solely on the positive or negative evidences, the average test misclassification numbers would have been 8.4 and 5.9 respectively. The combination of positive and negative evidence, which was enabled by Boolean operations, was crucial.

State-of-the-art machine learning methods achieve between 4 and 10 misclassifications on the test set (for instance [91, 97, 79, 15]). The Boolean-logic-conceptor-based classifier thus compares favorably with existing methods in terms of classification performance. The method is computationally cheap, with the entire learning procedure taking a fraction of a second only on a standard notebook computer. The most distinctive benefit however is incremental extensibility. If new training data become available, or if a new speaker would be incorporated into the recognition repertoire, the additional training can be done using only the new data without having to re-run previous training data. This feature is highly relevant in engineering applications and in cognitive modeling and missing from almost all state-of-the-art classification methods.

Autoconceptors and content-addressable memories. So far I have been describing examples where conceptors C^j associated with patterns p^j were computed at training time, to be later plugged in to re-generate or classify patterns. A conceptor C matrix has the same size as the reservoir connection matrix W. Storing conceptor matrices means to store network-sized objects. This is implausible under aspects of biological modeling. Here I describe how conceptors can be created on the fly, without having to store them, leading to content-addressable neural memories.

If the system has no pre-computed conceptors at its disposal, loaded patterns can still be re-generated in a two-stage process. First, the target pattern p is selected by driving the system with a brief initial "cueing" presentation of the pattern (possibly in a noisy version). During this phase, a preliminary conceptor C^{cue} is created by an online adaptation process. This preliminary C^{cue} already enables the system to re-generate an imperfect version of the pattern p. Second, after

the cueing phase has ended, the system continues to run in an autonomous mode (no external cue signal), initially using C^{cue}, to continuously generate a pattern. While this process is running, the conceptor in the loop is continuously adapted by a simple online adaptation rule. This rule can be described in geometrical terms as "adapt the current conceptor $C(n)$ such that its ellipsoid matches better the shape of the point cloud of the current reservoir state dynamics". Under this rule one obtains a reliable convergence of the generated pattern toward a highly accurate replica of the target pattern p that was given as a cue.

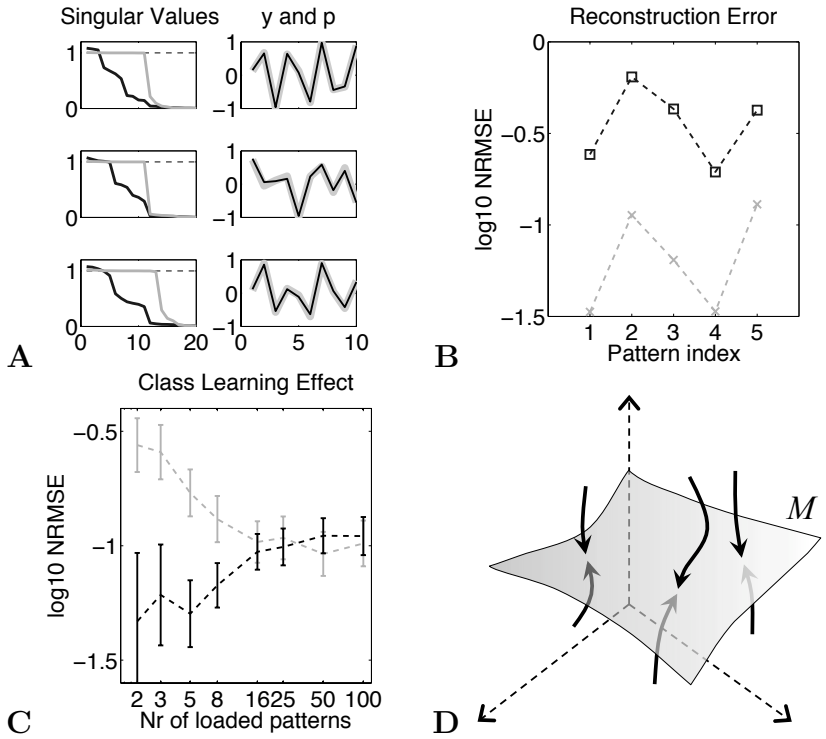

Figure 6: Content-addressable memory. **A** First three of five loaded patterns. Left panels show the leading 20 singular values of C^{cue} (black) and C^{auto} (gray). Right panels show an overlay of the original driver pattern (black, thin) and the reconstruction at the end of auto-adaptation (gray, thick). **B** Pattern reconstruction errors directly after cueing (black squares) and at end of auto-adaptation (gray crosses). **C** Reconstruction error of loaded patterns (black) and novel patterns drawn from the same parametric family (gray) versus the number of loaded patterns, averaged over 5 repetitions of the entire experiment and 10 patterns per plotting point. Error bars indicate standard deviations. **D** Autoconceptor adaptation dynamics described as evolution toward a plane attractor M (schematic).

Results of a demonstration are illustrated in Figure 6. A 200-neuron reservoir was loaded with 5 patterns consisting of a weighted sum of two irrational-period sines, sampled at integer timesteps. The weight ratio and the phaseshift were chosen at random; the patterns thus came from a family of patterns parametrized

by two parameters. The cueing time was 30 timesteps, the free-running auto-adaptation time was 10,000 timesteps, leading to an auto-adapted conceptor C^{auto} at the end of this process. On average, the reconstruction error improved from about -0.4 (log10 NRMSE measured directly after the cueing) to -1.1 (at the end of auto-adaptation). It can be shown analytically that the auto-adaptation process pulls many singular values down to zero. This effect renders the combined reservoir-conceptor loop very robust against noise, because all noise components in the directions of the nulled singular values become completely suppressed. In fact, all results shown in Figure 6 were obtained with strong state noise (signal-to-noise ratio equal to 1) inserted into the reservoir during the post-cue auto-adaptation.

The system functions as a *content-addressable memory* (CAM): loaded items can be recalled by cueing them. The paradigmatic example of a neural CAM are auto-associative neural networks (AANNs), pioneered by Palm [80] and Hopfield [48]. In contrast to conceptor-based CAM, which store and re-generate dynamical patterns, AANNs store and cue-recall static patterns. Furthermore, AANNs do not admit an incremental storing of new patterns, which is possible in conceptor-based CAMs. The latter thus represent an advance in neural CAMs in two fundamental aspects.

To further elucidate the properties of conceptor CAMs, I ran a suite of simulations where the same reservoir was loaded with increasing numbers of patterns, chosen at random from the same 2-parametric family (Figure 6 **C**). After loading with $k = 2, 3, 5, \ldots, 100$ patterns, the reconstruction accuracy was measured at the end of the auto-adaptation. Not surprisingly, it deteriorated with increasing memory load k (black line). In addition, I also cued the loaded reservoir with patterns that were *not* loaded, but were drawn from the same family. As one would expect, the re-construction accuracy of these novel patterns was worse than for the loaded patterns – but only for small k. When the number of loaded patterns exceeded a certain threshold, recall accuracy became essentially equal for loaded and novel patterns. These findings can be explained in intuitive terms as follows. When few patterns are loaded, the network memorizes individual patterns by "rote learning", and subsequently can recall these patterns better than other patterns from the family. When more patterns are loaded, the network learns a representation of the entire parametric class of patterns. I call this the *class learning effect*.

The class learning effect can be geometrically interpreted in terms of a *plane attractor* [24] arising in the space of conceptor matrices C (Figure 6 **D**). The learnt parametric class of patterns is represented by a d-dimensional manifold M in this space, where d is the number of defining parameters for the pattern family (in our example, $d = 2$). The cueing procedure creates an initial conceptor C^{cue} in the vicinity of M, which is then attracted toward M by the auto-adaptation dynamics. While an in-depth analysis of this situation reveals that this picture is not mathematically correct in some detail, the plane attractor metaphor yields a good phenomenal description of conceptor CAM class learning.

Plane attractors have been invoked as an explanation for a number of biological phenomena, most prominently gaze direction control [24]. In such phenomena, points on the plane attractor correspond to *static* fixed points (for instance, a direction of gaze). In contrast, points on M correspond to conceptors which in turn define *temporal* patterns. Again, the conceptor framework "dynamifies" concepts that have previously been worked out for static patterns only.

Toward biological feasibility: random feature conceptors. Several computations involved in adapting conceptor matrices are non-local and therefore biologically infeasible. It is however possible to approximate matrix conceptors with another mechanism which only requires local computations. The idea is to project (via random projection weights F) the reservoir state into a *random feature space* which is populated by a large number of neurons z_i; execute the conceptor operations individually on each of these neurons by multiplying a *conception weight* c_i into its state; and finally to project back to the reservoir by another set of random projection weights G (Figure 7).

The original reservoir-internal random connection weigths W are replaced by a dyade of two random projections of first F, then G, and the original reservoir state x segregates into a reservoir state r and a random feature state z. The conception weights c_i assume the role of conceptors. They can be learnt and adapted by procedures which are directly analog to the matrix conceptor case. What had to be non-local matrix computations before now turns into local, one-dimensional (scalar) operations. These operations are biologically feasible in the modest sense that any information needed to adapt a synaptic weight is locally available at that synapse. All laws and constructions concerning Boolean operations and aperture carry over.

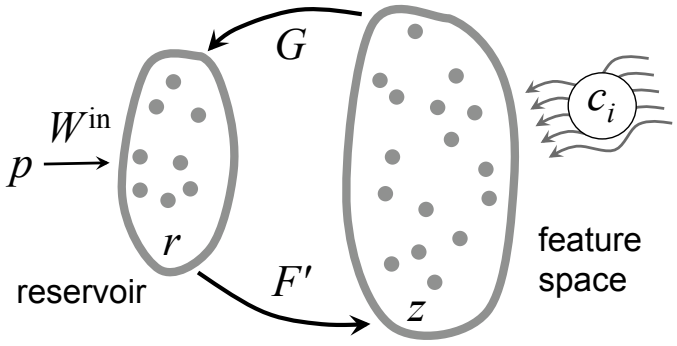

Figure 7: Random feature conceptors. This neural architecture has two pools of neurons, the reservoir and the feature space.

A set of conception weights c_i corresponding to a particular pattern can be neurally represented and "stored" in the form of the connections of a single neuron to the feature space. A dynamical pattern thus can be represented by a single neu-

ron. This enables a highly compact neural representation of dynamical patterns. A machine learning application is presented below.

I re-ran with such random feature conceptors a choice of the simulations that I did with matrix conceptors, using a number of random features that was two to five times as large as the reservoir. The outcome of these simulations: the accuracy of pattern re-generation is essentially the same as with matrix conceptors, but setting the aperture is more sensitive.

A hierarchical classification and de-noising architecture. Here I present a system which combines in a multi-layer neural architecture many of the items introduced so far. The input to this system is a (very) noisy signal which at a given time is being generated by one out of a number of possible candidate pattern generators. The task is to recognize the current generator, and simultaneously to re-generate a clean version of the noisy input pattern.

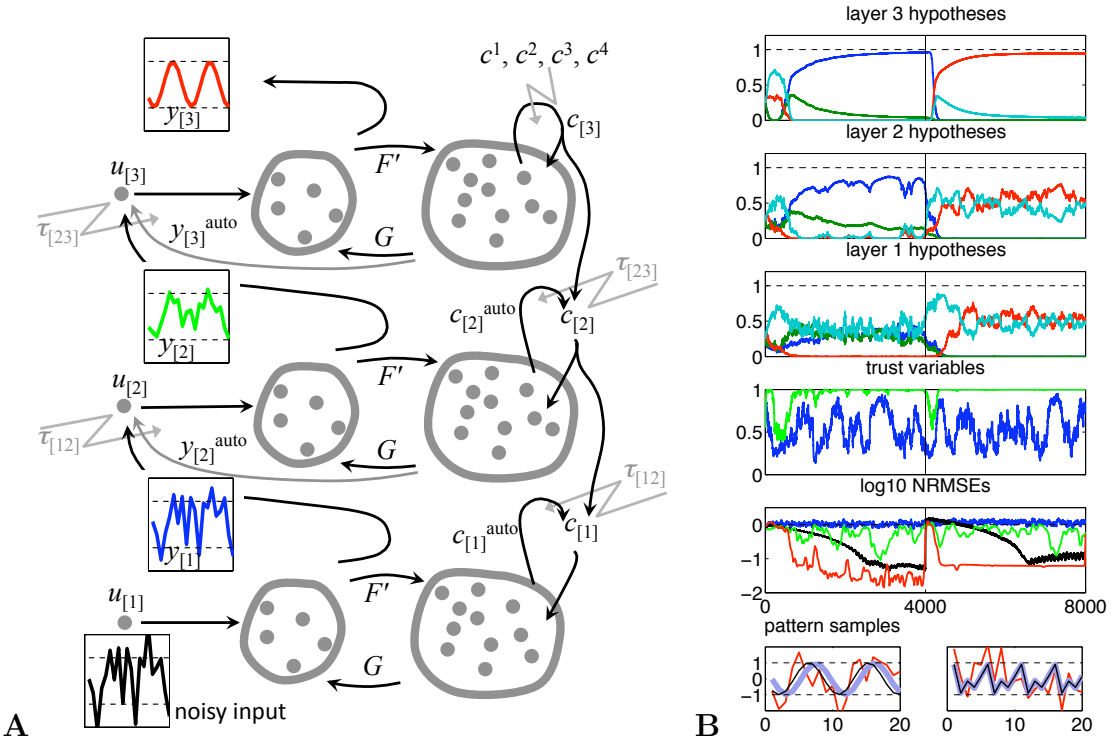

Figure 8: Simultaneous signal de-noising and classification. **A.** Schema of architecture. **B.** Simulation results. Panels from above: first three panels: hypothesis vectors $\gamma^j_{[l]}(n)$ in the three layers. Color coding: p^1 blue, p^2 green, p^3 red, p^4 cyan. Fourth panel: trust variables $\tau_{[1,2]}(n)$ (blue) and $\tau_{[2,3]}(n)$ (green). Fifth panel: signal reconstruction errors (log10 NRMSE) of $y_{[1]}$ (blue), $y_{[2]}$ (green) and $y_{[3]}$ (red) versus clean signal p^j. Black line: linear baseline filter. Bottom panels: 20-step samples from the end of the two presentation periods. Red: noisy input; black: clean input; thick gray: cleaned output signal $y_{[3]}$.

I explain the architecture with an example. It uses three processing layers to de-noise an input signal $u_{[1]}(n) = p^j(n) + noise$, with p^j being one of the four patterns $p^1, \ldots, p^4$ used before in this report (shown for instance in Figure 1 **B**). The architecture implements the following design principles (Figure 8 **A**). (i) Each layer is a random feature based conceptor system (as in Figure 7 **B**). The four patterns $p^1, \ldots, p^4$ are initially loaded into each of the layers, and four *prototype* conceptor weight vectors $c^1, \ldots, c^4$ corresponding to the patterns are computed and stored. (ii) In a bottom-up processing pathway, the noisy external input signal $u_{[1]}(n) = p^j(n) + noise$ is stagewise de-noised, leading to signals $y_{[1]}, y_{[2]}, y_{[3]}$ on layers $l = 1, 2, 3$, where $y_{[3]}$ should be a highly cleaned-up version of the input (subscripts $[l]$ refer to layers, bottom layer is $l = 1$). (iii) The top layer auto-adapts a conceptor $c_{[3]}$ which is constrained to be a weighted OR combination of the four prototype conceptors. In a suggestive notation this can be written as $c_{[3]}(n) = \gamma^1_{[3]}(n)\, c^1 \vee \ldots \vee \gamma^4_{[3]}(n)\, c^4$. The four weights $\gamma^j_{[3]}$ sum to one and represent a *hypothesis* vector expressing the system's current belief about the current driver p^j. If one of these $\gamma^j_{[3]}$ approaches 1, the system has settled on a firm classification of the current driving pattern. (iv) In a top-down pathway, conceptors $c_{[l]}$ from layers l are passed down to the respective layers $l-1$ below. Because higher layers should have a clearer conception of the current noisy driver pattern than lower layers, this passing-down of conceptors "primes" the processing in layer $l-1$ with valuable contextual information. (v) Between each pair of layers $l, l+1$, a *trust* variable $\tau_{[l,l+1]}(n)$ is adapted by an online procedure. These trust variables range between 0 and 1. A value of $\tau_{[l,l+1]}(n) = 1$ indicates maximal confidence that the signal $y_{[l+1]}(n)$ comes closer to the clean driver $p^j(n)$ than the signal $y_{[l]}(n)$ does, that is, the stage-wise denoising actually functions well when progressing from layer l to $l+1$. The trust $\tau_{[l,l+1]}(n)$ evolves by comparing certain noise ratios that are observable locally in layers l and $l+1$. (vi) Within layer l, an internal auto-adaptation process generates a candidate de-noised signal $y^{\text{auto}}_{[l]}$ and a candidate local autoconceptor $c^{\text{auto}}_{[l]}$. The local estimate $y^{\text{auto}}_{[l]}$ is linearly mixed with the signal $y_{[l-1]}$, where the trust $\tau_{[l-1,l]}$ sets the mixing rate. The mixture $u_{[l]} = \tau_{[l-1,l]}\, y^{\text{auto}}_{[l]} + (1 - \tau_{[l-1,l]})\, y_{[l-1]}$ is the effective signal input to layer l. If the trust $\tau_{[l-1,l]}$ reaches its maximal value of 1, layer l will ignore the signal from below and work entirely by self-generating a pattern. (vii) In a similar way, the effective conceptor in layer l is a trust-negotiated mixture $c_{[l]} = (1 - \tau_{[l,l+1]})\, c^{\text{auto}}_{[l]} + \tau_{[l,l+1]}\, c_{[l+1]}$. Thus if the trust $\tau_{[l,l+1]}$ is maximal, layer l will be governed entirely by the passed-down conceptor $c_{[l+1]}$.

Summarizing, the higher the trusts inside the hierarchy, the more will the system be auto-generating conceptor-shaped signals, or conversely, at low trust values the system will be strongly permeated from below by the outside driver. If the trust variables reach their maximum value of 1, the system will run in a pure "confabulation" mode and generate an entirely noise-free signal $y_{[3]}$ – at the risk of doing this under an entirely misguided hypothesis $c_{[3]}$. The key to make this architecture work thus lies in the trust variables. It seems to me that maintaining

a measure of trust (or call it confidence, certainty, etc.) is an intrinsically necessary component in any signal processing architecture which hosts a top-down pathway of guiding hypotheses (or call them context, priors, bias, etc.).

Figure 8 **B** shows an excerpt from a simulation run. The system was driven first by an initial 4000 step period of $p^1 + noise$, followed by 4000 steps of $p^3 + noise$. The signal-to-noise ratio was 0.5 (noise twice as strong as signal). The system successfully settles on the right hypothesis (top panel) and generates very clean de-noised signal versions (bottom panel). The crucial item in this figure is the development of the trust variable $\tau_{[2,3]}$. At the beginning of each 4000 step period it briefly drops, allowing the external signal to permeate upwards through the layers, thus informing the local auto-adaptation loops about "what is going on outside". After these initial drops the trust rises to almost 1, indicating that the system firmly "believes" to have detected the right pattern. It then generates pattern versions that have almost no mix-in from the noisy external driver.

As a baseline comparison I also trained a standard linear transversal filter which computed a de-noised input pattern point based on the preceding $K = 2600$ input values. The filter length K was set equal to the number of trainable parameters in the neural architecture. The performance of this linear de-noising filter (black line in Figure 8) is inferior to the architecture's performance both in terms of accuracy and response time.

It is widely believed that top-down hypothesis-passing through a processing hierarchy plays a fundamental role in biological cognitive systems [33, 16]. However, the current best artificial pattern recognition systems [39, 59] use purely bottom-up processing – leaving room for further improvement by including top-down guidance. A few hierarchical architectures which exploit top-down hypothesis-passing have been proposed [33, 47, 43, 35]. All of these are designed for recognizing static patterns, especially images. The conceptor-based architecture presented here appears to be the first hierarchical system which targets dynamical patterns and uses top-down hypothesis-passing. Furthermore, in contrast to state-of-the-art pattern recognizers, it admits an incremental extension of the pattern repertoire.

Intrinsic conceptor logic. In mathematical logics the *semantics* ("meaning") of a symbol or operator is formalized as its *extension*. For instance, the symbol cow in a logic-based knowledge representation system in AI is semantically interpreted by the set of all (physical) cows, and the OR-operator $\vee$ is interpreted as set union: $\mathsf{cow} \vee \mathsf{horse}$ would refer to the set comprising all cows and horses. Similarly, in cognitive science, *concepts* are semantically referring to their extensions, usually called *categories* in this context [74]. Both in mathematical logic and cognitive science, extensions need not be confined to physical objects; the modeler may also define extensions in terms of mathematical structures, sensory perceptions, hypothetical worlds, ideas or facts. But at any rate, there is an ontological difference between the two ends of the semantic relationship.

This ontological gap dissolves in the case of conceptors. The natural account

of the "meaning" of a matrix conceptor C is the shape of the neural state cloud it is derived from. This shape is given by the correlation matrix R of neural states. Both C and R have the same mathematical format: positive semi-definite matrices of identical dimension. Figure 9 visualizes the difference between classical extensional semantics of logics and the system-internal conceptor semantics. The symbol $\models$ is the standard mathematical notation for the semantical meaning relationship.

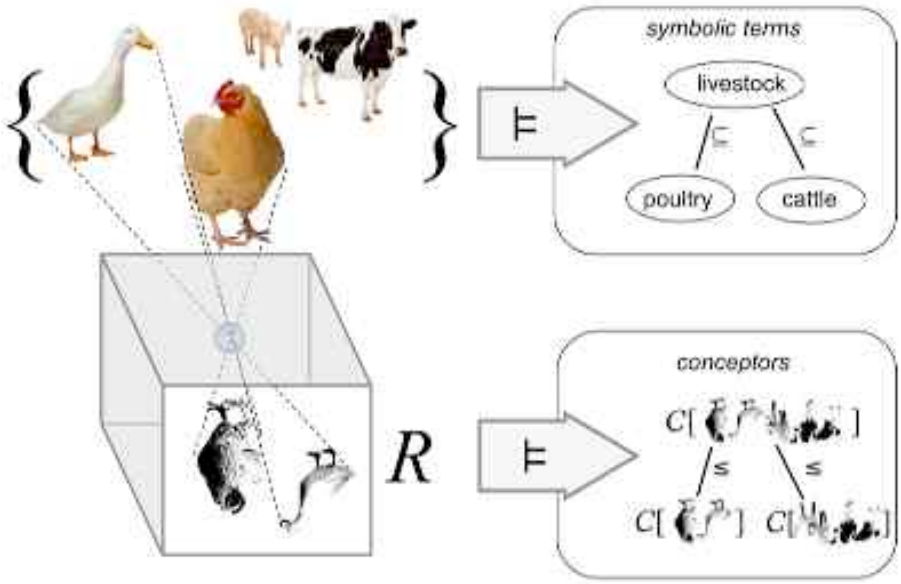

Figure 9: Contrasting the extensional semantics of classical knowledge representation formalisms (upper half of graphics) with conceptor semantics (lower half).

I have cast these intuitions into a formal specification of an *intrinsic conceptor logic* (ICL), where the semantic relationship outlined above is formalized within the framework of *institutions* [37]. This framework has been developed in mathematics and computer science to provide a unified view on the multitude of existing "logics". By formalizing ICL as an institution, conceptor logic can be rigorously compared to other existing logics. I highlight two findings. First, an ICL cast as an institution is a dynamcial system in its own right: the symbols used in this logic evolve over time. This is very much different from traditional views on logic, where symbols are static tokens. Second, it turns out that ICL is a logic which is *decidable*. Stated in intuitive terms, in a decidable logic it can be calculated whether a "concept" ψ subsumes a concept φ (as in "a cow is an animal"). Deciding concept subsumption is a core task in AI systems and human cognition. In most logic-based AI systems, deciding concept subsumption can become computationally expensive or even impossible. In ICL it boils down to determining whether all components of a certain conception weight vector c_i are smaller or equal to the corresponding components c'_i of another such vector, which can be

done in a single processing step. This may help explaining why humans can make classification judgements almost instantaneously.

Discussion. The human brain is a neurodynamical system which evidently supports logico-rational reasoning [49]. Since long this has challenged scientists to find computational models which connect neural dynamics with logic. Very different solutions have been suggested. At the dawn of computational neuroscience, McCulloch and Pitts have already interpreted networks of binary-state neurons as carrying out Boolean operations [73]. Logical inferences of various kinds have been realized in localist connectionist networks where neurons are labelled by concept names [82, 96]. In neurofuzzy modeling, feedforward neural networks are trained to carry out operations of fuzzy logic on their inputs [12]. In a field known as neuro-symbolic computation, deduction rules of certain formal logic systems are coded or trained into neural networks [8, 65, 10]. The combinatorial/compositional structure of symbolic knowledge has been captured by dedicated neural circuits to enable tree-structured representations [83] or variable-binding functionality [104].

All of these approaches require *interface* mechanisms. These interface mechanisms are non-neural and code symbolic knowledge representations into the numerical activation values of neurons and/or the topological structure of networks. One could say, previous approaches *code* logic *into* specialized neural networks, while conceptors *instantiate* the logic *of* generic recurrent neural networks. This novel, simple, versatile, computationally efficient, neurally not infeasible, bi-directional connection between logic and neural dynamics opens new perspectives for computational neuroscience and machine learning.

2 Introduction

In this section I expand on the brief characterization of the scientific context given in Section 1, and introduce mathematical notation.

2.1 Motivation

Intelligent behavior is desired for robots, demonstrated by humans, and studied in a wide array of scientific disciplines. This research unfolds in two directions. In "top-down" oriented research, one starts from the "higher" levels of cognitive performance, like rational reasoning, conceptual knowledge representation, planning and decision-making, command of language. These phenomena are described in symbolic formalisms developed in mathematical logic, artificial intelligence (AI), computer science and linguistics. In the "bottom-up" direction, one departs from "low-level" sensor data processing and motor control, using the analytical tools offered by dynamical systems theory, signal processing and control theory, statistics and information theory. For brevity I will refer to these two directions as the *conceptual-symbolic* and the *data-dynamical* sets of phenomena, and levels of description. The two interact bi-directionally. Higher-level symbolic concepts arise from low-level sensorimotor data streams in short-term pattern recognition and long-term learning processes. Conversely, low-level processing is modulated, filtered and steered by processes of attention, expectations, and goal-setting in a top-down fashion.

Several schools of thought (and strands of dispute) have evolved in a decades-long quest for a unification of the conceptual-symbolic and the data-dynamical approaches to intelligent behavior. The nature of symbols in cognitive processes has been cast as a philosophical issue [95, 29, 44]. In localist connectionistic models, symbolically labelled abstract processing units interact by nonlinear *spreading activation* dynamics [22, 96]. A basic tenet of behavior-based AI is that higher cognitive functions *emerge* from low-level sensori-motor processing loops which couple a behaving agent into its environment [11, 81]. Within cognitive science, a number of cognitive pheneomena have been described in terms of *self-organization* in nonlinear dynamical systems [94, 98, 105]. A pervasive idea in theoretical neuroscience is to interpret *attractors* in nonlinear neural dynamics as the carriers of conceptual-symbolic representations. This idea can be traced back at least to the notion of cell assemblies formulated by Hebb [45], reached a first culmination in the formal analysis of associative memories [80, 48, 4], and has since then diversified into a range of increasingly complex models of interacting (partial) neural attractors [109, 103, 87, 101]. Another pervasive idea in theoretical neuroscience and machine learning is to consider *hierarchical* neural architectures, which are driven by external data at the bottom layer and transform this raw signal into increasingly abstract feature representations, arriving at conceptual representations at the top layer of the hierarchy. Such hierarchical architectures mark the state of the art in pattern recognition technology [66, 38]. Many of these systems

process their input data in a uni-directional, bottom-to-top fashion. Two notable exceptions are systems where each processing layer is designed according to statistical principles from *Bayes' rule* [33, 47, 16], and models based on the iterative linear maps of *map seeking circuits* [35, 111], both of which enable top-down guidance of recognition by expectation generation. More generally, leading actors in theoretical neuroscience have characterized large parts of their field as an effort to understand how cognitive phenomena arise from neural dynamics [2, 36]. Finally, I point out two singular scientific efforts to design comprehensive cognitive brain models, the ACT-R architectures developed by Anderson et al. [5] and the Spaun model of Eliasmith et al. [26]. Both systems can simulate a broad selection of cognitive behaviors. They integrate numerous subsystems and processing mechanisms, where ACT-R is inspired by a top-down modeling approach, starting from cognitive operations, and Spaun from a bottom-up strategy, starting from neurodynamical processing principles.

Despite this extensive research, the problem of integrating the conceptual-symbolic with the data-dynamical aspects of cognitive behavior cannot be considered solved. Quite to the contrary, two of the largest current research initiatives worldwide, the Human Brain Project [1] and the NIH BRAIN initiative [51], are ultimately driven by this problem. There are many reasons why this question is hard, ranging from experimental challenges of gathering relevant brain data to fundamental oppositions of philosophical paradigms. An obstinate stumbling block is the different mathematical nature of the fundamental formalisms which appear most natural for describing conceptual-symbolic versus data-dynamical phenomena: symbolic logic versus nonlinear dynamics. Logic-oriented formalisms can easily capture all that is combinatorially constructive and hierarchically organized in cognition: building new concepts by logical definitions, describing nested plans for action, organizing conceptual knowledge in large and easily extensible abstraction hierarchies. But logic is inherently non-temporal, and in order to capture cognitive *processes*, additional, heuristic "scheduling" routines have to be introduced which control the order in which logical rules are executed. This is how ACT-R architectures cope with the integration problem. Conversely, dynamical systems formalisms are predestined for modeling all that is continuously changing in the sensori-motor interface layers of a cognitive system, driven by sensor data streams. But when dynamical processing modules have to be combined into compounds that can solve complex tasks, again additional design elements have to be inserted, usually by manually coupling dynamical modules in ways that are informed by biological or engineering insight on the side of the researcher. This is how the Spaun model has been designed to realize its repertoire of cognitive functions. Two important modeling approaches venture to escape from the logic-dynamics integration problem by taking resort to an altogether different mathematical framework which can accomodate both sensor data processing and concept-level representations: the framework of Bayesian statistics and the framework of iterated linear maps mentioned above. Both approaches lead to a unified

formal description across processing and representation levels, but at the price of a double weakness in accounting for the embodiment of an agent in a dynamical environment, and for the combinatorial aspects of cognitive operations. It appears that current mathematical methods can instantiate only one of the three: continuous dynamics, combinatorial productivity, or a unified level-crossing description format.

The conceptor mechanisms introduced in this report bi-directionally connect the data-dynamical workings of a recurrent neural network (RNN) with a conceptual-symbolic representation of different functional modes of the RNN. Mathematically, conceptors are linear operators which characterize classes of signals that are being processed in the RNN. Conceptors can be represented as matrices (convenient in machine learning applications) or as neural subnetworks (appropriate from a computational neuroscience viewpoint). In a bottom-up way, starting from an operating RNN, conceptors can be learnt and stored, or quickly generated on-the-fly, by what may be considered the simplest of all adaptation rules: learning a regularized identity map. Conceptors can be combined by elementary logical operations (AND, OR, NOT), and can be ordered by a natural abstraction relationship. These logical operations and relations are defined via a formal semantics. Thus, an RNN engaged in a variety of tasks leads to a learnable representation of these operations in a logic formalism which can be neurally implemented. Conversely, in a top-down direction, conceptors can be inserted into the RNN's feedback loop, where they robustly steer the RNN's processing mode. Due to their linear algebra nature, conceptors can be continuously morphed and "sharpened" or "defocussed", which extends the discrete operations that are customary in logics into the domain of continuous "mental" transformations. I highlight the versatility of conceptors in a series of demonstrations: generating and morphing many different dynamical patterns with a single RNN; managing and monitoring the storing of patterns in a memory RNN; learning a class of dynamical patterns from presentations of a small number of examples (with extrapolation far beyond the training examples); classification of temporal patterns; de-noising of temporal patterns; and content-addressable memory systems. The logical conceptor operations enable an incremental extension of a trained system by incorporating new patterns without interfering with already learnt ones. Conceptors also suggest a novel answer to a perennial problem of attractor-based models of concept representations, namely the question of how a cognitive trajectory can leave an attractor (which is at odds with the very nature of an attractor). Finally, I outline a version of conceptors which is biologically plausible in the modest sense that only local computations and no information copying are needed.

2.2 Mathematical Preliminaries

I assume that the reader is familiar with properties of positive semidefinite matrices, the singular value decomposition, and (in some of the analysis of adaptation

dynamics) the usage of the Jacobian of a dynamical system for analysing stability properties.

$[a, b], (a, b), (a, b], [a, b)$ denote the closed (open, half-open) interval between real numbers a and b.

A' or x' denotes the transpose of a matrix A or vector x. I is the identity matrix (the size will be clear from the context or be expressed as $I_{n\times n}$). The ith unit vector is denoted by e_i (dimension will be clear in context). The trace of a square matrix A is denoted by $\mathrm{tr}\, A$. The singular value decomposition of a matrix A is written as $USV' = A$, where U, V are orthonormal and S is the diagonal matrix containing the singular values of A, assumed to be in descending order unless stated otherwise. $A^\dagger$ is the pseudoinverse of A. All matrices and vectors will be real and this will not be explicitly mentioned.

I use the Matlab notation to address parts of vectors and matrices, for instance $M(:, 3)$ is the third column of a matrix M and $M(2 : 4, :)$ picks from M the submatrix consisting of rows 2 to 4. Furthermore, again like in Matlab, I use the operator diag in a "toggling" mode: $\mathrm{diag}\, A$ returns the diagonal vector of a square matrix A, and $\mathrm{diag}\, d$ constructs a diagonal matrix from a vector d of diagonal elements. Another Matlab notation that will be used is "$.*$" for the element-wise multiplication of vectors and matrices of the same size, and "$.\hat{}$" for element-wise exponentation of vectors and matrices.

$\mathcal{R}(A)$ and $\mathcal{N}(A)$ denote the range and null space of a matrix A. For linear subspaces $\mathcal{S}, \mathcal{T}$ of $\mathbb{R}^n$, $\mathcal{S}^\perp$ is the orthogonal complement space of $\mathcal{S}$ and $\mathcal{S} + \mathcal{T}$ is the direct sum $\{x + y \mid x \in \mathcal{S}, y \in \mathcal{T}\}$ of $\mathcal{S}$ and $\mathcal{T}$. $\mathbf{P}_\mathcal{S}$ is the $n \times n$ dimensional projection matrix on a linear subspace $\mathcal{S}$ of $\mathbb{R}^n$. For a k-dimensional linear subspace $\mathcal{S}$ of $\mathbb{R}^n$, $\mathbf{B}_\mathcal{S}$ denotes any $n \times k$ dimensional matrix whose columns form an orthonormal basis of $\mathcal{S}$. Such matrices $\mathbf{B}_\mathcal{S}$ will occur only in contexts where the choice of basis can be arbitrary. It holds that $\mathbf{P}_\mathcal{S} = \mathbf{B}_\mathcal{S}(\mathbf{B}_\mathcal{S})'$.

$E[x(n)]$ denotes the expectation (temporal average) of a stationary signal $x(n)$ (assuming it is well-defined, for instance, coming from an ergodic source).

For a matrix M, $\|M\|_{\mathrm{fro}}$ is the Frobenius norm of M. For real M, it is the square root of the summed squared elements of M. If M is positive semidefinite with SVD $M = USU'$, $\|M\|_{\mathrm{fro}}$ is the same as the 2-norm of the diagonal vector of S, i.e. $\|M\|_{\mathrm{fro}} = ((\mathrm{diag}S)'\,(\mathrm{diag}S))^{1/2}$. Since in this report I will exclusively use the Frobenius norm for matrices, I sometimes omit the subscript and write $\|M\|$ for simplicity.

In a number of simulation experiments, a network-generated signal $y(n)$ will be matched against a target pattern $p(n)$. The accuracy of the match will be quantified by the normalized root mean square error (NRMSE), $\sqrt{[(y(n) - p(n))^2]/[(p(n)^2]}$, where $[\cdot]$ is the mean operator over data points n.

The symbol N is reserved for the size of a reservoir (= number of neurons) throughout.

3 Theory and Demonstrations

This is the main section of this report. Here I develop in detail the concepts, mathematical analysis, and algorithms, and I illustrate various aspects in computer simulations.

Figure 10 gives a navigation guide through the dependency tree of the components of this section.

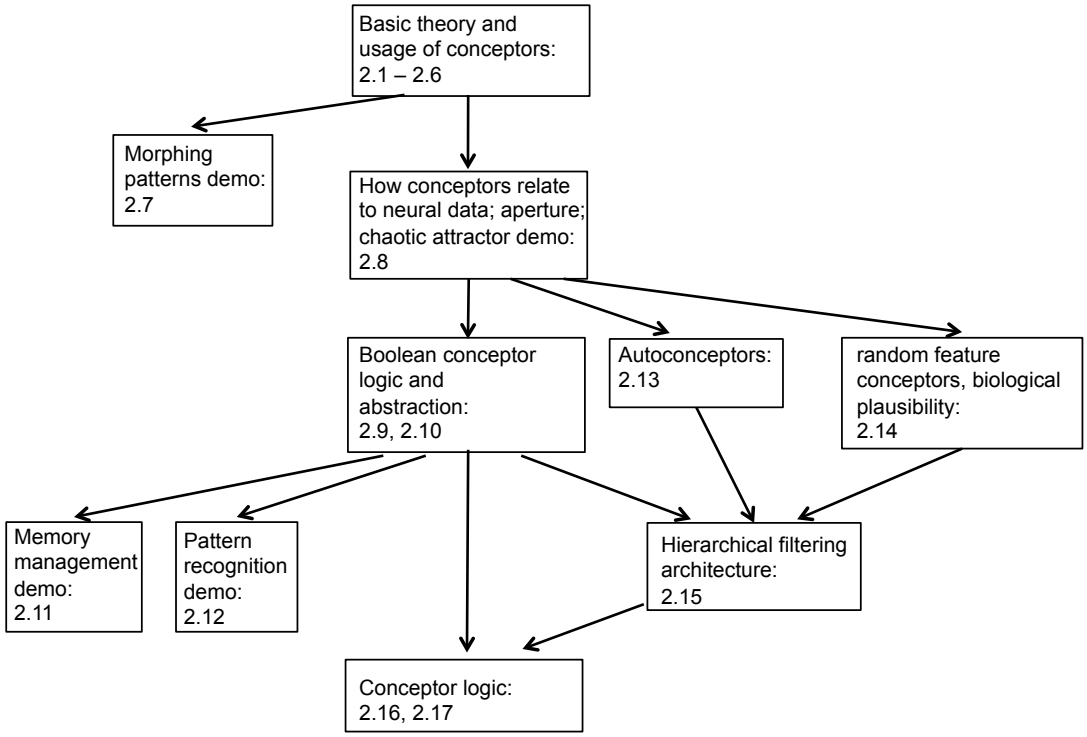

Figure 10: Dependency tree of subsections in Section 3.

The program code (Matlab) for all simulations can be retrieved from
http://minds.jacobs-university.de/sites/default/files/uploads/...
...SW/ConceptorsTechrepMatlab.zip.

3.1 Networks and Signals

Throughout this report, I will be using discrete-time recurrent neural networks made of simple tanh neurons, which will be driven by an input time series $p(n)$. In the case of 1-dimensional input, these networks consist of (i) a "reservoir" of N recurrently connected neurons whose activations form a state vector $x = (x_1, \ldots, x_N)'$, (ii) one external input neuron that serves to drive the reservoir with training or cueing signals $p(n)$ and (iii) another external neuron which serves to read out a scalar target signal $y(n)$ from the reservoir (Fig. 11). The system

operates in discrete timesteps $n = 0, 1, 2, \ldots$ according to the update equations

$$x(n+1) = \tanh(W\,x(n) + W^{\text{in}}\,p(n+1) + b) \quad (1)$$
$$y(n) = W^{\text{out}}\,x(n), \quad (2)$$

where W is the $N \times N$ matrix of reservoir-internal connection weights, W^{in} is the $N \times 1$ sized vector of input connection weights, W^{out} is the $1 \times N$ vector of readout weights, and b is a bias. The tanh is a sigmoidal function that is applied to the network state x component-wise. Due to the tanh, the *reservoir state space* or simply *state space* is $(-1, 1)^N$.

The input weights and the bias are fixed at random values and are not subject to modification through training. The output weights W^{out} are learnt. The reservoir weights W are learnt in some of the case studies below, in others they remain fixed at their initial random values. If they are learnt, they are adapted from a random initialization denoted by W^*. Figure 11 **A** illustrates the basic setup.

I will call the driving signals $p(n)$ *patterns*. In most parts of this report, patterns will be periodic. Periodicity comes in two variants. First, *integer-periodic* patterns have the property that $p(n) = p(n+k)$ for some positive integer k. Second, *irrational-periodic* patterns are discretely sampled from continuous-time periodic signals, where the sampling interval and the period length of the continuous-time signal have an irrational ratio. An example is $p(n) = \sin(2\,\pi\,n/(10\,\sqrt{2}))$. These two sorts of drivers will eventually lead to different kinds of attractors trained into reservoirs: integer-periodic signals with period length P yield attractors consisting of P points in reservoir state space, while irrational-periodic signals give rise to attracting sets which can be topologically characterized as one-dimensional cycles that are homeomorphic to the unit cycle in $\mathbb{R}^2$.

3.2 Driving a Reservoir with Different Patterns

A basic theme in this report is to develop methods by which a collection of different patterns can be loaded in, and retrieved from, a single reservoir. The key for these methods is an elementary dynamical phenomenon: if a reservoir is driven by a pattern, the entrained network states are confined to a linear subspace of network state space which is characteristic of the pattern. In this subsection I illuminate this phenomenon by a concrete example. This example will be re-used and extended on several occasions throughout this report.

I use four patterns. The first two are irrational periodic and the last two are integer-periodic: (1) a sinewave of period ≈ 8.83 sampled at integer times (pattern $p^1(n)$) (2) a sinewave $p^2(n)$ of period ≈ 9.83 (period of $p^1(n)$ plus 1), (3) a random 5-periodic pattern $p^3(n)$ and (4) a slight variation $p^4(n)$ thereof (Fig. 12 left column).

A reservoir with $N = 100$ neurons is randomly created. At creation time the input weights W^{in} and the bias b are fixed at random values; these will never be modified thereafter. The reservoir weights are initialized to random values W^*;

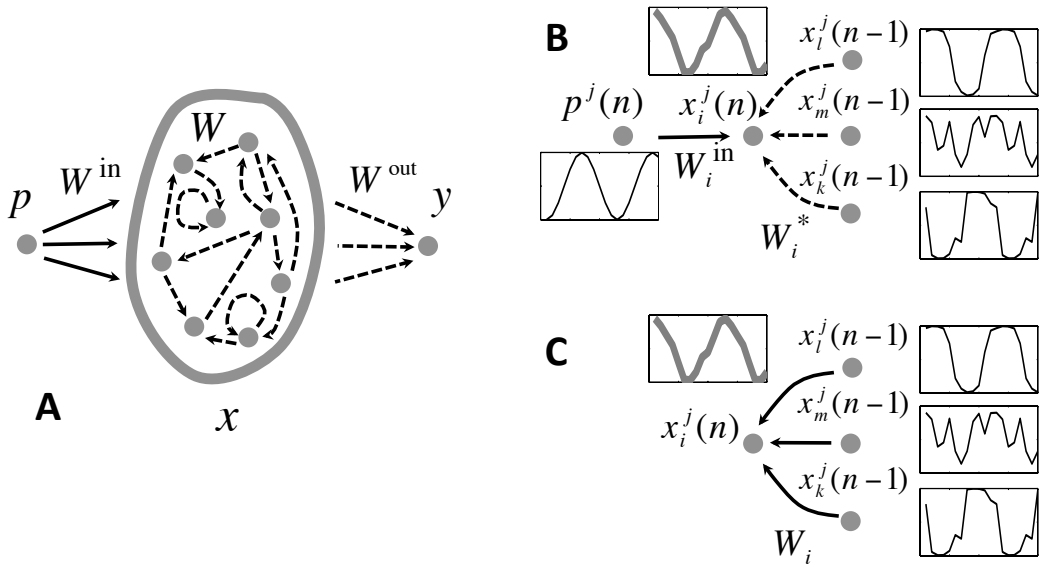

Figure 11: **A.** Basic system setup. Through input connections W^{in}, an input neuron feeds a driving signal p to a "reservoir" of $N = 100$ neurons which are recurrently connected to each other through connections W. From the N-dimensional neuronal activation state x, an output signal y is read out by connections W^{out}. All broken connections are trainable. **B.** During the initial driving of the reservoir with driver p^j, using initial random weights W^*, neuron x_i produces its signal (thick gray line) based on external driving input p and feeds from other neurons x from within the reservoir (three shown). **C.** After training new reservoir weights W, the same neuron should produce the same signal based only on the feeds from other reservoir neurons.

in this first demonstration they will not be subsequently modified either. The readout weights are initially undefined (details in Section 4.1).

In four successive and independent runs, the network is driven by feeding the respective pattern $p^j(n)$ as input ($j = 1, \ldots, 4$), using the update rule

$$x^j(n+1) = \tanh(W^* x^j(n) + W^{\text{in}} p^j(n+1) + b).$$

After an initial washout time, the reservoir dynamics becomes entrained to the driver and the reservoir state $x^j(n)$ exhibits an involved nonlinear response to the driver p^j. After this washout, the reservoir run is continued for $L = 1000$ steps, and the obtained states $x^j(n)$ are collected into $N \times L = 100 \times 1000$ sized state collection matrices X^j for subsequent use.

The second column in Fig. 12 shows traces of three randomly chosen reservoir neurons in the four driving conditions. It is apparent that the reservoir has become entrained to the driving input. Mathematically, this entrainment is captured by the concept of the *echo state property*: any random initial state of a reservoir is "forgotten", such that after a washout period the current network state is a

function of the driver. The echo state property is a fundamental condition for RNNs to be useful in learning tasks [54, 13, 46, 110, 99, 71]. It can be ensured by an appropriate scaling of the reservoir weight matrix. All networks employed in this report possess the echo state property.

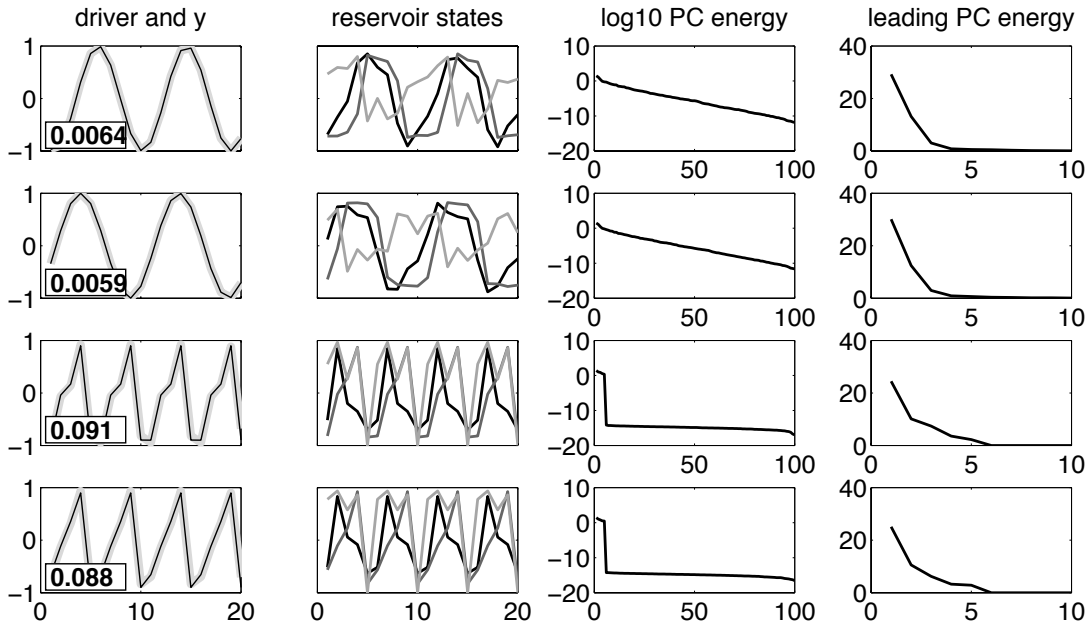

Figure 12: The subspace phenomenon. Each row of panels documents situation when the reservoir is driven by a particular input pattern. "Driver and y": the driving pattern (thin black line) and the signals retrieved with conceptors (broad light gray line). Number inset is the NRMSE between original driver and retrieved signal. "Reservoir states": activations of three randomly picked reservoir neurons. "Log10 PC energy": $\log_{10}$ of reservoir signal energies in the principal component directions. "Leading PC energy": close-up on first ten signal energies in linear scale. Notice that the first two panels in each row show discrete-time signals; points are connected by lines only for better visual appearance.

A principal component analysis (PCA) of the 100 reservoir signals reveals that the driven reservoir signals are concentrated on a few principal directions. Concretely, for each of the four driving conditions, the reservoir state correlation matrix was estimated by $R^j = X^j\,(X^j)'/L$, and its SVD $U^j\Sigma^j(U^j)' = R^j$ was computed, where the columns of U^j are orthonormal eigenvectors of R^j (the principal component (PC) vectors), and the diagonal of Σ^j contains the singular values of R^j, i.e. the energies (mean squared amplitudes) of the principal signal components. Figure 12 (third and last column) shows a plot of these principal component energies. The energy spectra induced by the two irrational-period sines look markedly different from the spectra obtained from the two 5-periodic signals. The latter lead to nonzero energies in exactly 5 principal directions because the driven reservoir dynamics periodically visits 5 states (the small but nonzero values in the $\log_{10}$

plots in Figure 12 are artefacts earned from rounding errors in the SVD computation). In contrast, the irrational-periodic drivers lead to reservoir states which linearly span all of $\mathbb{R}^N$ (Figure 12, upper two $\log_{10}$ plots). All four drivers however share a relevant characteristic (Figure 12, right column): the total reservoir energy is concentrated in a quite small number of leading principal directions.

When one inspects the excited reservoir dynamics in these four driving conditions, there is little surprise that the neuronal activation traces look similar to each other for the first two and in the second two cases (Figure 12, second column). This "similarity" can be quantified in a number of ways. Noting that the geometry of the "reservoir excitation space" in driving condition j is characterized by a hyperellipsoid with main axes U^j and axis lengths $\operatorname{diag}\Sigma^j$, a natural way to define a similarity between two such ellipsoids i, j is to put

$$\mathrm{sim}^R_{i,j} = \frac{(\Sigma^i)^{1/2}\,(U^i)'U^j(\Sigma^j)^{1/2}}{\|\mathrm{diag}\Sigma^i\|\,\|\mathrm{diag}\Sigma^j\|}. \tag{3}$$

The measure $\mathrm{sim}^R_{i,j}$ ranges in $[0, 1]$. It is 0 if and only if the reservoir signals x^i, x^j populate orthogonal linear subspaces, and it is 1 if and only if $R^i = a\,R^j$ for some scaling factor a. The measure $\mathrm{sim}^R_{i,j}$ can be understood as a generalized squared cosine between R^i and R^j. Figure 13 **A** shows the similarity matrix $(\mathrm{sim}^R_{i,j})_{i,j}$ obtained from (3). The similarity values contained in this matrix appear somewhat counter-intuitive, inasmuch as the reservoir responses to the sinewave patterns come out as having similarities of about 0.6 with the 5-periodic driven reservoir signals; this does not agree with the strong visual dissimilarity apparent in the state plots in Figure 12. In Section 3.5 I will introduce another similarity measure which agrees better with intuitive judgement.

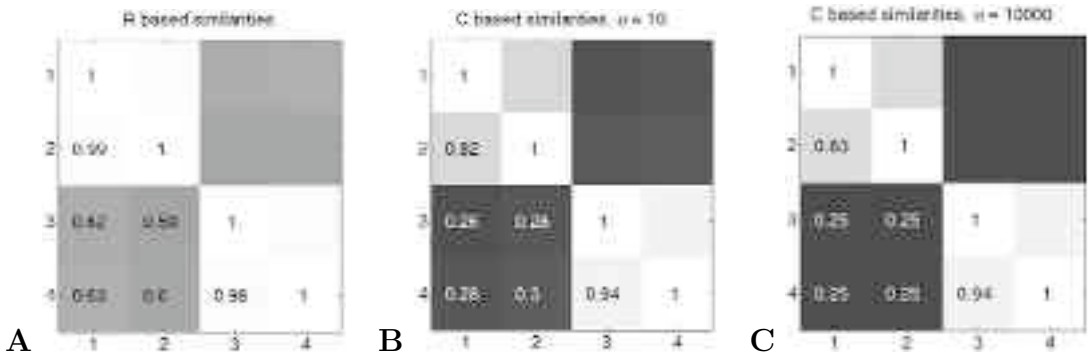

Figure 13: Matrix plots of pairwise similarity between the subspaces excited in the four driving conditions. Grayscale coding: 0 = black, 1 = white. **A:** similarity $\mathrm{sim}^R_{i,j}$ based on the data correlation matrices R^i. **B,C:** similarities based on conceptors $C(R^i, \alpha)$ for two different values of aperture α. For explanation see text.

3.3 Storing Patterns in a Reservoir, and Training the Readout

One of the objectives of this report is a method for storing several driving patterns in a single reservoir, such that these stored patterns can later be retrieved and otherwise be controlled or manipulated. In this subsection I explain how the initial "raw" reservoir weights W^* are adapted in order to "store" or "memorize" the drivers, leading to a new reservoir weight matrix W. I continue with the four-pattern-example used above.

The guiding idea is to enable the reservoir to re-generate the driven responses $x^j(n)$ *in the absence of the driving input.* Consider any neuron x_i (Fig. 11**B**). During the driven runs $j = 1, \ldots, 4$, it has been updated per

$$x_i^j(n+1) = \tanh(W_i^* \, x^j(n) + W_i^{\text{in}} \, p^j(n+1) + b_i),$$

where W_i^* is the i-th row in W^*, W_i^{in} is the i-th element of W^{in}, and b_i is the ith bias component. The objective for determining new reservoir weights W is that the trained reservoir should be able to oscillate in the same four ways as in the external driving conditions, but without the driving input. That is, the new weights W_i leading to neuron i should approximate

$$\tanh(W_i^* \, x^j(n) + W_i^{\text{in}} \, p^j(n+1) + b_i) \approx \tanh(W_i \, x^j(n) + b_i)$$

as accurately as possible, for $j = 1, \ldots, 4$. Concretely, we optimize a mean square error criterion and compute

$$W_i = \text{argmin}_{\tilde{W}_i} \sum_{j=1,\ldots,K} \sum_{n=1,\ldots,L} (W_i^* \, x^j(n) + W_i^{\text{in}} \, p^j(n+1) - \tilde{W}_i \, x^j(n))^2, \quad (4)$$

where K is the number of patterns to be stored (in this example $K = 4$). This is a linear regression task, for which a number of standard algorithms are available. I employ ridge regression (details in Section 4.1).

The readout neuron y serves as passive observer of the reservoir dynamics. The objective to determine its connection weights W^{out} is simply to replicate the driving input, that is, W^{out} is computed (again by ridge regression) such that it minimizes the squared error $(p^j(n) - W^{\text{out}} x^j(n))^2$, averaged over time and the four driving conditions.

I will refer to this preparatory training as *storing* patterns p^j in a reservoir, and call a reservoir *loaded* after patterns have been stored.

3.4 Conceptors: Introduction and Basic Usage in Retrieval

How can these stored patterns be individually *retrieved* again? After all, the storing process has superimposed impressions of all patterns on all of the recomputed connection weights W of the network – very much like the pixel-wise

addition of different images would yield a mixture image in which the individual original images are hard to discern. One would need some sort of filter which can disentangle again the superimposed components in the connection weights. In this section I explain how such filters can be obtained.

The guiding idea is that for retrieving pattern j from a loaded reservoir, the reservoir dynamics should be restricted to the linear subspace which is characteristic for that pattern. For didactic reasons I start with a simplifying assumption (to be dropped later). Assume that there exists a (low-dimensional) linear subspace $\mathcal{S}^j \subset \mathbb{R}^N$ such that all state vectors contained in the driven state collection X^j lie in $\mathcal{S}^j$. In our example, this is actually the case for the two 5-periodic patterns. Let $\mathbf{P}_{\mathcal{S}^j}$ be the projector matrix which projects $\mathbb{R}^N$ on $\mathcal{S}^j$. We may then hope that if we run the loaded reservoir autonomously (no input), constraining its states to $\mathcal{S}^j$ using the update rule

$$x(n+1) = \mathbf{P}_{\mathcal{S}^j} \tanh(W\, x(n) + b), \tag{5}$$

it will oscillate in a way that is closely related to the way how it oscillated when it was originally driven by p^j.

However, it is not typically the case that the states obtained in the original driving conditions are confined to a proper linear subspace of the reservoir state space. Consider the sine driver p^1 in our example. The linear span of the reservoir response state is all of $\mathbb{R}^N$ (compare the $\log_{10}$ PC energy plots in Figure 12). The associated projector would be the identity, which would not help to single out an individual pattern in retrieval. But actually we are not interested in those principal directions of reservoir state space whose excitation energies are negligibly small (inspect again the quick drop of these energies in the third column, top panel in Figure 12 – it is roughly exponential over most of the spectrum, except for an even faster decrease for the very first few singular values). Still considering the sinewave pattern p^1: instead of $\mathbf{P}_{\mathbb{R}^N}$ we would want a projector that projects on the subspace spanned by a "small" number of leading principal components of the "excitation ellipsoid" described by the sine-driver-induced correlation matrix R^1. What qualifies as a "small" number is, however, essentially arbitrary. So we want a method to shape projector-like matrices from reservoir state correlation matrices R^j in a way that we can adjust, with a control parameter, how many of the leading principal components should become registered in the projector-like matrix.

At this point I give names to the projector-like matrices and the adjustment parameter. I call the latter the *aperture* parameter, denoted by α. The projector-like matrices will be called *conceptors* and generally be denoted by the symbol C. Since conceptors are derived from the ellipsoid characterized by a reservoir state corrlation matrix R^j, and parametrized by the aperture parameter, I also sometimes write $C(R^j, \alpha)$ to make this dependency transparent.

There is a natural and convenient solution to meet all the intuitive objectives for conceptors that I discussed up to this point. Consider a reservoir driven by

a pattern $p^j(n)$, leading to driven states $x^j(n)$ collected (as columns) in a state collection matrix X^j, which in turn yields a reservoir state correlation matrix $R^j = X^j(X^j)'/L$. We define a conceptor $C(R^j, \alpha)$ with the aid of a cost function $\mathcal{L}(C \mid R^j, \alpha)$, whose minimization yields $C(R^j, \alpha)$. The cost function has two components. The first component reflects the objective that C should behave as a projector matrix for the states that occur in the pattern-driven run of the reservoir. This component is $E_n[\|x^j(n) - Cx^j(n)\|^2]$, the time-averaged deviation of projections Cx^j from the state vectors x^j. The second component of $\mathcal{L}$ adjusts how many of the leading directions of R^j should become effective for the projection. This component is $\alpha^{-2}\|C\|^2_{\text{fro}}$. This leads to the following definition.

Definition 1 *Let $R = E[xx']$ be an $N \times N$ correlation matrix and $\alpha \in (0, \infty)$. The* conceptor matrix $C = C(R, \alpha)$ associated with R and α *is*

$$C(R, \alpha) = \text{argmin}_C \, E[\|x - Cx\|^2] + \alpha^{-2} \, \|C\|^2_{\text{fro}}. \tag{6}$$

The minimization criterion (6) uniquely specifies $C(R, \alpha)$. The conceptor matrix can be effectively computed from R and α. This is spelled out in the following proposition, which also lists elementary algebraic properties of conceptor matrices:

Proposition 1 *Let $R = E[x\,x']$ be a correlation matrix and $\alpha \in (0, \infty)$. Then,*

1. *$C(R, \alpha)$ can be directly computed from R and α by*

$$C(R, \alpha) = R\,(R + \alpha^{-2}\, I)^{-1} = (R + \alpha^{-2}\, I)^{-1}\, R, \tag{7}$$

2. *if $R = U\Sigma U'$ is the SVD of R, then the SVD of $C(R, \alpha)$ can be written as $C = USU'$, i.e. C has the same principal component vector orientation as R,*

3. *the singular values s_i of C relate to the singular values σ_i of R by $s_i = \sigma_i/(\sigma_i + \alpha^{-2})$,*

4. *the singular values of C range in $[0, 1)$,*

5. *R can be recovered from C and α by*

$$R = \alpha^{-2}\,(I - C)^{-1}\, C = \alpha^{-2}\, C\,(I - C)^{-1}. \tag{8}$$

The proof is given in Section 5.1. Notice that all inverses appearing in this proposition are well-defined because $\alpha > 0$ is assumed, which implies that all singular values of $C(R, \alpha)$ are properly smaller than 1. I will later generalize conceptors to include the limiting cases $\alpha = 0$ and $\alpha = \infty$ (Section 3.8.1).

In practice, the correlation matrix $R = E[xx']$ is estimated from a finite sample X, which leads to the approximation $\hat{R} = XX'/L$, where $X = (x(1), \ldots, x(L))$ is a matrix containing reservoir states $x(n)$ collected during a learning run.

Figure 14 shows the singular value spectra of $C(R, \alpha)$ for various values of α, for our example cases of $R = R^1$ (irrational-period sine driver) and $R = R^3$ (5-periodic driver). We find that the nonlinearity inherent in (7) makes the conceptor matrices come out "almost" as projector matrices: the singular values of C are mostly close to 1 or close to 0. In the case of the 5-periodic driver, where the excited network states populate a 5-dimensional subspace of $\mathbb{R}^N$, increasing α lets $C(R, \alpha)$ converge to a projector onto that subspace.

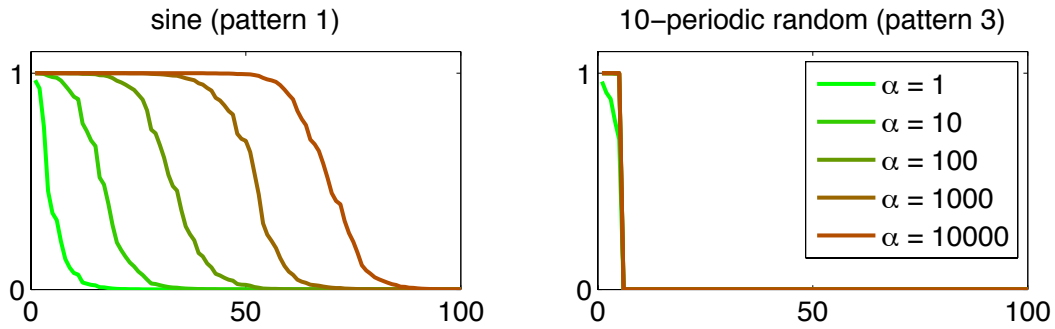

Figure 14: How the singular values of a conceptor depend on α. Singular value spectra are shown for the first sinewave pattern and the first 5-periodic random pattern. For explanation see text.

If one has a conceptor matrix $C^j = C(R^j, \alpha)$ derived from a pattern p^j through the reservoir state correlation matrix R^j associated with that pattern, the conceptor matrix can be used in an autonomous run (no external input) using the update rule

$$x(n+1) = C^j \tanh(W\, x(n) + b), \tag{9}$$

where the weight matrix W has been shaped by storing patterns among which there was p^j. Returning to our example, four conceptors $C^1, \ldots, C^4$ were computed with $\alpha = 10$ and the loaded reservoir was run under rule (9) from a random initial state $x(0)$. After a short washout period, the network settled on stable periodic dynamics which were closely related to the original driving patterns. The network dynamics was observed through the previously trained output neuron. The left column in Figure 12 shows the autonomous network output as a light bold gray line underneath the original driver. To measure the achieved accuracy, the autonomous output signal was phase-aligned with the driver (details in Section 4.1) and then the NRMSE was computed (insets in Figure panels). The NRMSEs indicate that the conceptor-constrained autonomous runs could successfully separate from each other even the closely related pattern pairs p^1 versus p^2 and p^3 versus p^4.

A note on terminology. Equation (9) shows a main usage of conceptor matrices: they are inserted into the reservoir state feedback loop and cancel (respectively, dampen) those reservoir state components which correspond to directions in state space associated with zero (or small, respectively) singular values in the

conceptor matrix. In most of this report, such a direction-selective damping in the reservoir feedback loop will be effected by way of inserting matrices C like in Equation (9). However, inserting a matrix is not the only way by which such a direction-selective damping can be achieved. In Section 3.14, which deals with biological plausibility issues, I will propose a neural circuit which achieves a similar functionality of direction-specific damping of reservoir state components by other means and with slightly differing mathematical properties. I understand the concept of a "conceptor" as comprising any mechanism which effects a pattern-specific damping of reservoir signal components. Since in most parts of this report this will be achieved with conceptor matrices, as in (9), I will often refer to these C matrices as "conceptors" for simplicity. The reader should however bear in mind that the notion of a conceptor is more comprehensive than the notion of a conceptor matrix. I will not spell out a formal definition of a "conceptor", deliberately leaving this concept open to become instantiated by a variety of computational mechanisms of which only two are formally defined in this report (via conceptor matrices, and via the neural circuit given in Section 3.14).

3.5 A Similarity Measure for Excited Network Dynamics

In Figure 13 **A** a similarity matrix is presented which compares the excitation ellipsoids represented by the correlation matrices R^j by the similarity metric (3). I remarked at that time that this is not a fully satisfactory metric, because it does not agree well with intuition. We obtain a more intuitively adequate similiarity metric if conceptor matrices are used as descriptors of "subspace ellipsoid geometry" instead of the raw correlation matrices, i.e. if we employ the metric

$$\mathrm{sim}^{\alpha}_{i,j} = \frac{(S^i)^{1/2}\,(U^i)'U^j(S^j)^{1/2}}{\|\mathrm{diag}S^i\|\,\|\mathrm{diag}S^j\|}, \tag{10}$$

where US^jU' is the SVD of $C(R^j,\alpha)$. Figure 13 **B**,**C** shows the similarity matrices arising in our standard example for $\alpha = 10$ and $\alpha = 10,000$. The intuitive dissimilarity between the sinewave and the 5-periodic patterns, and the intuitive similarity between the two sines (and the two 5-periodic pattern versions, respectively) is revealed much more clearly than on the basis of $\mathrm{sim}^{R}_{i,j}$.

When interpreting similarities $\mathrm{sim}^{R}_{i,j}$ or $\mathrm{sim}^{\alpha}_{i,j}$ one should bear in mind that one is not comparing the original driving patterns but the excited reservoir responses.

3.6 Online Learning of Conceptor Matrices

The minimization criterion (6) immediately leads to a stochastic gradient online method for adapting C:

Proposition 2 *Assume that a stationary source $x(n)$ of N-dimensional reservoir states is available. Let $C(1)$ be any $N \times N$ matrix, and $\lambda > 0$ a learning rate.*

Then the stochastic gradient adaptation

$$C(n+1) = C(n) + \lambda \left((x(n) - C(n)\, x(n))\, x'(n) - \alpha^{-2}\, C(n) \right) \tag{11}$$

will lead to $\lim_{\lambda \downarrow 0} \lim_{n \to \infty} C(n) = C(E[x\, x'], \alpha)$.

The proof is straightforward if one employs generally known facts about stochastic gradient descent and the fact that $E[\|x - Cx\|^2] + \alpha^{-2}\, \|C\|^2_{\text{fro}}$ is positive definite quadratic in the N^2-dimensional space of elements of C (shown in the proof of Proposition 1), and hence provides a Lyapunov function for the gradient descent (11). The gradient of $E[\|x - Cx\|^2] + \alpha^{-2}\, \|C\|^2_{\text{fro}}$ with respect to C is

$$\frac{\partial}{\partial C}\, E[\|x - Cx\|^2] + \alpha^{-2}\, \|C\|^2_{\text{fro}} = (I - C)\, E[xx'] - \alpha^{-2}\, C, \tag{12}$$

which immediately yields (11).

The stochastic update rule (11) is very elementary. It is driven by two components, (i) an error signal $x(n) - C(n)\, x(n)$ which simply compares the current state with its C-mapped value, and (ii) a linear decay term. We will make heavy use of this adaptive mechanism in Sections 3.13.1 *ff.* This observation is also illuminating the intuitions behind the definition of conceptors. The two components strike a compromise (balanced by α) between (i) the objective that C should leave reservoir states from the target pattern unchanged, and (ii) C should have small weights. In the terminology of machine learning one could say, "a conceptor is a regularized identity map".

3.7 Morphing Patterns

Conceptor matrices offer a way to morph RNN dynamics. Suppose that a reservoir has been loaded with some patterns, among which there are p^i and p^j with corresponding conceptors C^i, C^j. Patterns that are intermediate between p^i and p^j can be obtained by running the reservoir via (9), using a linear mixture between C^i and C^j:

$$x(n+1) = \left((1-\mu) C^i + \mu C^j \right) \tanh(W\, x(n) + b). \tag{13}$$

Still using our four-pattern example, I demonstrate how this morphing works out for morphing (i) between the two sines, (ii) between the two 5-periodic patterns, (iii) between a sine and a 5-periodic pattern.

Frequency Morphing of Sines In this demonstration, the morphing was done for the two sinewave conceptors $C^1 = C(R^1, 10)$ and $C^2 = C(R^2, 10)$. The morphing parameter μ was allowed to range from -2 to $+3$ (!). The four-pattern-loaded reservoir was run from a random initial state for 500 washout steps, using (13) with $\mu = -2$. Then recording was started. First, the run was continued with the intial $\mu = -2$ for 50 steps. Then, μ was linearly ramped up from $\mu = -2$ to $\mu = 3$ during 200 steps. Finally, another 50 steps were run with the final setting $\mu = 3$.

Note that morph values $\mu = 0$ and $\mu = 1$ correspond to situations where the reservoir is constrained by the original conceptors C^1 and C^2, respectively. Values $0 \leq \mu \leq 1$ correspond to interpolation. Values $-2 \leq \mu < 0$ and $1 < \mu \leq 3$ correspond to extrapolation. The extrapolation range on either side is twice as long as the interpolation range.

In addition, for eight equidistant values μ_k in $-2 \leq \mu < 3$, the reservoir was run with a mixed conceptor $C = (1-\mu_k)C^1 + \mu_k C^2$ for 500 steps, and the obtained observation signal $y(n)$ was plotted in a delay-embedded representation, yielding "snapshots" of the reservoir dynamics at these μ values (a delay-embedding plot of a 1-dimensional signal $y(n)$ creates a 2-dimensional plot by plotting value pairs $(y(n), y(n-d))$ with a delay d chosen to yield an appealing visual appearance).

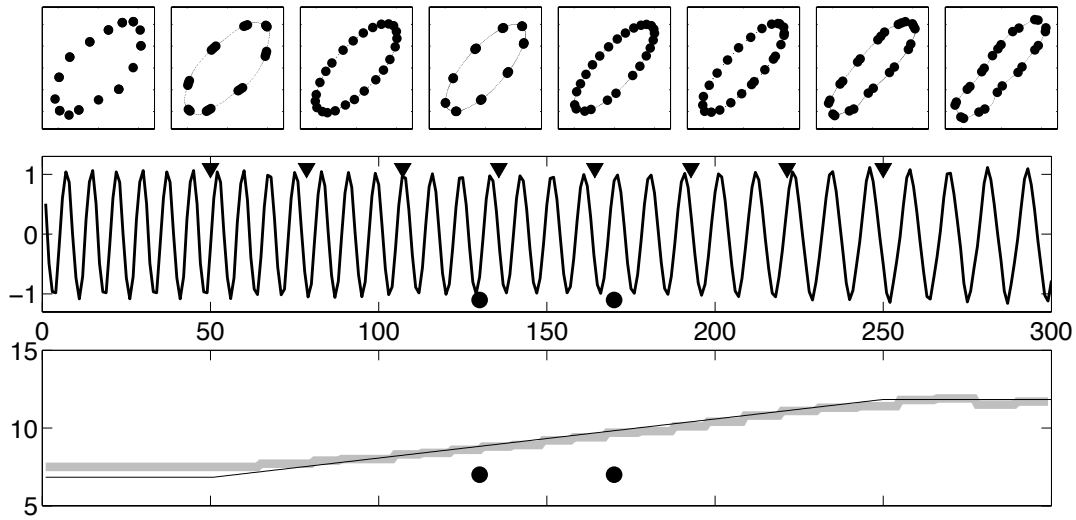

Figure 15: Morphing between (and beyond) two sines. The morphing range was $-2 \leq \mu \leq 3$. Black circular dots in the two bottom panels mark the points $\mu = 0$ and $\mu = 1$, corresponding to situations where the two original conceptors C^1, C^2 were active in unadulterated form. Top: Delay-embedding plots of network observation signal $y(n)$ (delay = 1 step). Thick points show 25 plotted points, thin points show 500 points (appearing as connected line). The eight panels have a plot range of $[-1.4, 1.4] \times [-1.4, 1.4]$. Triangles in center panel mark the morph positions corresponding to the delay embedding "snapshots". Center: the network observation signal $y(n)$ of a morph run. Bottom: Thin black line: the period length obtained from morphing between (and extrapolating beyond) the original period lengths. Bold gray line: period lengths measured from the observation signal $y(n)$.

Figure 15 shows the findings. The reservoir oscillates over the entire inter/extrapolation range with a waveform that is approximately equal to a sampled sine. At the morph values $\mu = 0$ and $\mu = 1$ (indicated by dots in the Figure), the system is in exactly the same modes as they were plotted earlier in the first two panels of the left column in Figure 12. Accordingly the fit between the original

driver's period lenghtes and the autonomously re-played oscillations is as good as it was reported there (i.e. corresponding to a steady-state NRMSE of about 0.01). In the extrapolation range, while the linear morphing of the mixing parameter μ does not lead to an exact linear morphing of the observed period lengths, still the obtained period lengths steadily continue to decrease (going left from $\mu = 0$) and to increase (going right from $\mu = 1$).

In sum, it is possible to use conceptor-morphing to extend sine-oscillatory reservoir dynamics from two learnt oscillations of periods $\approx 8.83, 9.83$ to a range between $\approx 7.5 - 11.9$ (minimal and maximal values of period lengths shown in the Figure). The post-training sinewave generation thus *extrapolated* beyond the period range spanned by the two training samples by a factor of about 4.4. From a perspective of machine learning this extrapolation is remarkable. Generally speaking, when neural pattern generators are trained from demonstration data (often done in robotics, e.g. [52, 89]), *interpolation* of recallable patterns is what one expects to achieve, while extrapolation is deemed hard.

From a perspective of neurodynamics, it is furthermore remarkable that the dimension of interpolation/extrapolation was the *speed* of the oscillation. Among the infinity of potential generalization dimensions of patterns, speedup/slowdown of pattern generation has a singular role and is particularly difficult to achieve. The reason is that speed cannot be modulated by postprocessing of some underlying generator's output – the prime generator itself must be modulated [108]. Frequency adaptation of neural oscillators is an important theme in research on biological pattern generators (CPGs) (reviews: [40, 50]). Frequency adaptation has been modeled in a number of ways, among which (i) to use a highly abstracted CPG model in the form of an ODE, and regulate speed by changing the ODE's time constant; (ii) to use a CPG model which includes a pacemaker neuron whose pace is adaptive; (iii) to use complex, biologically quite detailed, modular neural architectures in which frequency adapatation arises from interactions between modules, sensor-motoric feedback cycles, and tonic top-down input. However, the fact that humans can execute essentially arbitrary motor patterns at different speeds is not explained by these models. Presumably this requires a generic speed control mechanism which takes effect already at higher (cortical, planning) layers in the motor control hierarchy. Conceptor-controlled frequency adaptation might be of interest as a candidate mechanism for such a "cognitive-level" generic speed control mechanism.

Shape Morphing of an Integer-Periodic Pattern In this demonstration, the conceptors $C(R^3, 1000)$ and $C(R^4, 1000)$ from the 5-periodic patterns p^3 and p^4 were morphed, again with $-2 \leq \mu \leq 3$. Figure 16 depicts the network observer $y(n)$ for a morph run of 95 steps which was started with $\mu = -2$ and ended with $\mu = 3$, with a linear μ ramping in between. It can be seen that the differences between the two reference patterns (located at the points marked by dots) become increasingly magnified in both extrapolation segments. At each of the different

points in each 5-cycle, the "sweep" induced by the morphing is however neither linear nor of the same type across all 5 points of the period (right panel). A simple algebraic rule that would describe the geometric characteristics of such morphings cannot be given. I would like to say, it is "up to the discretion of the network's nonlinear dynamics" how the morphing command is interpreted; this is especially true for the extrapolation range. If reservoirs with a different initial random W^* are used, different morphing geometries arise, especially at the far ends of the extrapolation range (not shown).

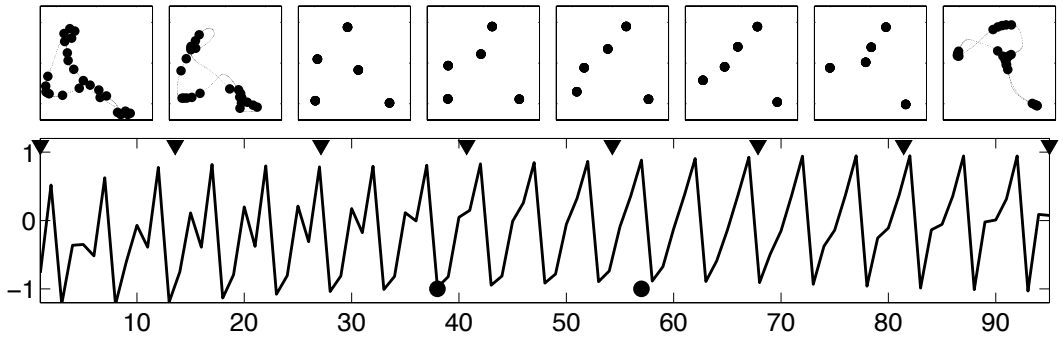

Figure 16: Morphing between, and extrapolating beyond, two versions of a 5-periodic random pattern. The morphing range was $-2 \leq \mu \leq 3$. Bottom: Network observation from a morphing run. Dots mark the points $\mu = 0$ and $\mu = 1$, corresponding to situations where the two original conceptors C^1, C^2 were active in unadulterated form. The network observation signal $y(n)$ is shown. Top: Delay-embedding "snapshots". Figure layout similar to Figure 15.

The snapshots displayed in Figure 16 reveal that the morphing sweep takes the reservoir through two bifurcations (apparent in the transition from snapshot 2 to 3, and from 7 to 8). In the intermediate morphing range (snapshots 3 – 7), we observe a discrete periodic attractor of 5 points. In the ranges beyond, on both sides the attracting set becomes topologically homomorphic to a continuous cycle. From a visual inspection, it appears that these bifurcations "smoothly" preserve some geometrical characteristics of the observed signal $y(n)$. A mathematical characterisation of this phenomenological continuity across bifurcations remains for future investigations.

Heterogeneous Pattern Morphing Figure 17 shows a morph from the 5-periodic pattern p^3 to the irrational-periodic sine p^2 (period length ≈ 9.83). This time the morphing range was $0 \leq \mu \leq 1$, (no extrapolation). The Figure shows a run with an initial 25 steps of $\mu = 0$, followed by a 50-step ramp to $\mu = 1$ and a tail of 25 steps at the same μ level. One observes a gradual change of signal shape and period along the morph. From a dynamical systems point of view this gradual change is unexpected. The reservoir is, mathematically speaking, an

autonomous system under the influence of a slowly changing control parameter μ. On both ends of the morph, the system is in an attractor. The topological nature of the attractors (seen as subsets of state space) is different (5 isolated points vs. a homolog of a 1-dim circle), so there must be a at least one bifurcation taking place along the morphing route. Such a bifurcations would usually be accompanied by a sudden change of some qualitative characteristic of the system trajectory. We find however no trace of a dynamic rupture, at least not by visual inspection of the output trajectory. Again, a more in-depth formal characterization of what geometric properties are "smoothly" carried through these bifurcations is left for future work.

Figure 2 in Section 1 is a compound demonstration of the three types of pattern morphing that I here discussed individually.

A possible application for the pattern morphing by conceptors is to effect smooth gait changes in walking robots, a problem that is receiving some attention in that field.

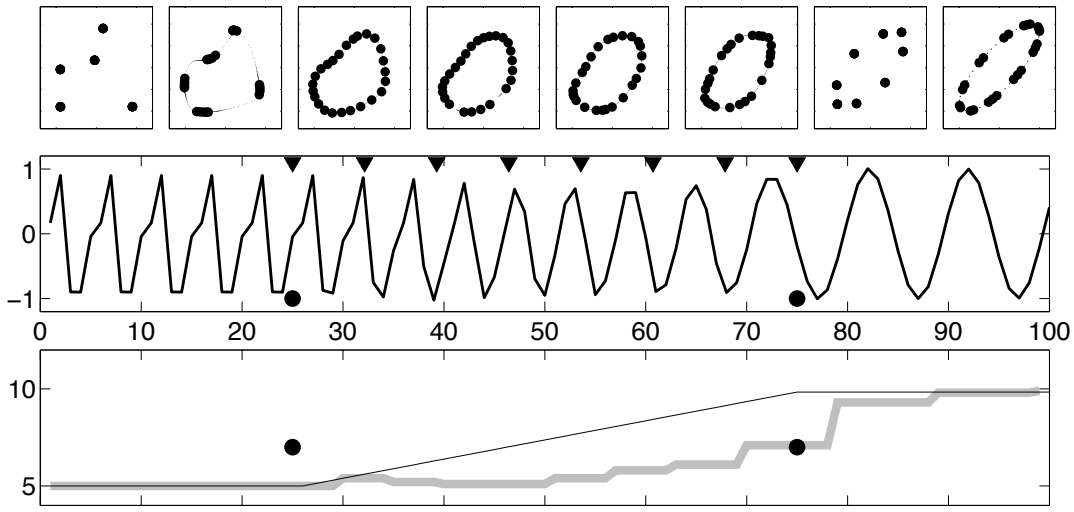

Figure 17: Morphing from a 5-periodic random pattern to an irrational-periodic sine. The morphing range was $0 \leq \mu \leq 1$. Figure layout otherwise is as in Figure 15.

3.8 Understanding Aperture

3.8.1 The Semantics of α as "Aperture"

Here I show how the parameter α can be interpreted as a scaling of signal energy, and motivate why I call it "aperture".

We can rewrite $C(R, \alpha) = C(E[xx'], \alpha)$ as follows:

$$\begin{aligned} C(E[xx'],\alpha) &= E[xx'](E[xx'] + \alpha^{-2}I)^{-1} = E[(\alpha x)(\alpha x)'](E[(\alpha x)(\alpha x)'] + I)^{-1} \\ &= C(E[(\alpha x)(\alpha x)'], 1) = C(\alpha^2\, E[xx'], 1). \end{aligned} \tag{14}$$

Thus, changing from $C(R,1)$ to $C(R,\alpha)$ can be interpreted as scaling the reservoir data by a factor of α, or expressed in another way, as scaling the signal energy of the reservoir signals by a factor of α^2. This is directly analog to what adjusting the aperture effects in an optical camera. In optics, the term *aperture* denotes the diameter of the effective lens opening, and the amount of light energy that reaches the film is proportional to the squared aperture. This has motivated the naming of the parameter α as *aperture*.

3.8.2 Aperture Adaptation and Final Definition of Conceptor Matrices

It is easy to verify that if $C_\alpha = C(R,\alpha)$ and $C_\beta = C(R,\beta)$ are two versions of a conceptor C differing in their apertures $0 < \alpha, \beta < \infty$, they are related to each other by

$$C_\beta = C_\alpha \left(C_\alpha + \left(\frac{\alpha}{\beta}\right)^2 (I - C_\alpha) \right)^{-1}, \tag{15}$$

where we note that $C_\alpha + (\alpha/\beta)^2(I - C_\alpha)$ is always invertible. C_β is thus a function of C_α and the ratio $\gamma = \beta/\alpha$. This motivates to introduce an *aperture adaptation* operation φ on conceptors C, as follows:

$$\varphi(C,\gamma) = C\,(C + \gamma^{-2}(I - C))^{-1}, \tag{16}$$

where $\varphi(C,\gamma)$ is the conceptor version obtained from C by adjusting the aperture of C by a factor of γ. Specifically, it holds that $C(R,\alpha) = \varphi(C(R,1),\alpha)$.

We introduce the notation $R_C = C(I - C)^{-1}$, which leads to the following easily verified data-based version of (16):

$$R_{\varphi(C,\gamma)} = \gamma^2\, R_C. \tag{17}$$

When we treat Boolean operations further below, it will turn out that the NOT operation will flip zero singular values of C to unit singular values. Because of this circumstance, we admit unit singular values in conceptors and formally define

Definition 2 *A* conceptor matrix *is a positive semidefinite matrix whose singular values range in* $[0,1]$. *We denote the set of all* $N \times N$ *conceptor matrices by* $\mathcal{C}_N$.

Note that Definition 1 defined the concept of a *conceptor matrix associated with a state correlation matrix* R *and an aperture* α, while Definition 2 specifies the more general class of *conceptor matrices*. Mathematically, conceptor matrices (as in Definition 2) are more general than the conceptor matrices associated with a state correlation matrix R, in that the former may contain unit singular values.

Furthermore, in the context of Boolean operations it will also become natural to admit aperture adaptations of sizes $\gamma = 0$ and $\gamma = \infty$. The inversion in Equation (16) is not in general well-defined for such γ and/or conceptors with unit singular values, but we can generalize those relationships to the more general versions of conceptors and aperture adaptations by a limit construction:

Definition 3 *Let C be a conceptor and $\gamma \in [0, \infty]$. Then*

$$\varphi(C, \gamma) = \begin{cases} C\,(C + \gamma^{-2}(I - C))^{-1} & \text{for } 0 < \gamma < \infty \\ \lim_{\delta \downarrow 0} C\,(C + \delta^{-2}(I - C))^{-1} & \text{for } \gamma = 0 \\ \lim_{\delta \uparrow \infty} C\,(C + \delta^{-2}(I - C))^{-1} & \text{for } \gamma = \infty \end{cases} \tag{18}$$

It is a mechanical exercise to show that the limits in (18) exist, and to calculate the singular values for $\varphi(C, \gamma)$. The results are collected in the following proposition.

Proposition 3 *Let $C = USU'$ be a conceptor and $(s_1, \ldots, s_N)' = \mathrm{diag}S$ the vector of its singular values. Let $\gamma \in [0, \infty]$. Then $\varphi(C, \gamma) = US_\gamma U'$ is the conceptor with singular values $(s_{\gamma,1}, \ldots, s_{\gamma,N})'$, where*

$$s_{\gamma,i} = \begin{cases} s_i/(s_i + \gamma^{-2}(1 - s_i)) & \text{for } \; 0 < s_i < 1,\; 0 < \gamma < \infty \\ 0 & \text{for } \; 0 < s_i < 1,\; \gamma = 0 \\ 1 & \text{for } \; 0 < s_i < 1,\; \gamma = \infty \\ 0 & \text{for } \; s_i = 0,\; 0 \leq \gamma \leq \infty \\ 1 & \text{for } \; s_i = 1,\; 0 \leq \gamma \leq \infty \end{cases} \tag{19}$$

Since aperture adaptation of $C = USU'$ only changes the singular values of C, the following fact is obvious:

Proposition 4 *If V is orthonormal, then $\varphi(VCV', \gamma) = V\,\varphi(C, \gamma)\,V'$.*

Iterated application of aperture adaptation corresponds to multiplying the adaptation factors:

Proposition 5 *Let C be a conceptor and $\gamma, \beta \in [0, \infty]$. Then $\varphi(\varphi(C, \gamma), \beta) = \varphi(C, \gamma\beta)$.*

The proof is a straightforward algebraic verification using (19).

Borrowing again terminology from photography, I call a conceptor with SVD $C = USU'$ *hard* if all singular values in S are 0 or 1 (in photography, a film with an extremely "hard" gradation yields pure black-white images with no gray tones.) Note that C is hard if and only if it is a projector matrix. If C is hard, the following holds:

$$C = C^\dagger = C' = CC, \tag{20}$$

$$\varphi(C, \gamma) = C \quad \text{for } \gamma \in [0, \infty]. \tag{21}$$

The first claim amounts to stating that C is a projection operator, which is obviously the case; the second claim follows directly from (19).

Besides the aperture, another illuminating characteristic of a conceptor matrix is the mean value of its singular values, i.e. its normalized trace $q(C) = \text{trace}(C)/N$. It ranges in $[0, 1]$. Intuitively, this quantity measures the fraction of dimensions from the N-dimensional reservoir state space that is claimed by C. I will call it the *quota* of C.

3.8.3 Aperture Adaptation: Example

In applications one will often need to adapt the aperture to optimize the quality of C. What "quality" means depends on the task at hand. I present an illustrative example, where the reservoir is loaded with very fragile patterns. Retrieving them requires a prudent choice of α. Specifically, I loaded a reservoir of size $N = 500$ with four chaotic patterns, derived from the well-known Rössler, Lorenz, Mackey-Glass, and Hénon attractors (details of this example are given in Section 4.2). Four conceptors C_R, C_L, C_{MG}, C_H were computed, one for each attractor, using $\alpha = 1$. Then, in four retrieval experiments, the aperture of each of these was adapted using (16) in a geometric succession of five different γ, yielding five versions of each of the C_R, C_L, C_{MG}, C_H. Each of these was used in turn for a constrained run of the reservoir according to the state update rule $x(n+1) = C\ \tanh(W\, x(n) + W^{\text{in}}\, p(n+1) + b)$, and the resulting output observation was plotted in a delay-embedding format.

Figure 18 displays the findings. Per each attractor, the five apertures were hand-selected such that the middle one (the third) best re-generated the original chaotic signal, while the first failed to recover the original. One should mention that it is not trivial in the first place to train an RNN to stably generate any single chaotic attractor timeseries, but here we require the loaded network to be able to generate any one out of four such signals, only by constraining the reservoir by a conceptor with a suitably adapted aperture. Number insets in the panels of figure 18 indicate the apertures and quotas used per run.

3.8.4 Guides for Aperture Adjustment

The four chaotic attractors considered in the previous subsection were "best" (according to visual inspection) reconstructed with apertures between 630 and 1000. A well-chosen aperture is clearly important for working with conceptors. In all demonstrations reported so far I chose a "good" aperture based on experimentation and human judgement. In practice one will often need automated criteria for optimizing the aperture which do not rely on human inspection. In this subsection I propose two measures which can serve as such a guiding criterion.

A criterion based on reservoir-conceptor interaction. Introducing an interim state variable $z(n)$ by splitting the conceptor-constrained reservoir update

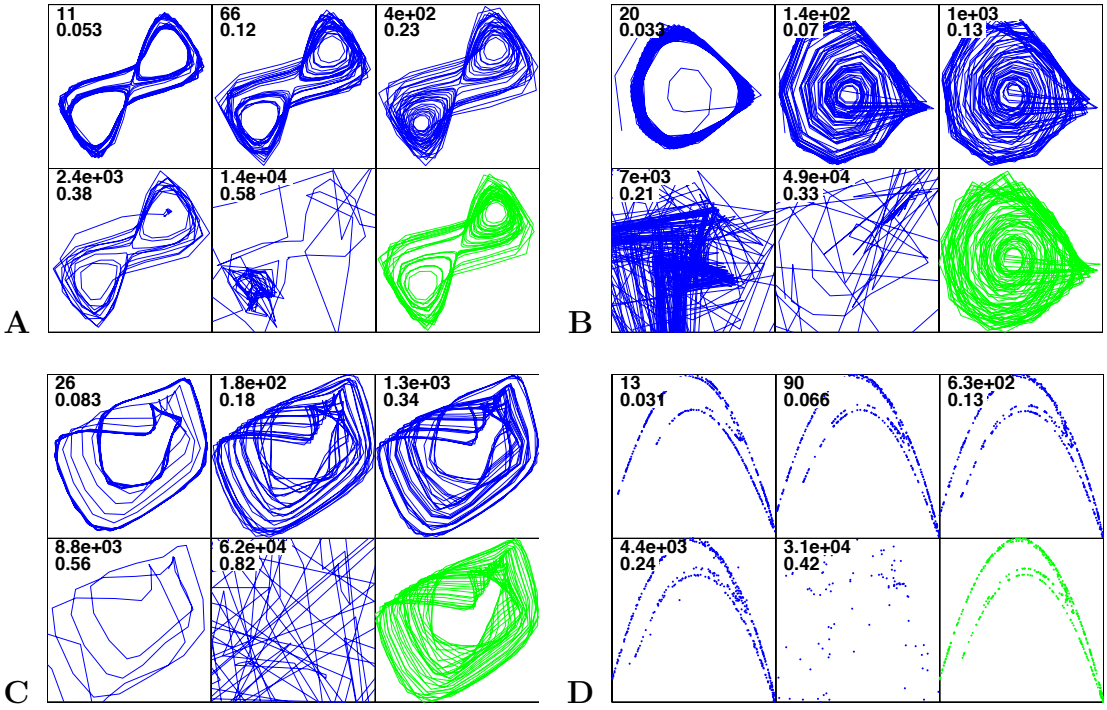

Figure 18: Invoking conceptors to retrieve four chaotic signals from a reservoir. **A** Lorenz, **B** Rössler, **C** Mackey-Glass, and **D** Hénon attractor. All four are represented by delay-embedding plots of the reservoir observation signal $y(n)$. The plot range is $[0,1] \times [0,1]$ in every panel. **A** – **C** are attractors derived from differential equations, hence subsequent points are joined with lines; **D** derives from an iterated map where joining lines has no meaning. Each 6-panel block shows five patterns generated by the reservoir under the control of differently aperture-adapted versions of a conceptor (blue, first five panels) and a plot of the original chaotic reference signal (green, last panel). Empty panels indicate that the $y(n)$ signal was outside the $[0,1]$ range. The right upper panel in each block shows a version which, judged by visual inspection, comes satisfactorily close to the original. First number given in a panel: aperture α; second number: quota $q(C)$.

equation (9) into

$$z(n+1) = tanh(W\,x(n) + b), \quad x(n+1) = C(R,\alpha)\,z(n+1), \tag{22}$$

I define the *attenuation* measurable a as

$$a_{C,\alpha} = E[\|z(n) - x(n)\|^2]/E[\|z(n)\|^2], \tag{23}$$

where the states $x(n), z(n)$ are understood to result from a reservoir constrained by $C(R,\alpha)$. The attenuation is the fraction of the reservoir signal energy which is suppressed by applying the conceptor. Another useful way to conceive of this

quantity is to view it as noise-to-signal ratio, where the "noise" is the component $z(n) - x(n)$ which is filtered away from the unconstrained reservoir signal $z(n)$. It turns out in simulation experiments that when the aperture is varied, the attenuation $a_{C,\alpha}$ passes through a minimum, and at this minimum, the pattern reconstruction performance peaks.

In Figure 19**A** the $\log_{10}$ of $a_{C,\alpha}$ is plotted for a sweep through a range of apertures α, for each of the four chaotic attractor conceptors (details in Section 4.2). As α grows, the attenuation $a_{C,\alpha}$ first declines roughly linearly in the log-log plots, that is, by a power law of the form $a_{C,\alpha} \sim \alpha^{-K}$. Then it enters or passes through a trough. The aperture values that yielded visually optimal reproductions of the chaotic patterns coincide with the point where the bottom of the trough is reached.

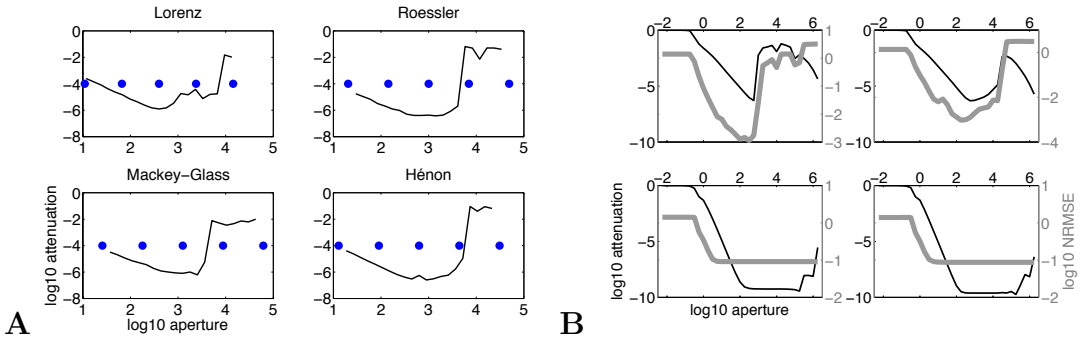

Figure 19: Using attenuation to locate optimal apertures. **A** Dependancy of attenuation on aperture for the four chaotic attractors. The blue dots mark the apertures used to generate the plots in Figure 18. **B** Dependancy of attenuation on aperture for the two sinewaves (top panels) and the two 5-point periodic patterns (bottom) used in Sections 3.2*ff*. These plots also provide the NRMSEs for the accuracy of the reconstructed patterns (gray). For explanation see text.

Figure 19**B** gives similar plots for the two irrational-period sines and the two 5-point periodic patterns treated in earlier sections. The same reservoir and storing procedures as described at that place were utilized again here. The dependence of attenuation on aperture is qualitatively the same as in the chaotic attractor example. The attenuation plots are overlaid with the NRMSEs of the original drivers vs. the conceptor-constrained reservoir readout signals. Again, the "best" aperture – here quantified by the NRMSE – coincides remarkably well with the trough minimum of the attenuation.

Some peculiarities visible in the plots **B** deserve a short comment. (i) The initial constant plateaus in all four plots result from $C(R, \alpha) \approx 0$ for the very small apertures in this region, which leads to $x(n) \approx 0, z(n) \approx \tanh(b)$. (ii) The jittery climb of the attenuation towards the end of the plotting range in the two bottom panels is an artefact due to roundoff errors in SVD computations which blows up singular values in conceptors which in theory should be zero. Without rounding

error involved, the attenuation plots would remain at their bottom value once it is reached. (iii) In the top two panels, some time after having passed through the trough the attenuation value starts to decrease again. This is due to the fact that for the irrational-period sinewave signals, all singular values of the conceptors are nonzero. As a consequence, for increasingly large apertures the conceptors will converge to the identity matrix, which would have zero attenuation.

A criterion based on conceptor matrix properties. A very simple criterion for aperture-related "goodness" of a conceptor can be obtained from monitoring the gradient of the squared Frobenius norm

$$\nabla(\gamma) = \frac{d}{d\ \log(\gamma)}\ \|\phi(C,\gamma)\|^2 \tag{24}$$

with respect to the logarithm of γ. To get an intuition about the semantics of this criterion, assume that C has been obtained from data with a correlation matrix R with SVD $R = U\Sigma U'$. Then $\phi(C,\gamma) = R(R+\gamma^{-2}I)^{-1}$ and $\|\phi(C,\gamma)\|^2 = \|\Sigma(\Sigma+\gamma^{-2}I)^{-1}\|^2 = \|\gamma^2\Sigma(\gamma^2\Sigma+I)^{-1}\|^2$. That is, $\phi(C,\gamma)$ can be seen as obtained from data scaled by a factor of γ compared to $\phi(C,1) = C$. The criterion $\nabla(\gamma)$ therefore measures the sensitivity of (the squared norm of) C on (expontential) scalings of data. Using again photography as a metaphor: if the aperture of a lens is set to the value where $\nabla(\gamma)$ is maximal, the sensitivity of the image (= conceptor) to changes in brightness (= data scaling) is maximal.

Figure 20 shows the behavior of this criterion again for the standard example of loading two irrational sines and two integer-periodic random patterns. Its maxima coincide largely with the minima of the attenuation criterion, and both with what was "best" performance of the respective pattern generator. The exception is the two integer-periodic patterns (Figure 20 **B** bottom panels) where the ∇ criterion would suggest a slightly too small aperture.

Comments on criteria for guiding aperture selection:

- The two presented criteria based on attenuation and norm gradient are purely heuristic. A theoretical analysis would require a rigorous definition of "goodness". Since tasks vary in their objectives, such an analysis would have to be carried out on a case-by-case basis for varying "goodness" characterizations. Other formal criteria besides the two presented here can easily be construed (I experimented with dozens of alternatives (not documented), some of which performed as well as the two instances reported here). Altogether this appears to be a wide field for experimentation.

- The attenuation-based criterion needs trial runs with the reservoir to be calculated, while the norm-gradient criterion can be computed offline. The former seems to be particularly suited for pattern-generation tasks where the conceptor-reservoir feedback loop is critical (for instance, with respect to stability). The latter may be more appropriate in machine learning tasks

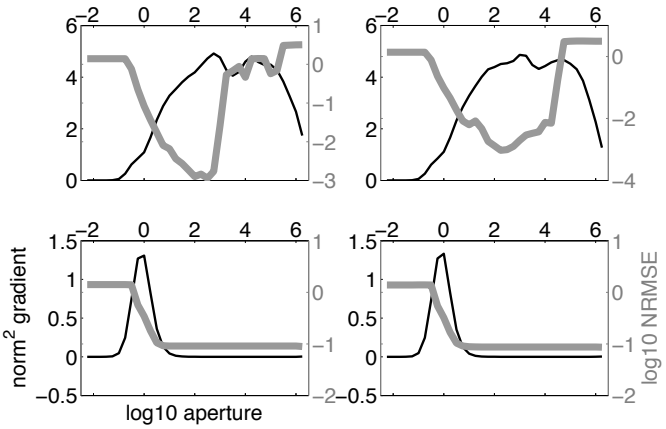

Figure 20: The norm-gradient based criterion to determine "good" apertures for the basic demo example from Sections 3.2 and 3.4. Plots show $\nabla(\gamma)$ against the $\log_1 0$ of aperture γ. Figure layout similar as in Figure 19 **B**. For explanation see text.

where conceptors are used for classifying reservoir dynamics in a "passive" way without coupling the conceptors into the network updates. I will give an example in Section 3.12.

3.9 Boolean Operations on Conceptors

3.9.1 Motivation

Conceptor matrices can be submitted to operations that can be meaningfully called AND, OR, and NOT. There are two justifications for using these classical logical terms:

Syntactical / algebraic: Many algebraic laws governing Boolean algebras are preserved; for hard conceptor matrices the preservation is exact.

Semantical: These operations on conceptor matrices correspond dually to operations on the data that give rise to the conceptors via (7). Specifically, the OR operation can be semantically interpreted on the data level by merging two datasets, and the NOT operation by inverting the principal component weights of a dataset. The AND operation can be interpreted on the data level by combining de Morgan's rule (which states that $x \wedge y = \neg(\neg x \vee \neg y)$) with the semantic interpretations of OR and NOT.

The mathematical structures over conceptors that arise from the Boolean operations are richer than standard Boolean logic, in that aperture adaptation operations can be included in the picture. One obtains a formal framework which one might call "adaptive Boolean logic".

There are two major ways how such a theory of conceptor logic may be useful:

A logic for information processing in RNNs (cognitive and neuroscience): The dynamics of any N-dimensional RNN (of any kind, autonomously active or driven by external input), when monitored for some time period L, yields an $N \times L$ sized state collection matrix X and its corresponding $N \times N$ correlation matrix R, from which a conceptor matrix $C = R(R + I)^{-1}$ can be obtained which is a "fingerprint" of the activity of the network for this period. The Boolean theory of conceptors can be employed to analyse the relationships between such "activity fingerprints" obtained at different intervals, different durations, or from different driving input. An interesting long-term research goal for cognitive neuroscience would be to map the logical structuring described on the network data level, to Boolean operations carried out by task-performing subjects.

An algorithmical tool for RNN control (machine learning): By controlling the ongoing activity of an RNN in a task through conceptors which are derived from logical operations, one can implement "logic control" strategies for RNNs. Examples will be given in Section 3.11, where Boolean operations on conceptors will be key for an efficient memory management in RNNs; in Section 3.12, where Boolean operations will enable to combine positive and negative evidences for finite-duration pattern recognition; and in Section 3.15, where Boolean operations will help to simultaneously de-noise and classify signals.

3.9.2 Preliminary Definition of Boolean Operations

Defining Boolean operators through their data semantics is transparent and simple when the concerned data correlation matrices are nonsingular. In this case, the resulting conceptor matrices are nonsingular too and have singular values ranging in the open interval $(0, 1)$. I treat this situation in this subsection. However, conceptor matrices with a singular value range of $[0, 1]$ frequently arise in practice. This leads to technical complications which will be treated in the next subsection. The definitions given in the present subsection are preliminary and serve expository purposes.

In the remainder of this subsection, conceptor matrices C, B are assumed to derive from nonsingular correlation matrices.

I begin with OR. Recall that a conceptor matrix C (with aperture 1) derives from a data source (network states) x through $R = E[xx'], C = C(R, 1) = R(R + I)^{-1}$. Now consider a second conceptor B of the same dimension N as C, derived from another data source y by $Q = E[yy'], B = B(Q, 1) = Q(Q + I)^{-1}$. I define

$$C \vee B := (R + Q)(R + Q + I)^{-1}, \tag{25}$$

and name this the OR operation. Observe that $R + Q = E[[x, y][x, y]']$, where $[x, y]$ is the $N \times 2$ matrix made of vectors x, y. $C \vee B$ is thus obtained by a merge of the two data sources which previously went into C and B, respectively. This provides a semantic interpretation of the OR operation.

Using (7), it is straightforward to verify that $C \vee B$ can be directly computed from C and B by

$$C \vee B = \left(I + \left(C(I - C)^{-1} + B(I - B)^{-1}\right)^{-1}\right)^{-1}, \tag{26}$$

where the assumption of nonsingular R, Q warrants that all inverses in this equation are well-defined.

I now turn to the NOT operation. For $C = C(R, 1) = R(R + I)^{-1}$ with nonsingular R I define it by

$$\neg C := R^{-1}(R^{-1} + I)^{-1}. \tag{27}$$

Again this can be semantically interpreted on the data level. Consider the SVDs $R = U\Sigma U'$ and $R^{-1} = U\Sigma^{-1}U'$. R and R^{-1} have the same principal components U, but the variances Σ, Σ^{-1} of data that would give rise to R and R^{-1} are inverse to each other. In informal terms, $\neg C$ can be seen as arising from data which co-vary inversely compared to data giving rise to C.

Like in the case of OR, the negation of C can be computed directly from C. It is easy to see that

$$\neg C = I - C. \tag{28}$$

Finally, I consider AND. Again, we introduce it on the data level. Let again $C = R(R + I)^{-1}, B = Q(Q + I)^{-1}$. The OR operation was introduced as addition on data correlation matrices, and the NOT operation as inversion. Guided by de Morgan's law $a \wedge b = \neg(\neg a \vee \neg b)$ from Boolean logic, we obtain a correlation matrix $(R^{-1} + Q^{-1})^{-1}$ for $C \wedge B$. Via (7), from this correlation matrix we are led to

$$C \wedge B := (R^{-1} + Q^{-1})^{-1}\left((R^{-1} + Q^{-1})^{-1} + I\right)^{-1}. \tag{29}$$

Re-expressing R, Q in terms of C, B in this equation, elementary transformations (using (7)) again allow us to compute AND directly:

$$C \wedge B = (C^{-1} + B^{-1} - I)^{-1}. \tag{30}$$

By a routine transformation of equations, it can be verified that the de Morgan's laws $C \vee B = \neg(\neg C \wedge \neg B)$ and $C \wedge B = \neg(\neg C \vee \neg B)$ hold for the direct computation expressions (26), (28) and (30).

3.9.3 Final Definition of Boolean Operations

We notice that the direct computations (26) and (30) for OR and AND are only well-defined for conceptor matrices whose singular values range in $(0, 1)$. I now generalize the definitions for AND and OR to cases where the concerned conceptors may contain singular values 0 or 1. Since the direct computation (30) of AND is simpler than the direct computation (26) of OR, I carry out the generalization for AND and then transfer it to OR through de Morgan's rule.

Assume that $C = USU', B = VTV'$ are the SVDs of conceptors C, B, where S and/or T may contain zero singular values. The direct computation (30) is then not well-defined.

Specifically, assume that $\mathrm{diag}(S)$ contains $l \leq N$ nonzero singular values and that $\mathrm{diag}(T)$ contains $m \leq N$ nonzero singular values, i.e. $\mathrm{diag}(S) = (s_1, \ldots, s_l, 0, \ldots, 0)'$ and $\mathrm{diag}(T) = (t_1, \ldots, t_m, 0, \ldots, 0)'$. Let δ be a positive real number. Define S_δ to be the diagonal matrix which has a diagonal $(s_1, \ldots, s_l, \delta, \ldots, \delta)'$, and similarly T_δ to have diagonal $(t_1, \ldots, t_m, \delta, \ldots, \delta)'$. Put $C_\delta = US_\delta U', B_\delta = VT_\delta V'$. Then $C_\delta \wedge B_\delta = (C_\delta^{-1} + B_\delta^{-1} - I)^{-1}$ is well-defined. We now define

$$C \wedge B = \lim_{\delta \to 0} (C_\delta^{-1} + B_\delta^{-1} - I)^{-1}. \tag{31}$$

The limit in this equation is well-defined and can be resolved into an efficient algebraic computation:

Proposition 6 *Let $\mathbf{B}_{\mathcal{R}(C) \cap \mathcal{R}(B)}$ be a matrix whose columns form an arbitrary orthonormal basis of $\mathcal{R}(C) \cap \mathcal{R}(B)$. Then, the matrix $\mathbf{B}'_{\mathcal{R}(C) \cap \mathcal{R}(B)}(C^\dagger + B^\dagger - I)\mathbf{B}_{\mathcal{R}(C) \cap \mathcal{R}(B)}$ is invertible, and the limit (31) exists and is equal to*

$$\begin{aligned} C \wedge B &= \lim_{\delta \to 0} (C_\delta^{-1} + B_\delta^{-1} - I)^{-1} = \\ &= \mathbf{B}_{\mathcal{R}(C) \cap \mathcal{R}(B)} \left(\mathbf{B}'_{\mathcal{R}(C) \cap \mathcal{R}(B)} \left(C^\dagger + B^\dagger - I \right) \mathbf{B}_{\mathcal{R}(C) \cap \mathcal{R}(B)} \right)^{-1} \mathbf{B}'_{\mathcal{R}(C) \cap \mathcal{R}(B)}. \end{aligned} \tag{32}$$

Equivalently, let $\mathbf{P}_{\mathcal{R}(C) \cap \mathcal{R}(B)} = \mathbf{B}_{\mathcal{R}(C) \cap \mathcal{R}(B)} \mathbf{B}'_{\mathcal{R}(C) \cap \mathcal{R}(B)}$ be the projector matrix on $\mathcal{R}(C) \cap \mathcal{R}(B)$. Then $C \wedge B$ can also be written as

$$C \wedge B = \left(\mathbf{P}_{\mathcal{R}(C) \cap \mathcal{R}(B)} \left(C^\dagger + B^\dagger - I \right) \mathbf{P}_{\mathcal{R}(C) \cap \mathcal{R}(B)} \right)^\dagger. \tag{33}$$

The proof and an algorithm to compute a basis matrix $\mathbf{B}_{\mathcal{R}(C) \cap \mathcal{R}(B)}$ are given in Section 5.2.

The formulas (32) and (33) not only extend the formula (30) to cases where C or B are non-invertible, but also ensures numerical stability in cases where C or B are ill-conditioned. In that situation, the pseudoinverses appearing in (32), (33) should be computed with appropriate settings of the numerical tolerance which one can specify in common implementations (for instance in Matlab) of the SVD. One should generally favor (32) over (30) unless one can be sure that C and B are well-conditioned.

The direct computation (28) of NOT is well-defined for C with a singular value range $[0, 1]$, thus nothing remains to be done here.

Having available the general and numerically robust computations of AND via (32) or (33) and of NOT via (28), we invoke de Morgan's rule $C \vee B = \neg(\neg C \wedge \neg B)$ to obtain a general and robust computation for OR on the basis of (32) resp. (33) and (28). Summarizing, we obtain the final definitions for Boolean operations on conceptors:

Definition 4

$$\begin{aligned} \neg C &:= I - C, \\ C \wedge B &:= \left(\mathbf{P}_{\mathcal{R}(C)\cap\mathcal{R}(B)} \left(C^\dagger + B^\dagger - I\right) \mathbf{P}_{\mathcal{R}(C)\cap\mathcal{R}(B)}\right)^\dagger, \\ C \vee B &:= \neg\left(\neg C \wedge \neg B\right), \end{aligned}$$

where $\mathbf{P}_{\mathcal{R}(C)\cap\mathcal{R}(B)}$ *is the projector matrix on* $\mathcal{R}(C) \cap \mathcal{R}(B)$.

This definition is consistent with the preliminary definitions given in the previous subsection. For AND this is clear: if C and B are nonsingular, $\mathbf{P}_{\mathcal{R}(C)\cap\mathcal{R}(B)}$ is the identity and the pseudoinverse is the inverse, hence (30) is recovered (fact 1). We noted in the previous subsection that de Morgan's laws hold for conceptors derived from nonsingular correlation matrices (fact 2). Furthermore, if C is derived from a nonsingular correlation matrix, then $\neg C$ also corresponds to a nonsingular correlation matrix (fact 3). Combining facts 1 – 3 yields that the way of defining OR via de Morgan's rule from AND and NOT in Definition 4 generalises (25)/(26).

For later use I state a technical result which gives a characterization of OR in terms of a limit over correlation matrices:

Proposition 7 *For a conceptor matrix* C *with SVD* $C = USU'$ *let* $S^{(\delta)}$ *be a version of* S *where all unit singular values (if any) have been replaced by* $1 - \delta$, *and let* $C^{(\delta)} = US^{(\delta)}U'$. *Let* $R_C^{(\delta)} = C^{(\delta)}(I - C^{(\delta)})^{-1}$. *Similarly, for another conceptor* B *let* $R_B^{(\delta)} = B^{(\delta)}(I - B^{(\delta)})^{-1}$. *Then*

$$C \vee B = I - \lim_{\delta \downarrow 0} (R_C^{(\delta)} + R_B^{(\delta)} + I)^{-1} = \lim_{\delta \downarrow 0} (R_C^{(\delta)} + R_B^{(\delta)}) (R_C^{(\delta)} + R_B^{(\delta)} + I)^{-1}. \quad (34)$$

The proof is given in Section 5.3. Finally I note that de Morgan's rule also holds for AND (proof in Section 5.4):

Proposition 8

$$C \wedge B = \neg\left(\neg C \vee \neg B\right).$$

3.9.4 Facts Concerning Subspaces

For an $N \times N$ matrix M, let $\mathcal{I}(M) = \{x \in \mathbb{R}^N \mid Mx = x\}$ be the *identity space* of M. This is the eigenspace of M to the eigenvalue 1, a linear subspace of $\mathbb{R}^N$. The identity spaces, null spaces, and ranges of conceptors are related to Boolean operations in various ways. The facts collected here are technical, but will be useful in deriving further results later.

Proposition 9 *Let C, B be any conceptor matrices, and H, G hard conceptor matrices of the same dimension. Then the following facts hold:*

1. $\mathcal{I}(C) \subseteq \mathcal{R}(C)$.
2. $\mathcal{I}(C^\dagger) = \mathcal{I}(C)$ *and* $\mathcal{R}(C^\dagger) = \mathcal{R}(C)$ *and* $\mathcal{N}(C^\dagger) = \mathcal{N}(C)$.
3. $\mathcal{R}(\neg C) = \mathcal{I}(C)^\perp$ *and* $\mathcal{I}(\neg C) = \mathcal{N}(C)$ *and* $\mathcal{N}(\neg C) = \mathcal{I}(C)$.
4. $\mathcal{R}(C \wedge B) = \mathcal{R}(C) \cap \mathcal{R}(B)$ *and* $\mathcal{R}(C \vee B) = \mathcal{R}(C) + \mathcal{R}(B)$.
5. $\mathcal{I}(C \wedge B) = \mathcal{I}(C) \cap \mathcal{I}(B)$ *and* $\mathcal{I}(C \vee B) = \mathcal{I}(C) + \mathcal{I}(B)$.
6. $\mathcal{N}(C \wedge B) = \mathcal{N}(C) + \mathcal{N}(B)$ *and* $\mathcal{N}(C \vee B) = \mathcal{N}(C) \cap \mathcal{N}(B)$.
7. $\mathcal{I}(\varphi(C, \gamma)) = \mathcal{I}(C)$ *for* $\gamma \in [0, \infty)$ *and* $\mathcal{R}(\varphi(C, \gamma)) = \mathcal{R}(C)$ *for* $\gamma \in (0, \infty]$ *and* $\mathcal{N}(\varphi(C, \gamma)) = \mathcal{N}(C)$ *for* $\gamma \in (0, \infty]$.
8. $A = A \wedge C \iff \mathcal{R}(A) \subseteq \mathcal{I}(C)$ *and* $A = A \vee C \iff \mathcal{I}(A)^\perp \subseteq \mathcal{N}(C)$.
9. $\varphi(C, 0)$ *and* $\varphi(C, \infty)$ *are hard.*
10. $\varphi(C, 0) = \mathbf{P}_{\mathcal{I}(C)}$ *and* $\varphi(C, \infty) = \mathbf{P}_{\mathcal{R}(C)}$.
11. $H = H^\dagger = \mathbf{P}_{\mathcal{I}(H)}$.
12. $\mathcal{I}(H) = \mathcal{R}(H) = \mathcal{N}(H)^\perp$.
13. $\neg H = \mathbf{P}_{\mathcal{N}(H)} = \mathbf{P}_{\mathcal{I}(H)^\perp}$.
14. $H \wedge G = \mathbf{P}_{\mathcal{I}(H) \cap \mathcal{I}(G)}$.
15. $H \vee G = \mathbf{P}_{\mathcal{I}(H) + \mathcal{I}(G)}$.

The proof is given in Section 5.5.

3.9.5 Boolean Operators and Aperture Adaptation

The Boolean operations are related to aperture adaptation in a number of ways:

Proposition 10 *Let C, B be $N \times N$ sized conceptor matrices and $\gamma, \beta \in [0, \infty]$. We declare $\infty^{-1} = \infty^{-2} = 0$ and $0^{-1} = 0^{-2} = \infty$. Then,*

1. $\neg\varphi(C, \gamma) = \varphi(\neg C, \gamma^{-1})$,
2. $\varphi(C, \gamma) \vee \varphi(B, \gamma) = \varphi(C \vee B, \gamma)$,
3. $\varphi(C, \gamma) \wedge \varphi(B, \gamma) = \varphi(C \wedge B, \gamma)$,
4. $\varphi(C, \gamma) \vee \varphi(C, \beta) = \varphi(C, \sqrt{\gamma^2 + \beta^2})$,
5. $\varphi(C, \gamma) \wedge \varphi(C, \beta) = \varphi(C, (\gamma^{-2} + \beta^{-2})^{-1/2})$.

The proof can be found in Section 5.6. Furthermore, with the aid of aperture adaptation and OR it is possible to implement an incremental model extension, as follows. Assume that conceptor C has been obtained from a dataset X comprised of m data vectors x, via $R = XX'/m, \;\; C = R(R + \alpha^{-2} I)^{-1}$. Then, n new data vectors y become available, collected as columns in a data matrix Y. One wishes to update the original conceptor C such that it also incorporates the information from Y, that is, one wishes to obtain

$$\tilde{C} = \tilde{R}(\tilde{R} + \alpha^{-2} I)^{-1}, \tag{35}$$

where $\tilde{R}$ is the updated correlation matrix obtained by $Z = [XY], \tilde{R} = ZZ'/(m + n)$. But now furthermore assume that the original training data X are no longer available. This situation will not be uncommon in applications. The way to a direct computation of (35) is barred. In this situation, the extended model $\tilde{C}$ can be computed from C, Y, m, n as follows. Let $C_Y = YY'(YY' + I)^{-1}$. Then,

$$\begin{aligned} \tilde{C} &= \varphi\left(\varphi(C,\; m^{1/2}\alpha^{-1}) \vee C_Y,\; (m+n)^{1/2}\alpha\right) && (36) \\ &= I - \left(\frac{m}{m+n}(I - C)^{-1}C + \frac{n}{m+n}\alpha^2 YY' + I\right)^{-1}. && (37) \end{aligned}$$

These formulas can be verified by elementary transformations using (7), (8), (25) and (17), noting that C cannot have unit singular values because it is obtained from a bounded correlation matrix R, thus $(I - C)$ is invertible.

3.9.6 Logic Laws

Many laws from Boolean logic carry over to the operations AND, OR, NOT defined for conceptors, sometimes with modifications.

Proposition 11 *Let I be the $N \times N$ identity matrix, 0 the zero matrix, and B, C, D any conceptor matrices of size $N \times N$ (including I or 0). Then the following laws hold:*

1. *De Morgan's rules: $C \vee B = \neg\,(\neg C \,\wedge\, \neg B)$ and $C \wedge B = \neg\,(\neg C \,\vee\, \neg B)$.*
2. *Associativity: $(B \wedge C) \wedge D = B \wedge (C \wedge D)$ and $(B \vee C) \vee D = B \vee (C \vee D)$.*
3. *Commutativity: $B \wedge C = C \wedge B$ and $B \vee C = C \vee B$.*
4. *Double negation: $\neg(\neg C) = C$.*
5. *Neutrality of 0 and I: $C \vee 0 = C$ and $C \wedge I = C$.*
6. *Globality of 0 and I: $C \vee I = I$ and $C \wedge 0 = 0$.*
7. *Weighted self-absorption for OR: $C \vee C = \varphi(C, \sqrt{2})$ and $\varphi(C, \sqrt{1/2}) \vee \varphi(C, \sqrt{1/2}) = C$.*
8. *Weighted self-absorption for AND: $C \wedge C = \varphi(C, 1/\sqrt{2})$ and $\varphi(C, \sqrt{2}) \wedge \varphi(C, \sqrt{2}) = C$.*

The proofs are given in Section 5.7. From among the classical laws of Boolean logic, the general absorption rules $A = A \wedge (A \vee B) = A \vee (A \wedge B)$ and the laws of distributivity do *not* hold in general for conceptors. However, they hold for hard conceptors, which indeed form a Boolean algebra:

Proposition 12 *The set of hard $N \times N$ conceptor matrices, equipped with the operations $\vee, \wedge, \neg$ defined in Definition 4, is a Boolean algebra with maximal element $I_{N\times N}$ and minimal element $0_{N\times N}$.*

The proof is obvious, exploiting the fact that hard conceptors can be identified with (projectors on) their identity subspaces (Proposition 9 *11.*), and that the operations $\vee, \wedge, \neg$ correspond to subspace operations $+, \cap, \cdot^{\perp}$ (Proposition 9 *13.* – *15.*). It is well known that the linear subspaces of $\mathbb{R}^N$ form a Boolean algebra with respect to these subspace operations.

While the absorption rules $A = A \wedge (A \vee B) = A \vee (A \wedge B)$ are not valid for conceptor matrices, it is possible to "invert" $\vee$ by $\wedge$ and vice versa in a way that is reminiscent of absorption rules:

Proposition 13 *Let A, B be conceptor matrices of size $N \times N$. Then,*

1. *$C = \left(\mathbf{P}_{\mathcal{R}(A)} \left(I + A^{\dagger} - (A \vee B)^{\dagger}\right) \mathbf{P}_{\mathcal{R}(A)}\right)^{\dagger}$ is a conceptor matrix and*
$$A = (A \vee B) \wedge C. \tag{38}$$
2. *$C = I - \left(\mathbf{P}_{\mathcal{I}(A)^{\perp}} \left(I + (I - A)^{\dagger} - (I - (A \wedge B))^{\dagger}\right) \mathbf{P}_{\mathcal{I}(A)^{\perp}}\right)^{\dagger}$ is a conceptor matrix and*
$$A = (A \wedge B) \vee C. \tag{39}$$

The proof is given in Section 5.8.

3.10 An Abstraction Relationship between Conceptors

The existence of (almost) Boolean operations between conceptors suggests that conceptors may be useful as models of *concepts* (extensive discussion in Section 3.16). In this subsection I add substance to this interpretation by introducing an abstraction relationship between conceptors, which allows one to organize a set of conceptors in an abstraction hierarchy.

In order to equip the set $\mathcal{C}_N$ of $N \times N$ conceptors with an "abstraction" relationship, we need to identify a partial ordering on $\mathcal{C}_N$ which meets our intuitive expectations concerning the structure of "abstraction". A natural candidate is the partial order $\leq$ defined on the set of $N \times N$ real matrices by $X \leq Y$ if $Y - X$ is positive semidefinite. This ordering is often called the *Löwner ordering*. I will interpret and employ the Löwner ordering as an abstraction relation. The key facts which connect this ordering to Boolean operations, and which justify to interpret $\leq$ as a form of logical abstraction, are collected in the following

Proposition 14 *Let $\mathcal{C}_N$ be the set of conceptor matrices of size N. Then the following facts hold.*

1. *An $N \times N$ matrix A is a conceptor matrix if and only if $0 \leq A \leq I_{N \times N}$.*
2. *$0_{N \times N}$ is the global minimal element and $I_{N \times N}$ the global maximal element of $(\mathcal{C}_N, \leq)$.*
3. *$A \leq B$ if and only if $\neg A \geq \neg B$.*
4. *Let $A, B \in \mathcal{C}_N$ and $B \leq A$. Then*

   $$C = \mathbf{P}_{\mathcal{R}(B)} \left(B^\dagger - \mathbf{P}_{\mathcal{R}(B)} A^\dagger \mathbf{P}_{\mathcal{R}(B)} + I\right)^{-1} \mathbf{P}_{\mathcal{R}(B)}$$

   is a conceptor matrix and

   $$B = A \wedge C.$$

5. *Let again $A, B \in \mathcal{C}_N$ and $A \leq B$. Then*

   $$C = I - \mathbf{P}_{\mathcal{I}(B)^\perp} \left((I - B)^\dagger - \mathbf{P}_{\mathcal{I}(B)^\perp} (I - A)^\dagger \mathbf{P}_{\mathcal{I}(B)^\perp} + I\right)^{-1} \mathbf{P}_{\mathcal{I}(B)^\perp}$$

   is a conceptor matrix and

   $$B = A \vee C.$$

6. *If for $A, B, C \in \mathcal{C}_N$ it holds that $A \wedge C = B$, then $B \leq A$.*
7. *If for $A, B, C \in \mathcal{C}_N$ it holds that $A \vee C = B$, then $A \leq B$.*
8. *For $A \in \mathcal{C}_N$ and $\gamma \geq 1$ it holds that $A \leq \varphi(A, \gamma)$; for $\gamma \leq 1$ it holds that $\varphi(A, \gamma) \leq A$.*
9. *If $A \leq B$, then $\varphi(A, \gamma) \leq \varphi(B, \gamma)$ for $\gamma \in [0, \infty]$.*

The proof is given in Section 5.9. The essence of this proposition can be re-expressed succinctly as follows:

Proposition 15 *For conceptors A, B the following conditions are equivalent:*

1. $A \leq B$.
2. *There exists a conceptor C such that $A \vee C = B$.*
3. *There exists a conceptor C such that $A = B \wedge C$.*

Thus, there is an equivalence between "going upwards" in the $\leq$ ordering on the one hand, and merging conceptors by OR on the other hand. In standard logic-based knowledge representation formalisms, a *concept* (or *class*) B is defined to be more *abstract* than some other concept/class A exactly if there is some concept/class C such that $A \vee C = B$. This motivates me to interpret $\leq$ as an *abstraction* ordering on $\mathcal{C}_N$.

3.11 Example: Memory Management in RNNs

In this subsection I demonstrate the usefulness of Boolean operations by introducing a memory management scheme for RNNs. I will show how it is possible

1. to store patterns in an RNN *incrementally*: if patterns $p^1, \ldots, p^k$ have already been stored, a new pattern p^{k+1} can be stored in addition without interfering with the previously stored patterns, and without having to know them;
2. to maintain a measure of the *remaining memory capacity* of the RNN which indicates how many more patterns can still be stored;
3. to *exploit redundancies*: if the new pattern is similar in a certain sense to already stored ones, loading it consumes less memory capacity than when the new pattern is dissimilar to the already stored ones.

Recall that in the original pattern storing procedure, the initial random weight matrix W^* is recomputed to obtain the weight matrix W of the loaded reservoir, such that

$$x^j(n+1) = \tanh(W^* \, x^j(n) + W^{\text{in}} \, p^j(n+1) + b) \approx \tanh(W \, x^j(n) + b),$$

where $p^j(n)$ is the j-th pattern signal and $x^j(n)$ is the reservoir state signal obtained when the reservoir is driven by the j-th pattern. For a transparent memory management, it is more convenient to keep the original W^* and record the weight changes into an *input simulation matrix* D, such that

$$x^j(n+1) = \tanh(W^* \, x^j(n) + W^{\text{in}} \, p^j(n+1)) \approx \tanh(W^* \, x^j(n) + D \, x^j(n)). \quad (40)$$

In a non-incremental batch training mode, D would be computed by regularized linear regression to minimize the following squared error:

$$D = \text{argmin}_{\tilde{D}} \sum_{j=1,\ldots,K} \sum_{n=1,\ldots,L} \|W^{\text{in}}\, p^j(n) - \tilde{D}\, x^j(n-1)\|^2, \tag{41}$$

where K is the number of patterns and L is the length of training samples (after subtracting an initial washout period). Trained in this way, the sum $W^* + D$ would be essentially identical (up to differences due to using another regularization scheme) to the weight W matrix obtained in the original storing procedure. In fact, the performance of loading a reservoir with patterns via an input simulation matrix D as in (40) is indistinguishable from what is obtained in the original procedure (not reported).

The present objective is to find a way to compute D incrementally, leading to a sequence D^j $(j = 1, \ldots, K)$ of input simulation matrices such that the following conditions are satisfied:

1. When the j-th input simulation matrix D^j is used in conjunction with a conceptor C^i associated with an already stored pattern p^i (i.e., $i \leq j$), the autonomous dynamics

   $$x(n+1) = C^i\, \tanh(W^*\, x(n) + D^j\, x(n) + b) \tag{42}$$

   re-generates the i-th pattern.

2. In order to compute D^{j+1}, one must not use explicit knowledge of the already stored patterns or their conceptors, and one only needs to drive the network a single time with the new pattern p^{j+1} in order to obtain the requisite training data.

3. If two training patterns $p^i = p^j$ are identical (where $i > j$), $D^j = D^{j-1}$ is obtained. The network already has stored p^i and does not need to change in order to accomodate to this pattern when it is presented a second time.

We will see that, as a side effect, the third condition also allows the network to exploit redundancies when p^j is similar but not identical to an earlier p^i; the additional memory consumption is reduced in such cases.

The key idea is to keep track of what parts of the reservoir memory space have already been claimed by already stored patterns, or conversely, which parts of the memory space are still freely disposable. Each pattern p^j is associated with a conceptor C^j. The "used-up" memory space after having stored patterns $p^1, \ldots, p^j$ will be characterized by the disjunction $A^j = \bigvee\{C^1, \ldots, C^j\}$, and the "still disposable" space by its complement $\neg A^j$.

Let a raw reservoir with an initial weight matrix W^* be given, as well as training pattern timeseries $p^j(n)$ where $j = 1, \ldots, K$ and $n = 1, \ldots, L$. Then the incremental storing algorithm proceeds as follows.

Initialization (no pattern yet stored): $D^0 = A^0 = 0_{N\times N}$. Choose an aperture α to be used for all patterns.

Incremental storage of patterns: For $j = 1, \ldots, K$ do:

1. Drive reservoir with pattern p^j for L timesteps using state updates

$$x^j(n+1) = \tanh(W^* \, x^j(n) + W^{\text{in}} p^j(n+1) + b)$$

and collect the states $x^j(1), \ldots, x^j(L-1)$ into a $N \times (L-1)$ sized state collection matrix X^j. Put $R^j = X^j \, (X^j)' \, / \, (L-1)$. Likewise, collect the patterns $p^j(2), \ldots, p^j(L)$ into a row vector P^j of length $L-1$. (Here, like elsewhere in this report, I tacitly assume that before one collects data, the network state has been purged by driving through an initial washout period).

2. Compute a conceptor for this pattern by $C^j = R^j \, (R^j + \alpha^{-2} \, I)^{-1}$.
3. Compute an $N \times N$ matrix D^j_{inc} (subsequently to be added as an increment to D^{j-1}, yielding $D^j = D^{j-1} + D^j_{\text{inc}}$) by putting
   (a) $F^{j-1} = \neg A^{j-1}$ (*comment: this conceptor characterizes the "still disposable" memory space for learning* p^j),
   (b) $T = W^{\text{in}} \, P^j - D^{j-1} \, X^j$ (*comment: this* $N \times (L-1)$ *sized matrix contains the targets for a linear regression to compute* D_{inc}),
   (c) $S = F^{j-1} \, X^j$ (*comment: this* $N \times (L-1)$ *sized matrix contains the arguments for the linear regression*),
   (d) $D^j_{\text{inc}} = \big((SS'/(L-1) + \alpha^{-2} I)^\dagger \, ST'/(L-1)\big)'$ (*comment: carry out the regression, regularized by* α^{-2}).
4. Update D: $D^j = D^{j-1} + D^j_{\text{inc}}$.
5. Update A: $A^j = A^{j-1} \vee C^j$.

Here is an intuitive explanation of the core ideas in the update step $(j-1) \to j$ in this algorithm:

1. *Step 3(a):* $A^{j-1} = \bigvee\{C^1, \ldots, C^{j-1}\}$ keeps track of the subspace directions that have already been "consumed" by the patterns already stored, and its complement $F^{j-1} = \neg A^{j-1}$ represents the "still unused" directions of reservoir state space.

2. *Step 3(b):* The regression targets for D^j_{inc} are given by the state contributions of the driving input $(W^{\text{in}} \, P^j)$, minus what the already installed input simulation matrix already contributes to predicting the input effects from the previous network state $(-D^{j-1} X^j)$. Setting the regression target in this way gives rise to the desired exploitation of redundancies. If pattern p^j had been learnt before (i.e., a copy of it was already presented earlier), $T = 0$ will result in this step, and hence, $D^j_{\text{inc}} = 0$.

3. *Step 3(c):* The arguments for the linear regression are confined to the projection $F^{j-1}\,X^j$ of the p^j-driven network states X^j on the still unused reservoir directions F^{j-1}. In this way, $D^j_{\rm inc}$ is decoupled from the already installed D^{j-1}: $D^j_{\rm inc}$ exploits for generating its output state only such information that was not used by D^{j-1}.

4. *Step 3(d):* This is the well-known Tychonov-regularized Wiener-Hopf formula for computing a linear regression of targets T on arguments S, also known as "ridge regression" [28, 106]. Setting the Tychonov regularizer to the inverse squared aperture α^{-2} is not accidental. It results from the mathematically identical roles of Tychonov regularizers and apertures.

The sketched algorithm provides a basic format. It can be easily extended/refined, for instance by using different apertures for different patterns, or multidimensional input.

It is interesting to note that it is intrinsically impossible to *unlearn* patterns selectively and "decrementally". Assume that patterns $p^1, \ldots, p^K$ have been trained, resulting in D^K. Assume that one wishes to unlearn again p^K. As a result of this unlearning one would want to obtain D^{K-1}. Thus one would have to compute $D^K_{\rm inc}$ from D^K, A^K and p^K (that is, from D^K and C^K), in order to recover $D^{K-1} = D^K - D^K_{\rm inc}$. However, the way to identify $D^K_{\rm inc}$ from D^K, A^K and C^K is barred because of the redundancy exploitation inherent in step 3(b). Given only D^K, and not knowing the patterns $p^1, \ldots, p^{K-1}$ which must be preserved, there is no way to identify which directions of reservoir space must be preserved to preserve those other patterns. The best one can do is to put $\tilde{A}^{K-1} = A^K - C^K = A^K \wedge \neg C^K$ and re-run step 3 using $\tilde{A}^{K-1}$ instead of A^{K-1} and putting $T = W^{\rm in}\,P^j$ in step 3(b). This leads to a version $\tilde{D}^K_{\rm inc}$ which coincides with the true $D^K_{\rm inc}$ only if there was no directional overlap between C^K and the other C^j, i.e. if $D^{K-1}X^K = 0$ in the original incremental learning procedure. To the extent that p^K shared state directions with the other patterns, i.e. to the extent that there was redundancy, unlearning p^K will degrade or destroy patterns that share state directions with p^K.

The incremental pattern learning method offers the commodity to measure how much "memory space" has already been used after the first j patterns have been stored. This quantity is the quota $q(A^j)$. When it approaches 1, the reservoir is "full" and an attempt to store another pattern will fail because the F matrix from step 3(a) will be close to zero.

Two demonstrations illustrate various aspects of the incremental storing procedure. Both demonstrations used an $N = 100$ sized reservoir, and $K = 16$ patterns were stored. In the first demonstration, the patterns (sines or random patterns) were periodic with integer period lengths ranging between 3 and 15. In the second demonstration, the patterns were sinewaves with irrational period lengths chosen between 4 and 20. Details are documented in Section 4.3. Figures 21 and 22 plot characteristic impressions from these demonstrations.

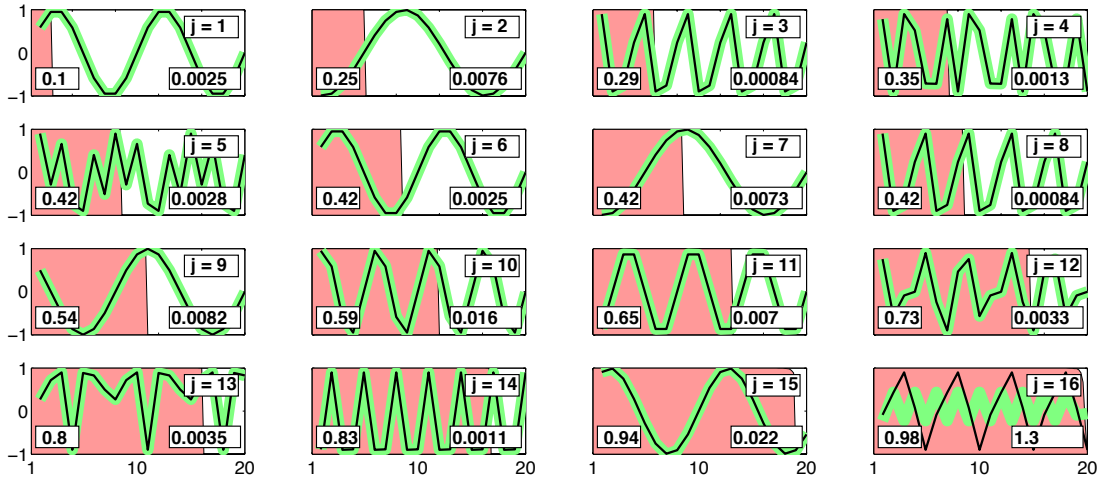

Figure 21: Incremental storing, first demonstration (figure repeated from Section 1 for convenience). 13 patterns with integer period lengths ranging between 3 and 15 were stored. Patterns were sinewaves with integer periods or random. Patterns $j = 6, 7, 8$ are identical to $j = 1, 2, 3$. Each panel shows a 20-timestep sample of the correct training pattern p^j (black line) overlaid on its reproduction (green line). The memory fraction used up until pattern j is indicated by the panel fraction filled in red; the quota value is printed in the left bottom corner of each panel. The red areas in each panel in fact show the singular value spectrum of A^j (100 values, x scale not shown). The NRMSE is inserted in the bottom right corners of the panels. For detail see text.

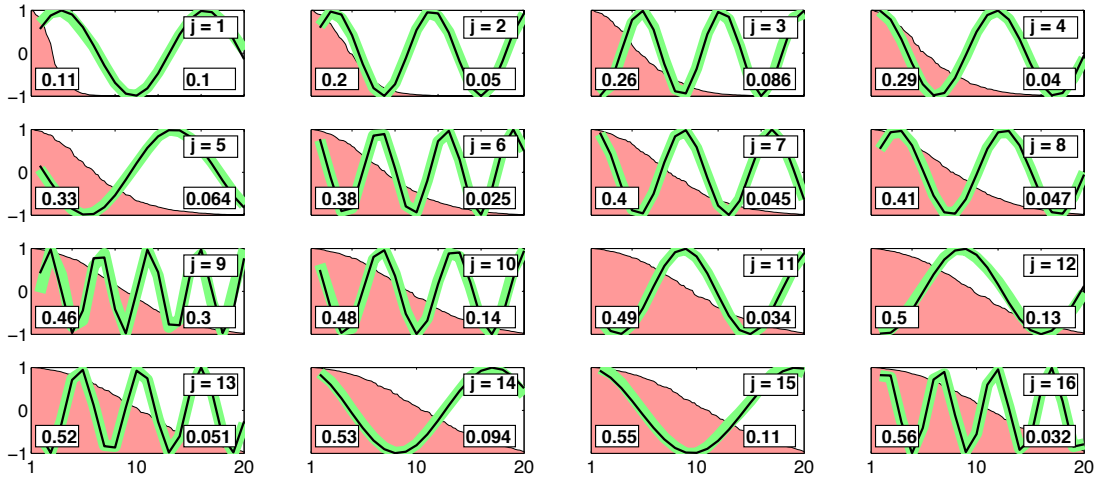

Figure 22: Incremental storing, second demonstration. 16 sinewave patterns with irrational periods ranging between 4 and 20 were used. Plot layout is the same as in Figure 21. For detail see text.

Comments on the demonstrations:

Demonstration 1. When a reservoir is driven with a signal that has an integer period, the reservoir states (after an initial washout time) will entrain to this period, i.e. every neuron likewise will exhibit an integer-periodic activation signal. Thus, if the period length of driver j is m, the correlation matrix R^j as well as the conceptor C^j will be matrices of rank m. An aperture $\alpha = 1000$ was used in this demonstration. The large size of this aperture and the fact that R^j has rank m leads to a conceptor C^j which comes close to a projector matrix, i.e. it has m singular values that are close to one and $N - m$ zero singular values. Furthermore, if a new pattern p^{j+1} is presented, the periodic reservoir state vectors arising from it will generically be linearly independent of all state vectors that arose from earlier drivers. Both effects together (almost projector C^j and linear independence of nonzero principal directions of these C^j) imply that the sequence $A^1, \ldots, A^K$ will essentially be a sequence of projectors, where $\mathcal{R}(A^{j+1})$ will comprise m more dimensions than $\mathcal{R}(A^j)$ (where m is the period length of p^{j+1}). This becomes clearly apparent in Figure 21: the area under the singular value plot of A^j has an almost rectangular shape, and the increments from one plot to the next match the periods of the respective drivers, except for the last pattern, where the network capacity is almost exhausted.

Patterns $j = 6, 7, 8$ were identical to $j = 1, 2, 3$. As a consequence, when the storage procedure is run for $j = 6, 7, 8$, A^j remains essentially unchanged – no further memory space is allocated.

When the network's capacity is almost exhausted in the sense that the quota $q(A^j)$ approaches 1, storing another pattern becomes inaccurate. In this demo, this happens for that last pattern $j = 16$ (see Figure 21).

Demonstration 2. When the driver has an irrational period length, the excited reservoir states will span the available reservoir space $\mathbb{R}^N$. Each R^j will have only nonzero singular values, albeit of rapidly decreasing magnitude (these tails are so small in magnitude that they are not visible in the first few plots of A^j in Figure 22). The fact that each driving pattern excites the reservoir in all directions leads to the "reverse sigmoid" kind of shapes of the singular values of the A^j visible in Figure 22.

The sinewave drivers p^j were presented in an order with randomly shuffled period lengths. A redundancy exploitation effect becomes apparent: while for the first 8 patterns altogether a quota $q(A^8) = 0.41$ was allocated, the next 8 patterns only needed an additional quota of $q(A^{16}) - q(A^8) = 0.15$. Stated in suggestive terms, at later stages of the storing sequence the network had already learnt how to oscillate in sinewaves in general, and only needed to learn in addition how to oscillate at the particular newly presented frequencies. An aperture of size $\alpha = 3$ was used in the second demonstration.

The two demonstrations showed that when n-point periodic patterns are stored, each new pattern essentially claims a new n-dimensional subspace. In contrast, each of the irrational-periodic sines affected all of the available reservoir dimensions, albeit to different degrees. This leads to problems when one tries to store n-periodic patterns *after* irrational-periodic patterns have already been stored. The latter already occupy all available reservoir state directions, and the new n-periodic storage candidates cannot find completely "free" directions. This leads to poor retrieval qualities for n-periodic patterns stored on top of irrational-periodic ones (not shown). The other order – storing irrational-periodic patterns on top of n-periodic ones – is not problematic. This problem can be mitigated with the aid of *autoconceptors*, which are developed in subsequent sections. They typically lead to almost hard conceptors with a majority of singular values being zero, and thus – like n-point periodic patterns – only claim low-dimensional subspaces, leaving unoccupied "dimension space" for loading further patterns.

An architecture variant. When one numerically computes the rank of the input simulation matrix D obtained after storing all patterns, one will find that it has rank 1. This can be readily explained as follows. The desired functionality of D is to replace the N-dimensional signals $W^{\text{in}}\, p^j(n+1)$ by $D\, x(n)$ (in a condition where the network state x is governed by the conceptor C^j). The signal $W^{\text{in}}\, p^j(n)$ has a rank-1 autocorrelation matrix $E[W^{\text{in}}\, p^j(n)(W^{\text{in}}\, p^j(n))']$. Therefore, also $E[Dx(n)(Dx(n))']$ should have unit rank. Since the reservoir states $x(n)$ will span all of $\mathbb{R}^N$ across the different patterns j, a unit rank of $E[Dx(n)(Dx(n))']$ implies a unit rank of D. In fact, the columns of D turn out to be scaled versions of W^{in}, i.e. $D = W^{\text{in}} d$ for some N-dimensional *row* vector d.

Furthermore, from $W^{\text{in}}\, d\, x(n) = Dx(n) \approx W^{\text{in}}\, p^j(n+1)$ we infer $d\, x(n) \approx p^j(n+1)$: projections of network states on d *predict* the next input.

This suggests an alternative way to design and train pattern-storing RNNs. The signal $d\, x(n)$ is assigned to a newly introduced single neuron π. The system equation for the new design in external driving conditions remains unchanged:

$$x(n+1) = \tanh(W^* x(n) + W^{\text{in}} p^j(n+1) + b).$$

However, the autonomous dynamics (42) is replaced by

$$\begin{aligned} \pi(n+1) &= d\, x(n) && (43)\\ x(n+1) &= C^j\, \tanh(W^* x(n) + W^{\text{in}} \pi(n+1) + b), && (44)\\ & && (45) \end{aligned}$$

or equivalently

$$x(n+1) = C^j\, \tanh(W^* x(n) + W^{\text{in}}\, d\, x(n) + b). \qquad (46)$$

The original readout $y(n) = W^{\text{out}}\, x(n)$ becomes superfluous, since $\pi(n) = y(n)$. Figure 23 is a diagram of the alternative architecture. Note that d is the only trainable item in this architecture.

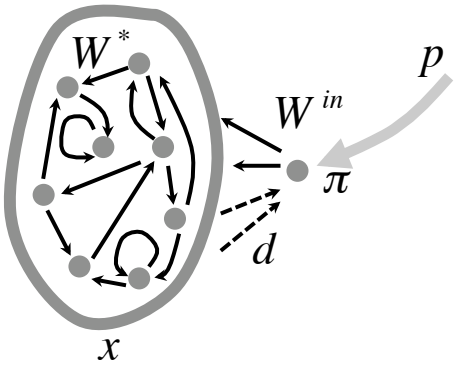

Figure 23: Alternative architecture for storing patterns. A multi-functional neuron π serves as input unit in external driving conditions: its value is then forced by the external driver p. In autonomous runs when p is absent, the value of π is determined by its readout weights d from the reservoir. Non-trainable connections are drawn as solid arrows, trainable ones are broken. The bias vector and the action of conceptors are omitted in this diagram.

The incremental storing procedure is adapted to the alternative architecture as follows.

Initialization (no pattern yet stored): $A^0 = 0_{N\times N}, d^0 = 0_{1\times N}$. Choose an aperture α to be used for all patterns.

Incremental storage of patterns: For $j = 1, \ldots, K$ do:

1. *(unchanged from original procedure)*
2. *(unchanged)*
3. Compute an $1 \times N$ vector d^j_{inc} by putting
   (a) $F^{j-1} = \neg A^{j-1}$ *(unchanged)*,
   (b) $t = P^j - (d^{j-1})\, X^j$ *(the essential change)*,
   (c) $S = F^{j-1}\, X^j$ *(unchanged)*,
   (d) $d^j_{\text{inc}} = \big((SS'/(L-1) + \alpha^{-2} I)^\dagger\, St'/(L-1)\big)'$.
4. Update d: $d^j = d^{j-1} + d^j_{\text{inc}}$.
5. Update A: $A^j = A^{j-1} \vee C^j$ *(unchanged)*.

For deterministic patterns (as they were used in the two demonstrations above), the alternative procedure should, and does, behave identically to the original one (not shown). For stochastic patterns it can be expected to be statistically more efficient (needing less training data for achieving same accuracy) since it exploits the valuable structural bias of knowing beforehand that D should have unit rank and have scaled versions of W^{in} as columns (remains to be explored).

Our demonstrations used 1-dimensional drivers. For m-dimensional drivers, the alternative architecture can be designed in an obvious way using m additional neurons π_ν.

The alternative architecture has maximal representational efficiency in the following sense. For storing integer-periodic signals p^j, where the sum of periods of the stored signals approaches the network size N (as in demonstration 1), only N parameters (namely, d) have to be trained. One may object that one also has to store the conceptors C^j in order to retrieve the patterns, i.e. one has to learn and store also the large number of parameters contained in the conceptors. We will however see in Section 3.13.4 that conceptors can be re-built on the fly in retrieval situations and need not be stored.

The alternative architecture also lends itself to storing arbitrarily large numbers of patterns on the basis of a single reservoir. If the used memory quota $q(A^j)$ approaches 1 and the above storing procedure starts stalling, one can add another π neuron and continue storing patterns using it. This however necessitates an additional switching mechanism to select between different available such π neurons in training and retrieval (not yet explored).

3.12 Example: Dynamical Pattern Recognition

In this subsection I present another demonstration of the usefulness of Boolean operations on conceptor matrices. I describe a training scheme for a pattern recognition system which reaches (or surpasses) the classification test performance of state-of-the-art recognizers on a widely used benchmark task. Most high-performing existing classifiers are trained in discriminative training schemes. Discriminative classifier training exploits the contrasting differences between the pattern classes. This implies that if the repertoire of to-be-distinguished patterns becomes extended by a new pattern, the classifier has to be re-trained on the entire dataset, re-visiting training data from the previous repertoire. In contrast, the system that I present is trained in a "pattern-local" scheme which admits an incremental extension of the recognizer if new pattern types were to be included in its repertoire. Furthermore, the classifier can be improved in its exploitation phase by incrementally incorporating novel information contained in a newly incoming test pattern. The key to this local-incremental classifier training is agin Boolean operations on conceptors.

Unlike in the rest of this report, where I restrict the presentation to stationary and potentially infinite-duration signals, the patterns here are nonstationary and of short duration. This subsection thus also serves as a demonstration how conceptors function with short nonstationary patterns.

I use is the *Japanese Vowels* benchmark dataset. It has been donated by [60] and is publicly available at the UCI Knowledge Discovery in Databases Archive (`http://kdd.ics.uci.edu/`). This dataset has been used in dozens of articles in machine learning as a reference demonstration and thus provides a quick first

orientation about the positioning of a new classification learning method. The dataset consists of 640 recordings of utterances of two successive Japanese vowels /ae/ from nine male speakers. It is grouped in a training set (30 recordings from each of the speakers = 270 samples) and a test set (370 further recordings, with different numbers of recordings per speaker). Each sample utterance is given in the form of a 12-dimensional timeseries made from the 12 LPC cepstrum coefficients. The durations of these recordings range between 7 and 29 sampling timesteps. The task is to classify the speakers in the test data, using the training data to learn a classifier. Figure 24 (top row) gives an impression of the original data.

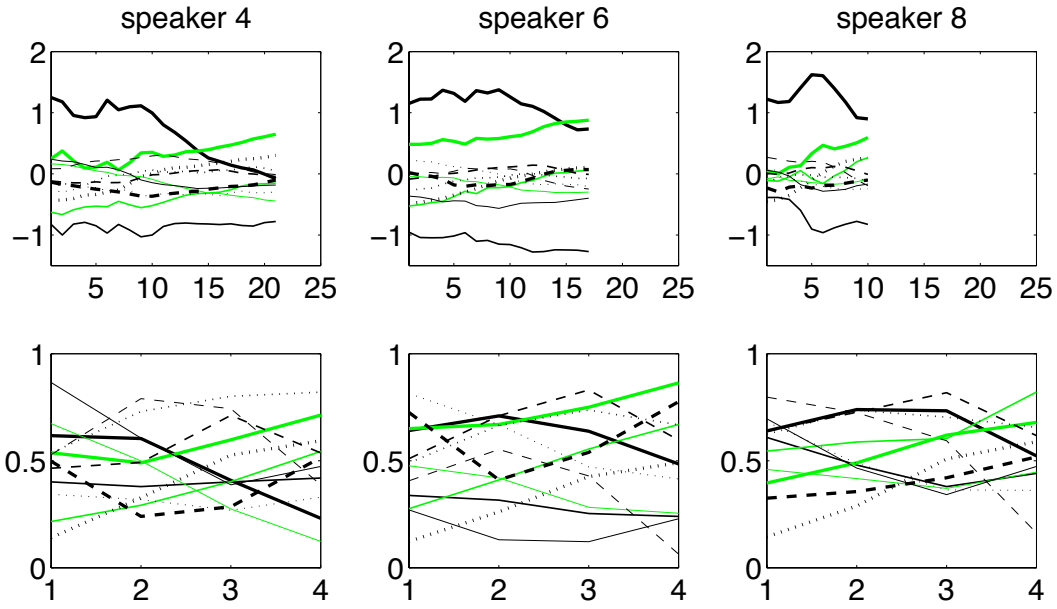

Figure 24: Three exemplary utterance samples from Japanese Vowels dataset. Plots show values of twelve signal channels against discrete timesteps. Top row: raw data as provided in benchmark repository, bottom row: standardized format after preprocessing.

I preprocessed the raw data into a standardized format by (1) shift-scaling each of the twelve channels such that per channel, the minimum/maximum value across all training samples was 0/1; (2) interpolating each channel trace in each sample by a cubic polynomial; (3) subsampling these on four equidistant support points. The same transformations were applied to the test data. Figure 24 (bottom row) illustrates the normalized data format.

The results reported in the literature for this benchmark typically reach an error rate (percentage of misclassifications on the test set) of about 5 – 10 test errors (for instance, [91, 97, 79] report from 5 – 12 test misclassifications, all using specialized versions of temporal support vector machines). The best result that I am aware of outside my own earlier attempts [57] is reported by [15] who reaches about 4 errors, using refined hidden Markov models in a non-discriminative train-

ing scheme. It is however possible to reach zero errors, albeit with an extraordinary effort: in own work [57] this was robustly achieved by combining the votes of 1,000 RNNs which were each independently trained in a discriminative scheme.

Here I present a "pocket-size" conceptor-based classification learning scheme which can be outlined as follows:

1. A single, small ($N = 10$ units) random reservoir network is initially created.

2. This reservoir is driven, in nine independent sessions, with the 30 preprocessed training samples of each speaker j ($j = 1, \ldots, 9$), and a conceptor C_j^+ is created from the network response (no "loading" of patterns; the reservoir remains unchanged throughout).

3. In exploitation, a preprocessed sample s from the test set is fed to the reservoir and the induced reservoir states $x(n)$ are recorded and transformed into a single vector z. For each conceptor then the *positive evidence* quantity $z' C_j^+ z$ is computed. This leads to a classification by deciding for $j = \text{argmax}_i \, z' C_i^+ z$ as the speaker of s. The idea behind this procedure is that if the reservoir is driven by a signal from speaker j, the resulting response z signal will be located in a linear subspace of the (transformed, see below) reservoir state space whose overlap with the ellipsoids given by the C_i^+ is largest for $i = j$.

4. In order to further improve the classification quality, for each speaker j also a conceptor $C_j^- = \neg \bigvee \{C_1^+, \ldots, C_{j-1}^+, C_{j+1}^+, \ldots, C_9^+\}$ is computed. This conceptor can be understood as representing the event "not any of the other speakers". This leads to a *negative evidence* quantity $z' C_j^- z$ which can likewise be used as a basis for classification.

5. By adding the positive and negative evidences, a *combined evidence* is obtained which can be paraphrased as "this test sample seems to be from speaker j and seems not to be from any of the others".

In more detail, the procedure was implemented as follows. A 10-unit reservoir system with 12 input units and a constant bias term with the update equation

$$x(n+1) = \tanh(W\, x(n) + W^{\text{in}} s(n) + b) \tag{47}$$

was created by randomly creating the 10×10 reservoir weight matrix W, the 12×10 input weight matrix W^{in} and the bias vector b (full specification in Section 4.5). Furthermore, a random starting state x_{start}, to be used in every run in training and testing, was created. Then, for each speaker j, the conceptor C_j^+ was learnt from the 30 preprocessed training samples $s_j^k(n)$ (where $j = 1, \ldots, 9$; $k = 1, \ldots, 30; n = 1, \ldots, 4$) of this speaker, as follows:

1. For each training sample s_j^k ($k = 1, \ldots, 30$) of this speaker, the system (47) was run with this input, starting from $x(0) = x_{\text{start}}$, yielding four network states $x(1), \ldots, x(4)$. These states were concatenated with each other and with the driver input into a $4 \cdot (10 + 12) = 88$ dimensional vector $z_j^k = [x(1); s_j^k(1); \ldots; x(4); s_j^k(4)]$. This vector contains the entire network response to the input s_j^k and the input itself.

2. The 30 z_j^k were assembled as columns into a 88×30 matrix Z from which a correlation matrix $R_j = ZZ'/30$ was obtained. A preliminary conceptor $\tilde{C}_j^+ = R_j(R_j + I)^{-1}$ was computed from R_j (preliminary because in a later step the aperture is optimized). Note that $\tilde{C}_j^+$ has size 88×88.

After all "positive evidence" conceptors $\tilde{C}_j^+$ had been created, preliminary "negative evidence" conceptors $\tilde{C}_j^-$ were computed as

$$\tilde{C}_j^- = \neg \bigvee \{\tilde{C}_1^+, \ldots, \tilde{C}_{j-1}^+, \tilde{C}_{j+1}^+, \ldots, \tilde{C}_9^+\}. \tag{48}$$

An important factor for good classification performance is to find optimal apertures for the conceptors, that is, to find aperture adaptation factors γ^+, γ^- such that final conceptors $C_j^+ = \varphi(\tilde{C}_j^+, \gamma^+), C_j^- = \varphi(\tilde{C}_j^-, \gamma^-)$ function well for classification. A common practice in machine learning would be to optimize γ by cross-validation on the training data. This, however, is expensive, and more crucially, it would defy the purpose to design a learning procedure which can be incrementally extended by novel pattern classes without having to re-inspect all training data. Instead of cross-validation I used the ∇ criterion described in Section 3.8.4 to find a good aperture. Figure 25 shows how this criterion varies with γ for an exemplary case of a $\varphi(\tilde{C}_j^+, \gamma^+)$ sweep. For each of the nine $\tilde{C}_j^+$, the value $\tilde{\gamma}_j^+$ which maximized ∇ was numerically computed, and the mean of these nine values was taken as the common γ^+ to get the nine $C_j^+ = \varphi(\tilde{C}_j^+, \gamma^+)$. A similar procedure was carried out to arrive at $C_j^- = \varphi(\tilde{C}_j^-, \gamma^-)$.

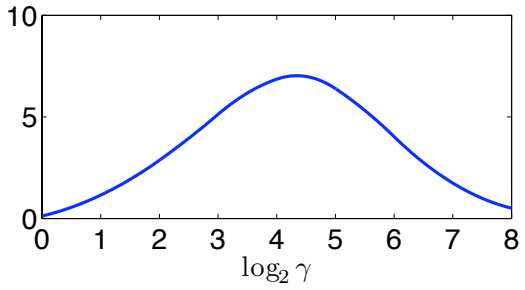

Figure 25: The criterion ∇ from an exemplary conceptor plotted against the log 2 of candidate aperture adaptations γ.

The conceptors C_j^+, C_j^- were then used for classification as follows. Assume z is an 88-dimensional combined states-and-input vector as described above, obtained

from driving the reservoir with a preprocessed test sample. Three kinds of classification hypotheses were computed, the first only based on C_j^+, the second based on C_j^-, and one based on a combination of both. Each classification hypothesis is a 9-dimensional vector with "evidences" for the nine speakers. Call these evidence vectors h^+, h^-, h^{+-} for the three kinds of classifications. The first of these was computed by setting $\tilde{h}^+(j) = z' C_j^+ z$, then normalizing $\tilde{h}^+$ to a range of $[0, 1]$ to obtain h^+. Similarly h^- was obtained from using $z' C_j^- z$, and $h^{+-} = (h^+ + h^-)/2$ was simply the mean of the two former. Each hypothesis vector leads to a classification decision by opting for the speaker j corresponding to the largest component in the hypothesis vector.

This classification procedure was carried out for all of the 370 test cases, giving 370 hypothesis vectors of each of the three kinds. Figure 26 gives an impression.

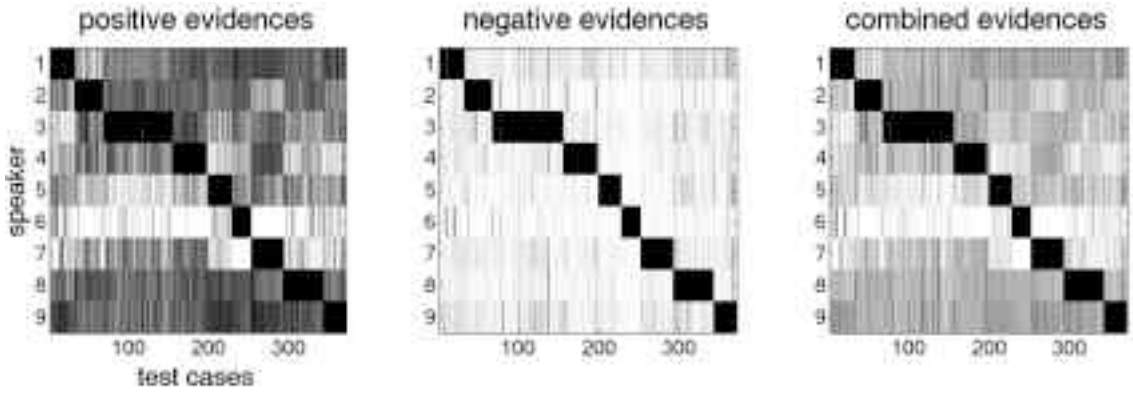

Figure 26: Collected evidence vectors h^+, h^-, h^{+-} obtained in a classification learning experiment. Grayscale coding: white = 0, black = 1. Each panel shows 370 evidence vectors. The (mostly) black segments along the diagonal correspond to the correct classifications (test samples were sorted by speaker). For explanation see text.

Results: The outlined classification experiment was repeated 50 times with random new reservoirs. On average across the 50 trials, the optimal apertures γ^+ / γ^- were found as 25.0 / 27.0 (standard deviations 0.48 / 0.75). The number of misclassifications for the three types of classification (positive, negative, combined evidence) were 8.5 / 5.9 / 4.9 (standard deviations 1.0 / 0.91 / 0.85). The training errors for the combined classification (obtained from applying the classification procedure on the training samples) was zero in all 50 trials. For comparison, a carefully regularized linear classifier based on the same z vectors (detail in Section 4.5) reached 5.1 misclassifications across the 50 trials.

While these results are at the level of state-of-the-art classifiers on this benchmark, this basic procedure can be refined, yielding a significant improvement. The idea is to compute the evidence for speaker j based on a conceptor $\bar{C}_j^+$ which itself is based on the assumption that the test sample s belongs to the class j, that is, the

computed evidence should reflect a quantity "if s belonged to class j, what evidence can we collect under this assumption?". Recall that C_j^+ is obtained from the 30 training samples through $C_j^+ = R\,(R + (\gamma^+)^{-2} I)^{-1}$, where $R = ZZ'/30$ is the correlation matrix of the 30 training coding vectors belonging to speaker j. Now add the test vector z to Z, obtaining $\bar{Z} = [Zz], \bar{R} = \bar{Z}\bar{Z}'/31, \bar{C}_j^+ = \bar{R}(\bar{R} + (\gamma^+)^{-2} I)^{-1}$, and use $\bar{C}_j^+$ in the procedure outlined above instead of C_j^+. Note that, in application scenarios where the original training data Z are no longer available at test time, $\bar{C}_j^+$ can be directly computed from C_j^+ and z through the model update formulas (36) or (37). The negative evidence conceptor is accordingly obtained by $\bar{C}_j^- = \neg \bigvee\{\bar{C}_1^+, \ldots, \bar{C}_{j-1}^+, \bar{C}_{j+1}^+, \ldots, \bar{C}_9^+\}$.

Results of refined classification procedure: Averaged over 50 learn-test trials with independently sampled reservoir weights, the number of misclassifications for the three types of classification (positive, negative, combined evidence) were 8.4 / 5.9 / 3.4 (standard deviations 0.99 / 0.93 / 0.61). The training misclassification errors for the combined classification was zero in all 50 trials.

The detection of good apertures through the ∇ criterion worked well. A manual grid search through candidate apertures found that a minimum test misclassification rate of 3.0 (average over the 50 trials) from the combined classificator was obtained with an aperture $\alpha^+ = 20, \alpha^- = 24$ for both the positive and negative conceptors. The automated aperture detection yielded apertures $\alpha^+ = 25, \alpha^- = 27$ and a (combined classificator) misclassification rate of 3.4, close to the optimum.

Discussion. The following observations are worth noting.

Method also applies to static pattern classification. In the presented classification method, temporal input samples s (short preprocessed nonstationary timeseries) were transformed into static coding vectors z as a basis for constructing conceptors. These z contained the original input signal s plus the state response from a small reservoir driven by s. The reservoir was only used to augment s by some random nonlinear interaction terms between the entries in s. Conceptors were created and used in classification without referring back to the reservoir dynamics. This shows that conceptors can also be useful in *static* pattern classification.

Extensibility. A classification model consisting of learnt conceptors C_j^+, C_j^- for k classes can be easily *extended by new classes*, because the computations needed for new C_{k+1}^+, C_{k+1}^- only require positive training samples of the new class. Similarly, an existing model C_j^+, C_j^- can be *extended by new training samples* without re-visiting original training data by an application of the model extension formulae (36) or (37). In fact, the refined classification procedure given above can be seen as an ad-hoc conditional model extension by the test sample.

Including an "other" class. Given a learnt classification model C_j^+, C_j^- for k classes it appears straightforward to include an "other" class by including

$C^+_{\text{other}} = \neg \bigvee \{C^+_1, \ldots, C^+_k\}$ and recomputing the negative evidence conceptors from the set $\{C^+_1, \ldots, C^+_k, C^+_{\text{other}}\}$ via (48). I have not tried this out yet.

Discriminative nature of combined classification. The classification of the combined type, paraphrased above as "sample seems to be from class j and seems not to be from any of the others", combines information from all classes into an evidence vote for a candidate class j. Generally, in discriminative learning schemes for classifiers, too, contrasting information *between* the classes is exploited. The difference is that in those schemes, these differences are worked in at learning time, whereas in the presented conceptor-based scheme they are evaluated at test time.

Benefits of Boolean operations. The three aforementioned points – extensibility, "other" class, discriminative classification – all hinge on the availability of the NOT and OR operations, in particular, on the associativity of the latter.

Computational efficiency. The computational steps involved in learning and applying conceptors are constructive. No iterative optimization steps are involved (except that standard implementations of matrix inversion are iterative). This leads to short computation times. Learning conceptors from the 270 preprocessed data samples, including determining good apertures, took 650 ms and classifying a test sample took 0.7 ms for the basic and 64 ms for the refined procedure (on a dual-core 2GHz Macintosh notebook computer, using Matlab).

Competitiveness. The test misclassification rate of 3.4 is slightly better than the best rate of about 4 that I am aware of in the literature outside own work [57]. Given that the zero error performance in [57] was achieved with an exceptionally expensive model (combining 1,000 independently sampled classifiers), which furthermore is trained in a discriminative setup and thus is not extensible, the attained performance level, the computational efficiency, and the extensibility of the conceptor-base model render it a competitive alternative to existing classification learning methods. It remains to be seen though how it performs on other datasets.

Regularization by aperture adaptation? In supervised classification learning tasks, it is generally important to regularize models to find the best balance between overfitting and under-exploiting training data. It appears that the role of regularization is here played by the aperture adaptation, though a theoretical analysis remains to be done.

Early stage of research. The proposed classifier learning scheme was based on numerous ad-hoc design decisions, and quite different ways to exploit conceptors for classification are easily envisioned. Thus, in sum, the presented

study should be regarded as no more than a first demonstration of the basic usefulness of conceptors for classification tasks.

3.13 Autoconceptors

3.13.1 Motivation and Overview

In the preceding sections I have defined conceptors as transforms $C = R(R + \alpha^{-2}I)^{-1}$ of reservoir state correlation matrices R. In order to obtain some conceptor C^j which captures a driving pattern p^j, the network was driven by p^j via $x(n+1) = \tanh(W^* x(n) + W^{\text{in}} p^j(n+1) + b)$, the obtained reservoir states were used to compute R^j, from which C^j was computed. The conceptor C^j could then later be exploited via the conceptor-constrained update rule $x(n+1) = C^j \tanh(Wx(n) + b)$ or its variant $x(n+1) = C^j \tanh(W^* x(n) + D x(n) + b)$.

This way of using conceptors, however, requires that the conceptor matrices C^j are computed at learning time (when the original drivers are active), and they have to be *stored* for later usage. Such a procedure is useful and feasible in engineering or machine learning applications, where the conceptors C^j may be written to file for later use. It is also adequate for theoretical investigations of reservoir dynamics, and logical analyses of relationships between reservoir dynamics induced by different drivers, or constrained by different conceptors.

However, storing conceptor matrices is entirely implausible from a perspective of neuroscience. A conceptor matrix has the same size as the original reservoir weight matrix, that is, it is as large an entire network (up to a saving factor of one half due to the symmetry of conceptor matrices). It is hard to envision plausible models for computational neuroscience where learning a new pattern by some RNN essentially would amount to creating an entire new network.

This motivates to look for ways of how conceptors can be used for constraining reservoir dynamics without the necessity to store conceptors in the first place. The network would have to create conceptors "on the fly" while it is performing some relevant task. Specifically, we are interested in tasks or functionalities which are relevant from a computational neuroscience point of view. This objective also motivates to focus on algorithms which are not immediately biologically implausible. In my opinion, this largely excludes computations which explicitly exploit the SVD of a matrix (although it has been tentatively argued that neural networks can perform principal component analysis [78] using biologically observed mechanisms).

In the next subsections I investigate a version of conceptors with associated modes of usage where there is no need to store conceptors and where computations are online adaptive and local in the sense that the information necessary for adapting a synaptic weight is available at the concerned neuron. I will demonstrate the workings of these conceptors and algorithms in two functionalities, (i) content-addressable memory (Section 3.13.3) and (ii) simultaneous de-noising and classification of a signal (Section 3.15).

In this line of modeling, the conceptors are created by the reservoir itself at the time of usage. There is no role for an external engineer or superordinate control algorithm to "plug in" a conceptor. I will speak of *autoconceptors* to distinguish these autonomously network-generated conceptors from the conceptors that are externally stored and externally inserted into the reservoir dynamics. In discussions I will sometimes refer to those "externalistic" conceptors as *alloconceptors*.

Autoconceptors, like alloconceptors, are positive semidefinite matrices with singular values in the unit interval. The semantic relationship to data, aperture operations, and Boolean operations are identical for allo- and autoconceptors. However, the way how autoconceptors are generated is different from alloconceptors, which leads to additional constraints on their algebraic characteristics. The set of autoconceptor matrices is a proper subset of the conceptor matrices in general, as they were defined in Definition 2, i.e. the class of positive semidefinite matrices with singular values ranging in the unit interval. The additional constraints arise from the circumstance that the reservoir states $x(n)$ which shape an autoconceptor C are themselves depending on C.

The treatment of autoconceptors will be structured as follows. I will first introduce the basic equations of autoconceptors and their adaptation dynamics (Section 3.13.2), demonstrate their working in a of content-addressable memory task (Section 3.13.3) and mathematically analyse central properties of the adaptation dynamics (Section 3.13.4). The adaptation dynamics however has non-local aspects which render it biologically implausible. In order to progress toward biologically feasible autoconceptor mechanisms, I will propose neural circuits which implement autoconceptor dynamics in ways that require only local information for synaptic weight changes (Section 3.14).

3.13.2 Basic Equations

The basic system equation for autoconceptor systems is

$$x(n+1) = C(n)\ \tanh(W^*\, x(n) + W^{\mathrm{in}} p(n+1) + b) \tag{49}$$

or variants thereof, like

$$x(n+1) = C(n)\ \tanh(W\, x(n) + b) \tag{50}$$

or

$$x(n+1) = C(n)\ \tanh(W^* x(n) + D x(n) + b), \tag{51}$$

the latter two for the situation after having patterns stored. The important novel element in these equations is that $C(n)$ is time-dependent. Its evolution will be governed by adaptation rules that I will describe presently. $C(n)$ need not be positive semidefinite at all times; only when the adaptation of $C(n)$ converges, the resulting C matrices will have the algebraic properties of conceptors.

One can conceive of the system (49) as a two-layered neural network, where the two layers have the same number of neurons, and where the layers are reciprocally

connected by the connection matrices C and W (Figure 27). The two layers have states

$$r(n+1) = \tanh(W^* z(n) + W^{\text{in}} p(n+1) + b) \quad (52)$$
$$z(n+1) = C\, r(n+1). \quad (53)$$

The r layer has sigmoidal (here: tanh) units and the z layer has linear ones. The customary reservoir state x becomes split into two states r and z, which can be conceived of as states of two pools of neurons.

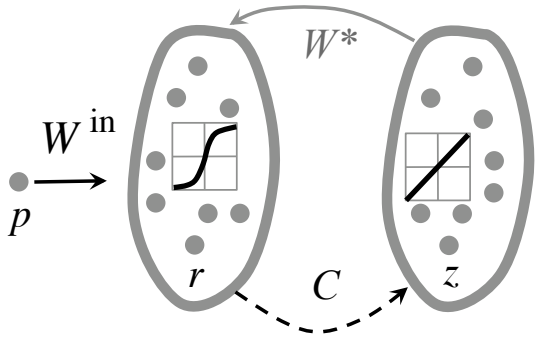

Figure 27: Network representation of a basic autoconceptor system. Bias b and optional readout mechanisms are omitted. The broken arrow indicates that C connections are online adaptive. For explanation see text.

In order to determine an adaptation law for $C(n)$, I replicate the line of reasoning that was employed to motivate the design of alloconceptors in Section 3.4. Alloconceptors were designed to act as "regularized identity maps", which led to the defining criterion (6) in Definition 1:

$$C(R, \alpha) = \text{argmin}_C \, E[\|x - Cx\|^2] + \alpha^{-2} \|C\|^2_{\text{fro}}.$$

The reservoir states x that appear in this criterion resulted from the update equation $x(n+1) = \tanh(W^* x(n) + W^{\text{in}} p(n+1) + b)$. This led to the explicit solution (7) stated in Proposition 1:

$$C(R, \alpha) = R\,(R + \alpha^{-2}\, I)^{-1},$$

where R was the reservoir state correlation matrix $E[xx']$. I re-use this criterion (6), which leads to an identical formula $C = R\,(R + \alpha^{-2}\, I)^{-1}$ for autoconceptors. The crucial difference is that now the state correlation matrix R depends on C:

$$R = E[zz'] = E[Cr(Cr)'] = C\, E[rr']\, C =: CQC, \quad (54)$$

where we introduce $Q = E[rr']$. This transforms the direct computation formula (7) to a fixed-point equation:

$$C = CQC\,(CQC + \alpha^{-2}\, I)^{-1},$$

which is equivalent to

$$(C - I)CQC - \alpha^{-2}C = 0. \tag{55}$$

Since Q depends on r states, which in turn depend on z states, which in turn depend on C again, Q depends on C and should be more appropriately be written as Q_C. Analysing the fixed-point equation $(C - I)CQ_CC - \alpha^{-2}C = 0$ is a little inconvenient, and I defer this to Section 3.13.4. When one uses autoconceptors, however, one does not need to explicitly solve (55). Instead, one can resort to a version of the incremental adaptation rule (11) from Proposition 2:

$$C(n+1) = C(n) + \lambda \left((z(n) - C(n)\, z(n))\, z'(n) - \alpha^{-2}\, C(n) \right),$$

which implements a stochastic gradient descent with respect to the cost function $E[\|z - Cz\|^2] + \alpha^{-2} \|C\|^2_{\text{fro}}$. In the new sitation given by (49), the state z here depends on C. This is, however, of no concern for using (11) in practice. We thus complement the reservoir state update rule (49) with the conceptor update rule (11) and comprise this in a definition:

Definition 5 *An* autoconceptive reservoir network *is a two-layered RNN with fixed weights W^*, W^{in} and online adaptive weights C, whose dynamics are given by*

$$\begin{aligned} z(n+1) &= C(n)\, \tanh(W^*\, z(n) + W^{in} p(n+1) + b) & (56) \\ C(n+1) &= C(n) + \lambda \left((z(n) - C(n)\, z(n))\, z'(n) - \alpha^{-2}\, C(n) \right), & (57) \end{aligned}$$

where λ is a learning rate and $p(n)$ an input signal. Likewise, when the update equation (56) is replaced by variants of the kind (50) or (51), we will speak of autoconceptive networks.

I will derive in Section 3.13.4 that if the driver p is stationary and if $C(n)$ converges under these rules, then the limit C is positive semidefinite with singular values in the set $(1/2, 1) \cup \{0\}$. Singular values asymptotically obtained under the evolution (57) are either "large" (that is, greater than 1/2) or they are zero, but they cannot be "small" but nonzero. If the aperture α is fixed at increasingly smaller values, increasingly many singular values will be forced to zero. Furthermore, the analysis in Section 3.13.4 will also reveal that among the nonzero singular values, the majority will be close to 1. Both effects together mean that autoconceptors are typically approximately hard conceptors, which can be regarded as an intrinsic mechanism of contrast enhancement, or noise suppression.

3.13.3 Example: Autoconceptive Reservoirs as Content-Addressable Memories

In previous sections I demonstrated how loaded patterns can be retrieved if the associated conceptors are plugged into the network dynamics. These conceptors

must have been stored beforehand. The actual memory functionality thus resides in whatever mechanism is used to store the conceptors; furthermore, a conceptor is a heavyweight object with the size of the reservoir itself. It is biologically implausible to create and "store" such a network-like object for every pattern that is to be recalled.

In this section I describe how autoconceptor dynamics can be used to create content-addressable memory systems. In such systems, recall is triggered by a cue presentation of the item that is to be recalled. The memory system then should in some way autonomously "lock into" a state or dynamics which autonomously re-creates the cue item. In the model that will be described below, this "locking into" spells out as running the reservoir in autoconceptive mode (using equations (51) and (57)), by which process a conceptor corresponding to the cue pattern shapes itself and enables the reservoir to autonomously re-generate the cue.

The archetype of content-addressable neural memories is the Hopfield network [48]. In these networks, the cue is a static pattern (technically a vector, in demonstrations often an image), which typically is corrupted by noise or incomplete. If the Hopfield network has been previously trained on the uncorrupted complete pattern, its recurrent dynamics will converge to an attractor state which re-creates the trained original from the corrupted cue. This *pattern completion* characteristic is the essence of the memory functionality in Hopfield networks. In the autoconceptive model, the aspect of completion manifests itself in that the cue is a *brief* presentation of a dynamic pattern, too short for a conceptor to be properly adapted. After the cue is switched off, the autoconceptive dynamics continues to shape the conceptor in an entirely autonomous way, until it is properly developed and the reservoir re-creates the cue.

This autoconceptive adaptation is superficially analog to the convergence to an attractor point in Hopfield networks. However, there are important conceptual and mathematical differences between the two models. I will discuss them at the end of this section.

Demonstration of basic architecture. To display the core idea of a content-addressable memory, I ran simulations according to the following scheme:

1. **Loading.** A collection of k patterns p^j $(j = 1, \ldots, k)$ was loaded in an N-dimensional reservoir, yielding an input simulation matrix D as described in Equation (40), and readout weights W^{out}, as described in Section 3.3. No conceptors are stored.

2. **Recall.** For each pattern p^j, a recall run was executed which consisted of three stages:

   (a) **Initial washout.** Starting from a zero network state, the reservoir was driven with p^j for n_{washout} steps, in order to obtain a task-related reservoir state.

(b) **Cueing.** The reservoir was continued to be driven with p^j for another n_{cue} steps. During this cueing period, C^j was adapted by using $r(n+1) = \tanh(Wr(n) + W^{\text{in}}p^j(n) + b)$, $C^j(n+1) = C^j(n) + \lambda^{\text{cue}}\left((r(n) - C^j(n)\,r(n))\,r'(n) - \alpha^{-2}\,C^j(n)\right)$. At the beginning of this period, C^j was initialized to the zero matrix. At the end of this period, a conceptor $C^{j\ \text{cue}}$ was obtained.

(c) **Autonomous recall.** The network run was continued for another n_{recall} steps in a mode where the input was switched off and replaced by the input simulation matrix D, and where the conceptor $C^{j\ \text{cue}}$ was further adapted autonomously by the autoconceptive update mechanism, via $z(n+1) = C^j(n)\,\tanh(Wz(n) + Dz(n) + b)$, $C^j(n+1) = C^j(n) + \lambda^{\text{recall}}\left((z(n) - C^j(n)\,z(n))\,z'(n) - \alpha^{-2}\,C^j(n)\right)$. At the end of this period, a conceptor $C^{j\ \text{recall}}$ was available.

3. **Measuring quality of conceptors.** The quality of the conceptors $C^{j\ \text{cue}}$ and $C^{j\ \text{recall}}$ was measured by separate offline runs without conceptor adaptation using $r(n) = \tanh(Wz(n) + Dz(n) + b)$; $z(n+1) = C^{j\ \text{cue}}\,r(n)$ (or $z(n+1) = C^{j\ \text{recall}}\,r(n)$, respectively). A reconstructed pattern $y(n) = W^{\text{out}}r(n)$ was obtained and its similarity with the original pattern p^j was quantified in terms of a NRMSE.

I carried out two instances of this experiment, using two different kinds of patterns and parametrizations:

4-periodic pattern. The patterns were random integer-periodic patterns of period 4, where per pattern the four pattern points were sampled from a uniform distribution and then shift-scaled such that the range became $[-1\ \ 1]$. This normalization implies that the patterns are drawn from an essentially 3-parametric family (2 real-valued parameters for fixing the two pattern values not equal to -1 or 1; one integer parameter for fixing the relative temporal positioning of the -1 and 1 values). Experiment parameters: $k = 10, N = 100, \alpha = 100, n_{\text{washout}} = 100, n_{\text{cue}} = 15, n_{\text{recall}} = 300, \gamma^{\text{cue}} = 0.02, \gamma^{\text{recall}} = 0.01$ (full detail in Section 4.4).

Mix of 2 irrational-period sines. Two sines of period lengths $\sqrt{30}$ and $\sqrt{30}/2$ were added with random phase angles and random amplitudes, where however the two amplitudes were constrained to sum to 1. This means that patterns were drawn from a 2-parametric family. Parameters: $k = 10, N = 200, \alpha = 100, n_{\text{washout}} = 100, n_{\text{cue}} = 30, n_{\text{recall}} = 10000, \gamma^{\text{cue}} = \gamma^{\text{recall}} = 0.01$. Furthermore, during the auto-adaptation period, strong Gaussian iid noise was added to the reservoir state before applying the tanh, with a signal-to-noise rate of 1.

Figure 28 illustrates the outcomes of these two experiments. Main observations:

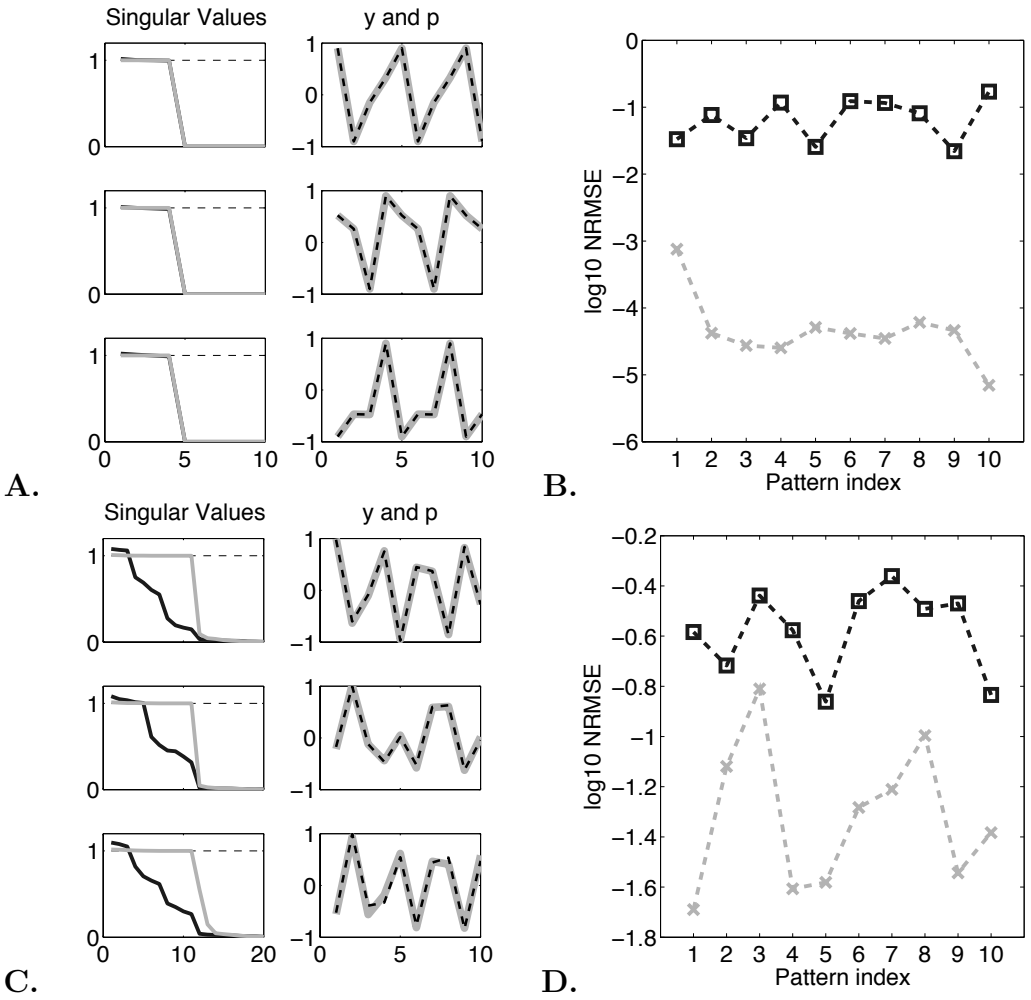

Figure 28: Basic content-addressable memory demos. **A**, **B**: 4-periodic pattern, **C**, **D**: mix of sines pattern. Panels **A**, **C** show the first three of the 10 patterns. The singular value plots show the first 10 (20, respectively) singular values of $C^{j\ \mathrm{cue}}$ (black) and $C^{j\ \mathrm{recall}}$ (light gray). The "y and p" panels show the reconstructed pattern y obtained with $C^{j\ \mathrm{recall}}$ (bold light gray) and the original training pattern p^j (broken black), after optimal phase-alignment. **B**, **D** plot the pattern reconstruction NRMSEs in log10 scale for the reconstructions obtained from $C^{j\ \mathrm{cue}}$ (black squares) and from $C^{j\ \mathrm{recall}}$ (gray crosses). For explanation see text.

1. In all cases, the quality of the preliminary conceptor $C^{j\,\mathrm{cue}}$ was very much improved by the subsequent auto-adaptation (Panels **B**, **D**), leading to an ultimate pattern reconstruction whose quality is similar to the one that would be obtained from precomputed/stored conceptors.

2. The effects of the autoconceptive adaptation are reflected in the singular value profiles of $C^{\mathrm{j}\,\mathrm{cue}}$ versus $C^{j\,\mathrm{recall}}$ (Panels **A**, **C**). This is especially well visible in the case of the sine mix patterns (for the period-4 patterns the effect is too small to show up in the plotting resolution). During the short cueing time, the online adaptation of the conceptor from a zero matrix to $C^{j\,\mathrm{cue}}$ only manages to build up a preliminary profile that could be intuitively called "nascent", which then "matures" in the ensuing network-conceptor interaction during the autoconceptive recall period.

3. The conceptors $C^{j\,\mathrm{recall}}$ have an almost rectangular singular value profile. In the next section I will show that if autoconceptive adaptation converges, singular values are either exactly zero or greater than 0.5 (in fact, typically close to 1), in agreement with what can be seen here. Autoconceptive adaptation has a strong tendency to lead to almost hard conceptors.

4. The fact that adapted autoconceptors typically have a close to rectangular singular value spectrum renders the auto-adaptation process quite immune against even strong state noise. Reservoir state noise components in directions of the nulled eigenvectors are entirely suppressed in the conceptor-reservoir loop, and state noise components within the nonzero conceptor eigenspace do not impede the development of a "clean" rectangular profile. In fact, state noise is even beneficial: it speeds up the auto-adaptation process without a noticeable loss in final pattern reconstruction accuracy (comparative simulations not documented here).

   This noise robustness however depends on the existence of zero singular values in the adapting autoconceptor C. In the simulations reported above, such zeros were present from the start because the conceptor was initialized as the zero matrix. If it had been initialized differently (for instance, as identity matrix), the auto-adaptation would only asymptotically pull (the majority of) singular values to zero, with noise robustness only gradually increasing to the degree that the singular value spectrum of C becomes increasingly rectangular. If noise robustness is desired, it can be reached by additional adaptation mechanisms for C. In particular, it is helpful to include a thresholding mechanism: all singular values of $C(n)$ exceeding a suitable threshold are set to 1, all singular values dropping below a certain cutoff are zeroed (not shown).

Exploring the effects of increasing memory load – patterns from a parametrized family. A central theme in neural memory research is the ca-

pacity of a neural storage system. In order to explore how recall accuracy depends on the memory load, I carried out two further experiments, one for each pattern type. Each of these experiments went along the following scheme:

1. Create a reservoir.
2. In separate trials, load this reservoir with an increasing number k of patterns (ranging from $k = 2$ to $k = 200$ for the 4-period and from $k = 2$ to $k = 100$ for the mixed sines).
3. After loading, repeat the recall scheme described above, with the same parameters. Monitor the recall accuracy obtained from $C^{j\,\mathrm{recall}}$ for the first 10 of the loaded patterns (if less than 10 were loaded, do it only for these).
4. In addition, per trial, also try to cue and "recall" 10 novel patterns that were drawn randomly from the 4-periodic and mixed-sine family, respectively, and which were not part of the collection loaded into the reservoir. Monitor the "recall" accuracy of these novel patterns as well.
5. Repeat this entire scheme 5 times, with freshly created patterns, but re-using always the same reservoir.

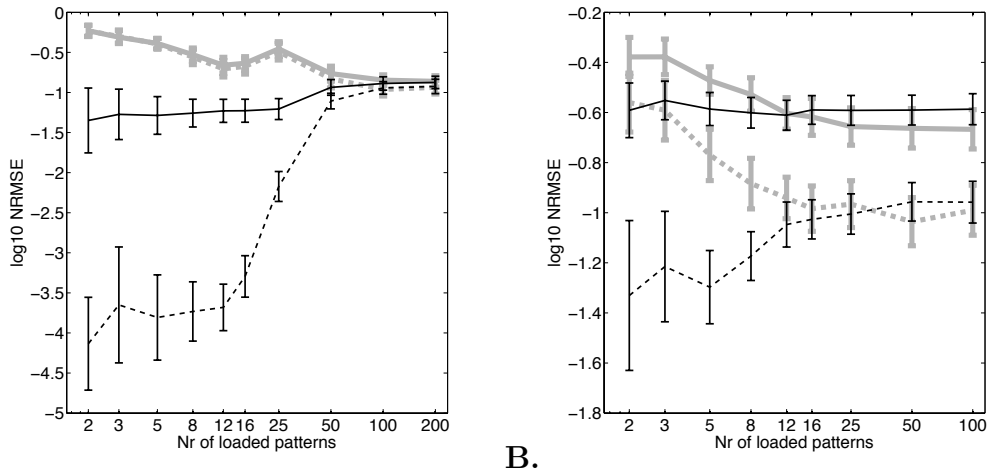

Figure 29: Exploring the effects of memory load. **A:** 4-periodic patterns, **B:** mix-of-sines patterns. Each diagram shows the log10 NRMSE of recalling loaded patterns with C^{cue} (black solid line) and with C^{recall} (black broken line), as well as of "recalling" patterns not contained in the loaded set, again obtained from C^{cue} (gray solid line) and with C^{recall} (gray broken line). Error bars indicate 95 % confidence intervals. Both axes are in logarithmic scaling. For explanation see text.

Figure 29 shows the results of these experiments. The plotted curves are the summary averages over the 10 recall targets and the 5 experiment repetitions.

Each plot point in the diagrams thus reflects an average over 50 NRMSE values (except in cases where $k < 10$ patterns were stored; then plotted values correspond to averages over $5k$ NRMSE values for recalling of loaded patterns). I list the main findings:

1. For all numbers of stored patterns, and for both the *recall loaded patterns* and *recall novel patterns* conditions, the autoconceptive "maturation" from C^{cue} to C^{recall} with an improvement of recall accuracy is found again.

2. The final C^{recall}-based recall accuracy in the *recall loaded pattern* condition has a sigmoid shape for both pattern types. The steepest ascent of the sigmoid (fastest deterioration of recall accuracy with increase of memory load) occurs at about the point where the summed quota of all C^{cue} reaches the reservoir size N – the point where the network is "full" according to this criterion (a related effect was encountered in the incremental loading study reported in Section 3.11). When the memory load is further increased beyond this point (one might say the network becomes "overloaded"), the recall accuracy does not break down but levels out on a plateau which still translates into a recall performance where there is a strong similarity between the target signal and the reconstructed one.

3. In the *recall novel patterns* conditions, one finds a steady improvement of recall accuracy with increasing memory load. For large memory loads, the accuracy in the *recall novel patterns* condition is virtually the same as in the *recall loaded patterns* conditions.

Similar findings were obtained in other simulation studies (not documented here) with other types of patterns, where in each study the patterns were drawn from a parametrized family.

A crucial characteristic of these experiments is that the patterns were samples from a parametrized family. They shared a family resemblance. This mutual relatedness of patterns is exploited by the network: for large numbers k of stored patterns, the storing/recall mechanism effectively acquires a model of the entire parametric family, a circumstance revealed by the essentially equal recall accuracy in the *recall loaded patterns* and *recall novel patterns* conditions. In contrast, for small k, the *recall loaded patterns* condition enables a recall accuracy which is superior to the *recall novel patterns* condition: the memory system stores/recalls individual patterns. I find this worth a special emphasis:

- For small numbers of loaded patterns (before the point of network overloading) the system stores and recalls individual patterns. *The input simulation matrix D represents* **individual** *patterns.*

- For large numbers of loaded patterns (overloading the network), the system learns a representation of the parametrized pattern family and can be cued

with, and will "recall", any pattern from that family. *The input simulation matrix D represents the* **class** *of patterns.*

At around the point of overloading, the system, in a sense, changes its nature from a mere storing-of-individuals device to a learning-of-class mechanism. I call this the *class learning effect.*

Effects of increasing memory load – mutually unrelated patterns. A precondition for the class learning effect is that the parametric pattern family is simple enough to become represented by the network. If the pattern family is too richly structured to be captured by a given network size, or if patterns do not have a family resemblance at all, the effect cannot arise. If a network is loaded with such patterns, and then cued with novel patterns, the "recall" accuracy will be on chance level; furthermore, as k increases beyond the overloading region, the recall accuracy of patterns contained in the loaded collection will decline to chance level too.

In order to demonstrate this, I loaded the same 100-unit reservoir that was used in the 4-periodic pattern experiments with random periodic patterns whose periods ranged from 3 through 9. While technically this is still a parametric family, the number of parameters needed to characterize a sample pattern is 8, which renders this family far too complex for a 100-unit reservoir. Figure 30 illustrates what, expectedly, happens when one loads increasingly large numbers of such effectively unrelated patterns. The NRMSE for the *recall novel patterns* condition is about 1 throughout, which corresponds to entirely uncorrelated pattern versus reconstruction pairs; and this NRMSE is also approached for large k in the *recall loaded patterns* condition.

To be or not to be an attractor. (This part addresses readers who are familiar with dynamical systems theory.) In the light of basic concepts provided by dynamical systems theory, and in the light of how Hopfield networks are known to function as content-addressable memories, one might conjecture that the cueing/recalling dynamics should be mathematically described as follows:

- The loading process creates a number of attractors (periodic attractors in our demonstrations) in the combined conceptor/reservoir dynamics.
- Each loaded pattern becomes represented by such an attractor.
- During the cueing phase, the combined conceptor/network state becomes located in the basin of the attractor corresponding to the cue pattern.
- In the remaining recall phase, the combined conceptor/network dynamics converges toward this attractor.

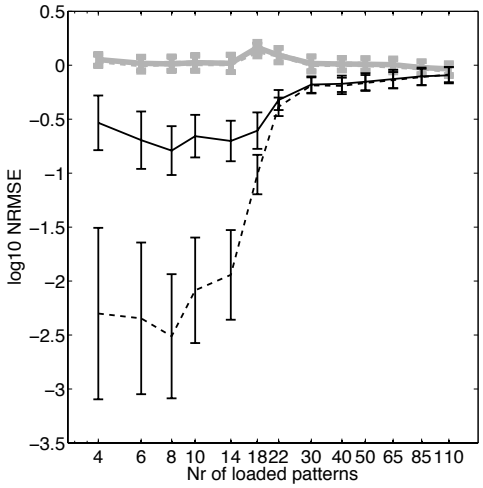

Figure 30: Effects of memory load on recall accuracy for unrelated patterns. Figure layout as in Figure 29. For explanation see text.

This, roughly, would be the picture if our content-addressable memory would function in an analog way as Hopfield networks, with the only difference being that instead of point attractors (in Hopfield networks) we are now witnessing cyclic attractors. To submit this picture to a more in-depth test, I re-ran the 4-periodic recall experiment documented in Figure 29 **A** with some modifications. Only a single experiment was run (no averaging over different collections of loaded patterns), and only patterns contained in the loaded collection were cued/recalled. Five of the loaded patterns were cued (or less if fewer had been loaded) and the found accuracy NRMSEs were averaged. The recall accuracy was measured offline after 200, 2000, and 20000 steps of the post-cue autoconceptive adaptation. The development of accuracy with recall runtime should be illustrative of convergence characteristics. This experiment was done in two versions, first with clean cue signals and then again with cues that were overlaid with relatively weak noise (sampled from the standard normal distribution, scaled by 1/20). If the autoconceptive adaptation would obey the attractor interpretation, one would expect that both in the noise-free and the noisy cue conditions, a convergence to identical good recall patterns is obtained.

The outcome of this experiment, illustrated in Figure 31, reveals that this is not the case. For clean cues (panel **A**), the recall accuracies after 200, 2000, 20000 are virtually identical (and very good for small numbers of loaded patterns). This is consistent with an interpretation of (fast) convergence to an attractor. If this interpretation were correct, then adding a small amount of noise to the cue should not affect the system's convergence to the same attractor. However this is not what is obtained experimentally (panel **B**). The curves for increasing post-cue runlengths cross each other and do not reach the accuracy levels from the

clean cue condition. While it is not clear how to interpret this outcome in terms of dynamical systems convergence, it certainly is not convergence to the same attractors (if they exist) as in the clean cue case.

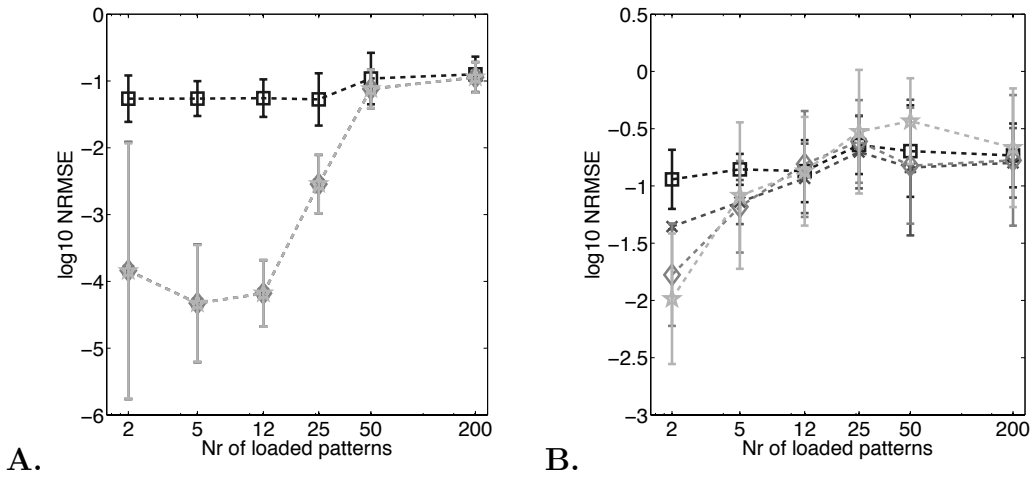

Figure 31: A staged re-run of the 4-periodic pattern recall with noise-free cues (**A**) and cues distorted by weak noise (**B**). Black line with square marks: NRMSE after cue; lighter-shaded lines with cross / diamond / star marks: NRMSEs after 200, 2000, 20000 time steps of recall adaptation. The 200, 2000, 20000 curves are almost identical in (**A**) and appear as a single line. For explanation see text.

A tentative explanation of these findings could be attempted as follows (a refined account will be given in Section 3.13.4). First, a simplification. If the adaptation rate γ in (57) is sufficiently small, a separation of timescales between a fast network state dynamics (56) and a slow conceptor adaptation dynamics occurs. It then makes sense to average over states $z(n)$. Let $\bar{z}_C$ be the fast-timescale average of network states under the governance of a conceptor C. This results in an autonomous continuous-time N^2-dimensional dynamical system in C parameters,

$$\tau \dot{C} = (\bar{z}_C - C)\, \bar{z}'_C - \alpha^{-2} C, \tag{58}$$

where τ is a time constant which is of no concern for the qualitative properties of the dynamics. The dynamics (58) represents the adaptation dynamics of an autoconceptor. Note that this adaptation dynamics does not necessarily preserve C as a positive semidefinite matrix, that is, matrices C obtained during adaptation need not be conceptor matrices. However, as proved in the next section, fixed point solutions of (58) are conceptor matrices.

Let us consider first a situation where a reservoir has been "overloaded" with a finite but large number of patterns from a simple parametric family. According to our simulation findings, it is then possible to re-generate with high accuracy any of the infinitely many patterns from the family by cued auto-adaptation of conceptors.

A natural candidate to explain this behavior is to assume that in the dynamical system (58) governing the C evolution, the loading procedure has established an instance of what is known as *line attractor* or *plane attractor* [25]. An attractor of this kind is characterized by a manifold (line or plane or higher-dimensional surface), consisting of fixed points of the system dynamics, which attracts trajectories from its neighborhood. Figure 32 attempts to visualize this situation. In our case, the plane attractor would consist of fixed point solutions of (58), that is, conceptors that will ultimately be obtained when the recall adaptation converges. The arrows in this figure represent trajectories of (58).

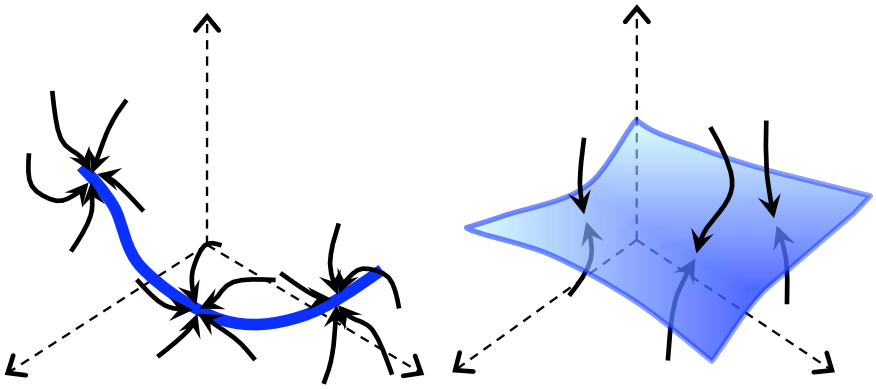

Figure 32: Schematic phase portraits of plane attractors. An m-dimensional invariant manifold consisting of neutrally stable fixed points is embedded in an n-dimensional state space of a dynamical system. The figure shows an example where $m = 1, n = 3$ (a "line attractor", left) and an example where $m = 2$ ("plane attractor"). Trajectories in a neighborhood of the fixed-point manifold are attracted toward it (arrows show selected trajectories). For explanations see text.

In cases where the reservoir has been loaded with only a small number of patterns (from the same parametric family or otherwise), no such plane attractor would be created. Instead, each of the loaded patterns becomes represented by an isolated point attractor in (58). I mention again that this picture, while it is in agreement with the simulations done so far, is preliminary and will be refined in Section 3.13.4.

Discussion. Neural memory mechanisms – how to store patterns in, and retrieve from, neural networks – is obviously an important topic of research. Conceptor-based mechanisms bring novel aspects to this widely studied field.

The paradigmatic model for content-addressable storage of patterns in a neural network is undoubtedly the family of auto-associative neural networks (AANNs) whose analysis and design was pioneered by Palm [80] and Hopfield [48] (with a rich history in theoretical neuroscience, referenced in [80]). Most of these models are characterized by the following properties:

- AANNs with N units are used to store static patterns which are themselves N-dimensional vectors. The activity profile of the entire network coincides with the very patterns. In many demonstrations, these patterns are rendered as 2-dimensional images.

- The networks are typically employed, after training, in pattern completion or restauration tasks, where an incomplete or distorted N-dimensional pattern is set as the initial N-dimensional network state. The network then should evolve toward a completed or restored pattern state.

- AANNs have symmetric connections and (typically) binary neurons. Their recurrent dynamics can be formalized as a descent along an energy function, which leads to convergence to fixed points which are determined by the input pattern.

- An auto-associative network is trained from a set of k reference patterns, where the network weights are adapted such that the network state energy associated with each training pattern is minimized. If successful, this leads to an energy landscape over state space which assumes local minima at the network states that are identical to the reference patterns.

The comprehensive and transparent mathematical theory available for AANNs has left a strong imprint on our preconceptions of what are essential features of a content-addressable neural memory. Specifically, AANN research has settled the way how the task of storing items in an associative memory is framed in the first place: "given k reference patterns, train a network such that in exploitation, these patterns can be reconstructed from incomplete cues". This leads naturally to identifying stored memory items with attractors in the network dynamics. Importantly, memory items are seen as *discrete*, individual entities. For convenience I will call this the "discrete items stored as attractors" (DISA) paradigm.

Beyond modeling memory functionality proper, the DISA paradigm is historically and conceptually connected to a wide range of models of neural representations of conceptual knowledge, where attractors are taken as the neural representatives of discrete concepts. To name only three kinds of such models: point attractors (cell assemblies and bistable neurons) in the working memory literature [23]; spatiotemporal attractors in neural field theories of cortical representation [93, 30, 31]; (lobes of) chaotic attractors as richly structured object and percept representations [109, 7].

Attractors, by definition, keep the system trajectory confined within them. Since clearly cognitive processes do not become ultimately trapped in attractors, it has been a long-standing modeling challenge to account for "attractors that can be left again" – that is, to partly disengage from a strict DISA paradigm. Many answers have been proposed. Neural noise is a plausible agent to "kick" a trajectory out of an attractor, but a problem with noise is its unspecificity which is not easily reconciled with systematic information processing. A number

of alternative "attractor-like" phenomena have been considered that may arise in high-dimensional nonlinear dynamics and offer escapes from the trapping problem: *saddle point dynamics* or *homoclinic cycles* [87, 41]; *chaotic itinerancy* [103]; *attractor relics*, *attractor ruins*, or *attractor ghosts* [101]; *transient attractors* [53]; *unstable attractors* [102]; *high-dimensional attractors* (initially named *partial attractors*) [70]; *attractor landscapes* [76].

All of these lines of work revolve around a fundamental conundrum: on the one hand, neural representations of conceptual entities need to have some kind of stability – this renders them identifiable, noise-robust, and temporally persistent when needed. On the other hand, there must be cognitively meaningful mechanisms for a fast switching between neural representational states or modes. This riddle is not yet solved in a widely accepted way. Autoconceptive plane attractor dynamics may lead to yet another answer. This kind of dynamics intrinsically combines dynamical stability (in directions complementary to the plane of attraction) with dynamical neutrality (within the plane attractor). However, in the next section we will see that this picture, while giving a good approximation, is too simple.

3.13.4 Analysis of Autoconceptor Adaptation Dynamics

Here I present a formal analysis of some asymptotic properties of the conceptor adaptation dynamics.

Problem Statement

We consider the system of the coupled fast network state updates and slow conceptor adaptation given by

$$z(n+1) = C(n)\ \tanh(W\, z(n)) \tag{59}$$

and

$$\begin{aligned} C(n+1) &= C(n) + \lambda \left((z(n+1) - C(n)\, z(n+1))\, z'(n+1) - \alpha^{-2}\, C(n) \right) \\ &= C(n) + \lambda \left((I - C(n))\, z(n+1) z'(n+1) - \alpha^{-2}\, C(n), \right) \end{aligned} \tag{60}$$

where λ is a learning rate. When λ is small enough, the instantaneous state correlation $z(n)z'(n)$ in (60) can be replaced by its expectation under C fixed at $C(n)$, that is, we consider the dynamical system in time k

$$z(k+1) = C(n) \tanh(W z(k))$$

and take the expectation of zz' under this dynamics,

$$\begin{aligned} E_n[zz'] &:= E_k[C(n) \tanh(W\, z(k)) \tanh(W\, z(k))' C(n)'] \\ &= C(n)\, E_k[\tanh(W\, z(k)) \tanh(W\, z(k))'] C(n)' \quad =: \quad C(n) Q(n) C(n)', \end{aligned}$$

where $Q(n)$ is a positive semi-definite correlation matrix. Note that $Q(n)$ is a function of C and itself changes on the slow timescale of the C adaptation. For further analysis it is convenient to change to continuous time and instead of (60) consider

$$\dot{C}(t) = (I - C(t))\, C(t)Q(t)C'(t) - \alpha^{-2}C(t). \tag{61}$$

I now investigate the nature of potential fixed point solutions under this dynamics. If C is a fixed point of this dynamics, $Q(t)$ is constant. In order to investigate the nature of such fixed point solutions, we analyse solutions in C for the general fixed point equation associated with (61), i.e. solutions in C of

$$0 = (I - C)\, CQC' - \alpha^{-2}C, \tag{62}$$

where Q is some positive semidefinite matrix. We will denote the dimension of C by N throughout the remainder of this section. Let $VDV' = Q$ be the SVD of Q, where D is a diagonal matrix containing the singular values of Q on its diagonal, without loss of generality in descending order. Then (62) is equivalent to

$$\begin{aligned} 0 &= V'((I - C)\, CQC' - \alpha^{-2}C)V \\ &= (I - V'CV)\, V'CV\, D(V'CV)' - \alpha^{-2}\, V'CV. \end{aligned} \tag{63}$$

We may therefore assume that Q is in descending diagonal form D, analyse solutions of

$$0 = (I - C)\, CDC' - \alpha^{-2}C, \tag{64}$$

and then transform these solutions C of (64) back to solutions of (62) by $C \to VCV'$. In the remainder we will only consider solutions of (64). I will characterize the fixed points of this system and analyse their stability properties.

Characterizing the Fixed-point Solutions

The case $\alpha = 0$. In this degenerate case, neither the discrete-time update rule (60) nor the dynamical equation (61) is well-defined. The aperture cannot be set to zero in practical applications where (60) is used for conceptor adaptation.

However, it is clear that (i) for any $\alpha > 0$, $C = 0$ is a fixed point solution of (61), and that (ii) if we define $B(\alpha) = \sup\{\|C\| \mid C \text{ is a fixed-point solution of (61)}\}$, then $\lim_{\alpha \to 0} B(\alpha) = 0$. This justifies to set, by convention, $C = 0$ as the unique fixed point of (61) in the case $\alpha = 0$. In practical applications this could be implemented by a reset mechanism: whenever $\alpha = 0$ is set by some superordinate control mechanism, the online adaptation (60) is over-ruled and $C(n)$ is immediately set to 0.

The case $\alpha = \infty$. In the case $\alpha = \infty$ (i.e., $\alpha^{-2} = 0$) our task is to characterize the solutions C of

$$0 = (I - C)CDC'. \tag{65}$$

We first assume that D has full rank. Fix some $k \leq N$. We proceed to characterize rank-k solutions C of (65). CDC' is positive semidefinite and has rank k, thus it has an SVD $CDC' = U\Sigma U'$ where Σ is diagonal nonnegative and can be assumed to be in descending order, i.e. its diagonal is $(\sigma_1, \ldots, \sigma_k, 0, \ldots, 0)'$ with $\sigma_i > 0$. Any solution C of (65) must satisfy $U\Sigma U' = CU\Sigma U'$, or equivalently, $\Sigma = U'CU\Sigma$. It is easy to see that this entails that $U'CU$ is of the form

$$U'CU = \begin{pmatrix} I_{k\times k} & 0 \\ 0 & A \end{pmatrix}$$

for some arbitrary $(n-k)\times(n-k)$ submatrix A. Requesting $\mathrm{rank}(C) = k$ implies $A = 0$ and hence

$$C = U \begin{pmatrix} I_{k\times k} & 0 \\ 0 & 0 \end{pmatrix} U'. \tag{66}$$

Since conversely, if U is any orthonormal matrix, a matrix C of the form given in (66) satisfies $C^2 = C$, any such C solves (65). Therefore, the rank-k solutions of (65) are exactly the matrices of type (66).

If D has rank $l < N$, again we fix a desired rank k for solutions C. Again let $CDC' = U\Sigma U'$, with Σ in descending order. Σ has a rank m which satisfies $m \leq k, l$. From considering $\Sigma = U'CU\Sigma$ it follows that $U'CU$ has the form

$$U'CU = \begin{pmatrix} I_{m\times m} & 0 \\ 0 & A \end{pmatrix}$$

for some $(N-m)\times(N-m)$ submatrix A. Since we prescribed C to have rank k, the rank of A is $k-m$. Let $U_{>m}$ be the $n \times (N-m)$ submatrix of U made from the columns with indices greater than m. We rearrange $U\Sigma U' = CDC'$ to

$$\Sigma = \begin{pmatrix} I_{m\times m} & 0 \\ 0 & A \end{pmatrix} U'\, D\, U \begin{pmatrix} I_{m\times m} & 0 \\ 0 & A \end{pmatrix},$$

from which it follows (since the diagonal of Σ is zero at positions greater than m) that $AU'_{>m}DU_{>m}A' = 0$. Since the diagonal of D is zero exactly on positions $> l$, this is equivalent to

$$A\,(U(1:l, m+1:N))' = A\,U(m+1:N, 1:l) = 0. \tag{67}$$

We now find that (67) is already sufficient to make $C = U\,(I_{m\times m}|0\,/\,0|A)\,U'$ solve (65), because a simple algebraic calculation yields

$$CD = CCD = U \begin{pmatrix} (U(1:N, 1:m))' \\ 0 \end{pmatrix} D.$$

We thus have determined the rank-k solutions of (65) to be all matrices of form $C = U\,(I_{m\times m}|0\,/\,0|A)\,U'$, subject to (i) $m \leq l, k$, (ii) $\mathrm{rank}(A) = k-m$, (iii) $A\,(U(1:l, m+1:N))' = 0$. Elementary considerations (omitted here) lead to the following generative procedure to obtain all of these matrices:

1. Choose m satisfying $l - N + k \leq m \leq k$.

2. Choose a size $N \times l$ matrix $\tilde{U}'$ made from orthonormal columns which is zero in the last $k - m$ rows (this is possible due to the choice of m).

3. Choose an arbitrary $(N - m) \times (N - m)$ matrix A of SVD form $A = V \Delta W'$ where the diagonal matrix Δ is in ascending order and is zero exactly on the first $N - k$ diagonal positions (hence $\mathrm{rank}(A) = k - m$).

4. Put
$$\tilde{\tilde{U}}' = \begin{pmatrix} I_{m \times m} & 0 \\ 0 & W \end{pmatrix} \tilde{U}'.$$
This preserves orthonormality of columns, i.e. $\tilde{\tilde{U}}'$ is still made of orthonormal columns. Furthermore, it holds that $(0|A)\,\tilde{\tilde{U}}' = 0$.

5. Pad $\tilde{\tilde{U}}'$ by adding arbitrary $N - l$ further orthonormal colums to the right, obtaining an $N \times N$ orthonormal U'.

6. We have now obtained a rank-k solution
$$C = U \begin{pmatrix} I_{m \times m} & 0 \\ 0 & A \end{pmatrix} U', \tag{68}$$
where we have put U to be the transpose of the matrix U' that was previously constructed.

The case $0 < \alpha < \infty$. We proceed under the assumption that $\alpha < \infty$, that is, $\infty > \alpha^{-2} > 0$.

I first show that any solution C of (64) is a positive semidefinite matrix. The matrix CDC' is positive semidefinite and therefore has a SVD of the form $U\Sigma U' = CDC'$, where U is orthonormal and real and Σ is the diagonal matrix with the singular values of CDC' on its diagonal, without loss of generality in descending order. From $(I - C)U\Sigma U' = \alpha^{-2}\, C$ it follows that

$$\begin{aligned} U\Sigma U' &= \alpha^{-2}\, C + C\, U\Sigma U' = C\,(\alpha^{-2} I + U\Sigma U') \\ &= C\,(U(\alpha^{-2} I + \Sigma)U'). \end{aligned}$$

$\alpha^{-2} I + \Sigma$ and hence $U(\alpha^{-2} I + \Sigma)U'$ are nonsingular because $\alpha^{-2} > 0$, therefore

$$C = U\Sigma U'(U(\alpha^{-2} I + \Sigma)U')^{-1} = U\Sigma(\alpha^{-2}\, I + \Sigma)^{-1} U' =: USU',$$

where $S = \Sigma(\alpha^{-2}\, I + \Sigma)^{-1}$ is a diagonal matrix, and in descending order since Σ was in descending order. We therefore know that any solution C of (64) is of the form $C = USU'$, where U is the same as in $CDC' = U\Sigma U'$. From $S = \Sigma(\alpha^{-2}\, I + \Sigma)^{-1}$ it furthermore follows that $s_i < 1$ for all singular values s_i of C, that is, C is a conceptor matrix.

We now want to obtain a complete overview of all solutions $C = USU'$ of (64), expressed in terms of an orthonormal real matrix U and a nonnegative real diagonal matrix S. This amounts to finding the solutions in S and U of

$$(S - S^2)U'DUS = \alpha^{-2}\, S, \tag{69}$$

subject to S being nonnegative real diagonal and U being real orthonormal. Without loss of generality we furthermore may assume that the entries in S are in descending order.

Some observations are immediate. First, the rank of S is bounded by the rank of D, that is, the number of nonzero diagonal elements in S cannot exceed the number of nonzero elements in D. Second, if U, S is a solution, and S^* is the same as S except that some nonzero elements in S are nulled, then U, S^* is also a solution (to see this, left-right multiply both sides of (69) with a thinned-out identity matrix that has zeros on the diagonal positions which one wishes to null).

Fix some $k \leq \mathrm{rank}(D)$. We want to determine all rank-k solutions U, S, i.e. where S has exactly k nonzero elements that appear in descending order in the first k diagonal positions. We write S_k to denote diagonal real matrices of size $k \times k$ whose diagonal entries are all positive. Furthermore, we write U_k to denote any $N \times k$ matrix whose columns are real orthonormal.

It is clear that if S, U solve (69) and $\mathrm{rank}(S) = k$ (and S is in descending order), and if U^* differs from U only in the last $N - k$ columns, then also S, U^* solve (69). Thus, if we have all solutions S_k, U_k of

$$(S_k - S_k^2)U_k'DU_kS_k = \alpha^{-2}\, S_k, \tag{70}$$

then we get all rank-k solutions S, U to (64) by padding S_k with $N - k$ zero rows/columns, and extending U_k to full size $N \times n$ by appending any choice of orthonormal columns from the orthogonal complement of U_k. We therefore only have to characterize the solutions S_k, U_k of (70), or equivalently, of

$$U_k'DU_k = \alpha^{-2}\,(S_k - S_k^2)^{-1}. \tag{71}$$

To find such S_k, U_k, we first consider solutions $\tilde{S}_k, U_k$ of

$$U_k'DU_k = \tilde{S}_k, \tag{72}$$

subject to $\tilde{S}_k$ being diagonal with positive diagonal elements. For this we employ the Cauchy interlacing theorem and its converse. I restate, in a simple special case adapted to the needs at hand, this result from [27] where it is presented in greater generality.

Theorem 1 *(Adapted from Theorem 1 in [27], see remark of author at the end of the proof of that theorem for a justification of the version that I render here.) Let A, B be two symmetric real matrices with $dim(A) = n \geq k = dim(B)$, and singular values $\sigma_1, \ldots, \sigma_n$ and $\tau_1, \ldots, \tau_k$ (in descending order). Then there exists a real $n \times k$ matrix U with $U'U = I_{k\times k}$ and $U'AU = B$ if and only if for $j = 1, \ldots, k$ it holds that $\sigma_i \geq \tau_i \geq \sigma_{n-k+i}$.*

This theorem implies that if $U_k, \tilde{S}_k$ is any solution of (72), with U_k made of k orthonormal columns and $\tilde{S}_k$ diagonal with diagonal elements $\tilde{s}_i$ (where $j = 1, \ldots, k$, and the enumeration is in descending order), then the latter "interlace" with the diagonal entries $d_1, \ldots, d_N$ of D per $d_i \geq \tilde{s}_i \geq d_{N-k+i}$. And conversely, any diagonal matrix $\tilde{S}_k$, whose elements interlace with the diagonal elements of D, appears in a solution $U_k, \tilde{S}_k$ of (72).

Equipped with this overview of solutions to (72), we revert from (72) to (71). Solving $\tilde{S}_k = \alpha^{-2}\,(S_k - S_k^2)^{-1}$ for S_k we find that the diagonal elements s_i of S_k relate to the $\tilde{s}_i$ by

$$s_i = \frac{1}{2}\left(1 \pm \sqrt{1 - \frac{4\alpha^{-2}}{\tilde{s}_i}}\right). \tag{73}$$

Since s_i must be positive real and smaller than 1, only such solutions $\tilde{S}_k$ to (72) whose entries are all greater than $4\alpha^{-2}$ yield admissible solutions to our original problem (71). The interlacing condition then teaches us that the possible rank of solutions C of (64) is bounded from above by the number of entries in D greater than $4\alpha^{-2}$.

For each value $\tilde{s}_i > 4\alpha^{-2}$, (73) gives two solutions $s_{i,1} < 1/2 < s_{i,2}$. We will show further below that the solutions smaller than $1/2$ are unstable while the solutions greater than $1/2$ are stable in a certain sense.

Summarizing and adding algorithmic detail, we obtain all rank-k solutions $C = USU'$ for (64) as follows:

1. Check whether D has at least k entries greater than $4\alpha^{-2}$. If not, there are no rank-k solutions. If yes, proceed.

2. Find a solution in $U_k, \tilde{S}_k$ of $U_k' D U_k = \tilde{S}_k$, with $\tilde{S}_k$ being diagonal with diagonal elements greater than $4\alpha^{-2}$, and interlacing with the elements of D. (Note: the proof of Theorem 1 in [27] is constructive and could be used for finding U_k given $\tilde{S}_k$.)

3. Compute S_k via (73), choosing between the $\pm$ options at will.

4. Pad U_k with any orthogonal complement and S_k with further zero rows and columns to full $n \times n$ sized U, S, to finally obtain a rank-k solution $C = USU'$ for (64).

Stability Analysis of Fixed-point Solutions

The case $\alpha = \infty$. Note again that $\alpha = \infty$ is the same as $\alpha^{-2} = 0$. We consider the time evolution of the quantity $\|I - C\|^2$ as C evolves under the zero-α^{-2} version of (61):

$$\dot{C}(t) = (I - C(t))\, C(t) Q(t) C'(t). \tag{74}$$

We obtain

$$\begin{aligned}(\|I-C\|^2)^{\cdot} &= \operatorname{trace}((I-C)(I-C')^{\cdot}\\ &= \operatorname{trace}((C-C^2)QC'C' + CCQ(C'-C'^2) - (C-C^2)QC' - CQ(C'-C'^2))\\ &= 2\operatorname{trace}((C-C^2)Q(C'^2-C'))\\ &= -2\operatorname{trace}((C-C^2)Q(C'-C'^2)) \le 0, \end{aligned} \tag{75}$$

where in the last line we use that the trace of a positive semidefinite matrix is nonnegative. This finding instructs us that no other than the identity $C = I$ can be a stable solution of (74), in the sense that all eigenvalues of the associated Jacobian are negative. If $Q(t)$ has full rank for all t, then indeed this is the case (it is easy to show that $\|I - C(t)\|^2$ is strictly decreasing, hence a Lyapunov function in a neighborhood of $C = I$).

The stability characteristics of other (not full-rank) fixed points of (74) are intricate. If one computes the eigenvalues of the Jacobian at rank-k fixed points C (i.e. solutions of sort $C = U(I_{k\times k}|0\ /\ 0|0)U'$, see (66)), where $k < N$, one finds negative values and zeros, but no positive values. (The computation of the Jacobian follows the pattern of the Jacobian for the case $\alpha < \infty$, see below, but is simpler; it is omitted here). Some of the zeros correspond to perturbation directions of C which change only the coordinate transforming matrices U. These perturbations are neutrally stable in the sense of leading from one fixed point solution to another one, and satisfy $C + \Delta = (C + \Delta)^2$. However, other perturbations $C + \Delta$ with the property that $C+\Delta \neq (C+\Delta)^2$ lead to $(\|I-C\|^2)^{\cdot} < 0$. After such a perturbation, the matrix $C + \Delta$ will evolve toward I in the Frobenius norm. Since the Jacobian of C has no positive eigenvalues, this instability is non-hyperbolic. In simulations one accordingly finds that after a small perturbation Δ is added, the divergence away from C is initially extremely slow, and prone to be numerically misjudged to be zero.

For rank-deficient Q, which leads to fixed points of sort $C = U(I_{m\times m}|0\ /\ 0|A)U'$, the computation of Jacobians becomes involved (mainly because A may be non-symmetric) and I did not construct them. In our context, where Q derives from a random RNN, Q can be expected to have full rank, so a detailed investigation of the rank-deficient case would be an academic exercise.

The case $0 < \alpha < \infty$. This is the case of greatest practical relevance, and I spent a considerable effort on elucidating it.

Note that $\alpha < \infty$ is equivalent to $\alpha^{-2} > 0$. Let $C_0 = USU'$ be a rank-k fixed point of $\dot{C} = (I - C)CDC' - \alpha^{-2}C$, where $\alpha^{-2} > 0$, USU' is the SVD of C and without loss of generality the singular values $s_1, \ldots, s_N$ in the diagonal matrix S are in descending order, with $s_1, \ldots, s_k > 0$ and $s_i = 0$ for $i > k$ (where $1 \le k \le N$). In order to understand the stability properties of the dynamics $\dot{C}$ in a neighborhood of C_0, we compute the eigenvalues of the Jacobian $J_C = \partial\dot{C}/\partial C$ at point C_0. Notice that C is an $N \times N$ matrix whose entries must be rearranged

into a vector of size $N^2 \times 1$ in order to arrive at the customary representation of a Jacobian. J_C is thus an $N^2 \times N^2$ matrix which should be more correctly written as $J_C(\mu,\nu) = \partial\, vec\, \dot{C}(\mu)/\partial\, vec\, C(\nu)$, where *vec* is the rearrangement operator ($1 \leq \mu, \nu \leq N^2$ are the indices of the matrix J_C). Details are given in Section 5.10 within the proof of the following central proposition:

Proposition 16 *The Jacobian $J_C(\mu,\nu) = \partial\, vec\, \dot{C}(\mu)/\partial\, vec\, C(\nu)$ of a rank-k fixed point of (61) has the following multiset of eigenvalues:*

1. *$k(N-k)$ instances of 0,*
2. *$N(N-k)$ instances of $-\alpha^{-2}$,*
3. *k eigenvalues $\alpha^{-2}(1-2s_l)/(1-s_l)$, where $l = 1, \ldots, k$,*
4. *$k(k-1)$ eigenvalues which come in pairs of the form*

$$\lambda_{1,2} = \frac{\alpha^{-2}}{2}\left(\frac{s_l}{s_m - 1} + \frac{s_m}{s_l - 1} \pm \sqrt{\left(\frac{s_l}{s_m - 1} - \frac{s_m}{s_l - 1}\right)^2 + 4}\right),$$

where $m < l < k$.

An inspection of sort *3.* eigenvalues reveals that whenever one of the s_l is smaller than $1/2$, this eigenvalue is positive and hence the fixed point C_0 is unstable.

If some s_l is exactly equal to $1/2$, one obtains additional zero eigenvalues by *3.* I will exclude such cases in the following discussion, considering them to be non-generic.

If all s_l are greater than $1/2$, it is straightforward to show that the values of sorts *3.* and *4.* are negative. Altogether, J_P thus has $k(N-k)$ times the eigenvalue 0 and otherwise negative ones. I will call such solutions *1/2-generic*. All solutions that one will effectively obtain when conceptor auto-adaptation converges are of this kind.

This characterization of the eigenvalue spectrum of 1/2-generic solutions does not yet allow us to draw firm conclusions about how such a solution will react to perturbations. There are two reasons why Proposition 16 affords but a partial insight in the stability of 1/2-generic solutions. (A) The directions connected to zero eigenvalues span a $k(N-k)$-dimensional center manifold whose dynamics remains un-analysed. It may be stable, unstable, or neutral. (B) When a 1/2-generic solution is perturbed, the matrix D which reflects the conceptor-reservoir interaction will change: D is in fact a function of C and should be more correctly written $D = D(C)$. In our linearization around fixed point solutions we implicitly considered D to be constant. It is unclear whether a full treatment using $D = D(C)$ would lead to a different qualitative picture. Furthermore, (A) and (B) are liable to combine their effects. This is especially relevant for the dynamics on the

center manifold, because its qualitative dynamics is determined by components from higher-order approximations to (61) which are more susceptible to become qualitatively changed by non-constant D than the dynamical components of (61) orthogonal to the center manifold.

Taking (A) and (B) into account, I now outline a hypothetical picture of the dynamics of (61) in the vicinity of 1/2-generic fixed-point solutions. This picture is based only on plausibility considerations, but it is in agreement with what I observe in simulations.

First, a dimensional argument sheds more light on the nature of the dynamics in the $k(N-k)$-dimensional center manifold. Consider a 1/2-generic rank-k solution $C = USU'$ of (69). Recall that the singular values s_i in S were derived from $\tilde{s}_i$ which interlace with the diagonal elements $d_1, \ldots, d_n$ of D by $d_i \geq \tilde{s}_i \geq d_{N-k+i}$, and where $U_k' D U_k = \tilde{S}_k$ (Equation (72)). I call C a *1/2&interlacing-generic* solution if the interlacing is proper, i.e. if $d_i > \tilde{s}_i > d_{N-k+i}$. Assume furthermore that $D(C)$ is constant in a neighborhood of C. In this case, differential changes to U_k in (72) will lead to differential changes in $\tilde{S}_k$. If these changes to U_k respect the conditions (i) that U_k remains orthonormal and (ii) that $\tilde{S}_k$ remains diagonal, the changes to U_k lead to new fixed point solutions. The first constraint (i) allows us to change U_k with $(N-1)+(N-2)+\ldots+(N-k) = kN - k(k+1)/2$ degrees of freedom. The second constraint (ii) reduces this by $(k-1)k/2$ degrees of freedom. Altogether we have $kN - k(k+1)/2 - (k-1)k/2 = k(N-k)$ differential directions of change of C that lead to new fixed points. This coincides with the dimension of the center manifold associated with C. We can conclude that the center manifold of a 1/2&interlacing-generic C extends exactly in the directions of neighboring fixed point solutions. This picture is based however on the assumption of constant D. If the dependency of D on C is included in the picture, we would not expect to find any other fixed point solutions at all in a small enough neighborhood of C. Generically, fixed point solutions of an ODE are isolated. Therefore, in the light of the considerations made so far, we would expect to find isolated fixed point solutions C, corresponding to close approximations of stored patterns. In a local vicinity of such solutions, the autoconceptor adaptation would presumably progress on two timescales: a fast convergence toward the center manifold $\mathcal{K}$ associated with the fixed point C, superimposed on a slow convergence toward C within $\mathcal{K}$ (Figure 33 **A**).

The situation becomes particularly interesting when many patterns from a d-parametric class have been stored. When I first discussed this situation (Section 3.13.3), I tentatively described the resulting autoadaptation dynamics as convergence toward a plane attractor (Figure 32). However, plane attractors cannot be expected to exist in generic dynamical systems. Taking into account what the stability analysis above has revealed about center manifolds of fixed points C, I would like to propose the following picture as a working hypothesis for the geometry of conceptor adaptation dynamics that arises when a d-parametric pattern class has been stored by overloading:

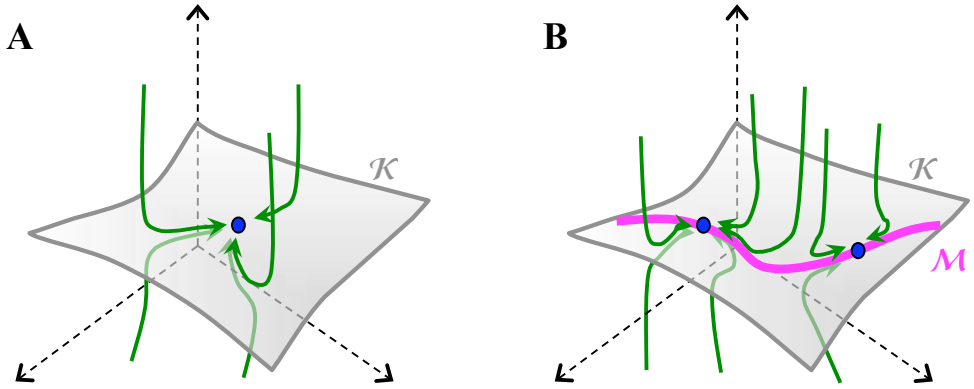

Figure 33: Hypothetical phase portraits of C autoadaptation in the parameter space of C (schematic). Blue points show stable fixed point solutions C. Gray plane represents the merged center manifold $\mathcal{K}$. Green arrows represent sample trajectories of C adaptation. **A.** When a small number of patterns has been loaded, individual stable fixed point conceptors C are created. **B.** In the case of learning a d-parametric pattern class, fixed point solutions C_i become located within a d-dimensional pattern manifold $\mathcal{M}$ (bold magenta line). For explanation see text.

- The storing procedure leads to a number of stable fixed point solutions C_i for the autoconceptor adaptation (blue dots in Figure 33 **B**). These C_i are associated with patterns from the pattern family, but need not coincide with the sample patterns that were loaded.

- The $k(N-k)$-dimensional center manifolds of the C_i merge into a comprehensive manifold $\mathcal{K}$ of the same dimension. In the vicinity of $\mathcal{K}$, the autoadaptive C evolution leads to a convergence toward $\mathcal{K}$.

- Within $\mathcal{K}$ a d-dimensional submanifold $\mathcal{M}$ is embedded, representing the learnt class of patterns. Notice that we would typically expect $d << k(N-k)$ (examples in the previous section had $d = 2$ or $d = 3$, but $k(N-k)$ in the order of several 100). Conceptor matrices located on $\mathcal{M}$ correspond to patterns from the learnt class.

- The convergence of C adaptation trajectories toward $\mathcal{K}$ is superimposed with a slower contractive dynamics within $\mathcal{K}$ toward the class submanifold $\mathcal{M}$.

- The combined effects of the attraction toward $\mathcal{K}$ and furthermore toward $\mathcal{M}$ appear in simulations as if $\mathcal{M}$ were acting as a plane attractor.

- On an even slower timescale, within $\mathcal{M}$ there is an attraction toward the isolated fixed point solutions C_i. This timescale is so slow that the motion within $\mathcal{M}$ toward the fixed points C_i will be hardly observed in simulations.

In order to corroborate this refined picture, and especially to confirm the last point from the list above, I carried out a long-duration content-addressable memory simulation along the lines described in Section 3.13.3. Ten 5-periodic patterns were loaded into a small (50 units) reservoir. These patterns represented ten stages of a linear morph between two similar patterns p^1 and p^{10}, resulting in a morph sequence $p^1, p^2, \ldots, p^{10}$ where $p^i = (1 - (i-1)/9)\, p^1 + ((i-1)/9)\, p^{10}$, thus representing instances from a 1-parametric family. Considering what was found in Section 3.13.3, loading these ten patterns should enable the system to re-generate by auto-adaptation any linear morph p_{test} between p^1 and p^{10} after being cued with p_{test}.

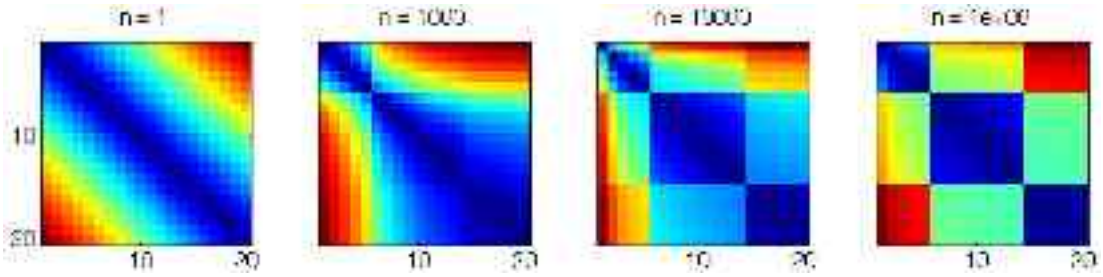

Figure 34: Numerical exploration of fixed point solutions under C auto-adaptation. Each panel shows pairwise distances of 20 conceptor matrices obtained after n auto-adaptation steps, after being cued along a 20-step morph sequence of cue signals. Color coding: blue – zero distance; red – maximum distance. For explanation see text.

After loading, the system was cued with 20 different cues. In each of these $j = 1, \ldots, 20$ conditions, the cueing pattern p^j_{test} was the j-th linear interpolation between the stored p^1 and p^{10}. The cueing was done for 20 steps, following the procedure given at the beginning of Section 3.13.3. At the end of the cueing, the system will be securely driven into a state z that is very accurately connected to re-generating the pattern p^j_{test}, and the conceptor matrix that has developed by the end of the cueing would enable the system to re-generate a close simile of p^j_{test} (a post-cue log10 NRMSE of about -2.7 was obtained in this simulation).

After cueing, the system was left running in conceptor auto-adaptation mode using (57) for 1 Mio timesteps, with an adaptation rate of $\lambda = 0.01$.

At times $n = 1, 1000, 10000, 1e^6$ the situation of convergence was assessed as follows. The pairwise distances between the current twenty autoconceptors $C^j(n)$ were compared, resulting in a 20×20 distance matrix $D(n) = (\|C^k(n) - C^l(n)\|_{\text{fro}})_{k,l=1,\ldots,20}$. Figure 34 shows color plots of these distance matrices. The outcome: at the beginning of autoadaptation ($n = 1$), the 20 autoconceptors are spaced almost equally widely from each other. In terms of the schematic in Figure 33 **B**, they would all be almost equi-distantly lined up close to $\mathcal{M}$. Then, as the adaptation time n grows, they contract toward three point attractors within $\mathcal{M}$ (which would correspond to a version of 33 **B** with three blue dots). These three

point attractors correspond to the three dark blue squares on the diagonal of the last distance matrix shown in Figure 34.

This singular simulation cannot, of course, provide conclusive evidence that the qualitative picture proposed in Figure 33 is correct. A rigorous mathematical characterization of the hypothetical manifold $\mathcal{M}$ and its relation to the center manifolds of fixed point solutions of the adaptation dynamics needs to be worked out.

Plane attractors have been proposed as models for a number of biological neural adaptation processes (summarized in [24]). A classical example is gaze direction control. The fact that animals can fix their gaze in arbitrary (continuously many) directions has been modelled by plane attractors in the oculomotoric neural control system. Each gaze direction corresponds to a (controlled) constant neural activation profile. In contrast to and beyond such models, conceptor auto-adaptation organized along a manifold $\mathcal{M}$ leads not to a continuum of *constant* neural activity profiles, but explains how a continuum of *dynamical* patterns connected by continuous morphs can be generated and controlled.

In sum, the first steps toward an analysis of autoconceptor adaptation have revealed that this adaptation dynamics is more involved than either the classical fixed-point dynamics in autoassociative memories or the plane attractor models suggested in computational neuroscience. For small numbers of stored patterns, the picture bears some analogies with autoassociative memories in that stable fixed points of the autonomous adaptation correspond to stored patterns. For larger numbers of stored patterns (class learning), the plane attractor metaphor captures essential aspects of phenomena seen in simulations of not too long duration.

3.14 Toward Biologically Plausible Neural Circuits: Random Feature Conceptors

The autoconceptive update equations

$$\begin{aligned} z(n+1) &= C(n)\, \tanh(Wz(n) + Dz(n) + b) \\ C(n+1) &= C(n) + \lambda\, (z(n) - C(n)z(n))\, z'(n) - \alpha^{-2} C(n) \end{aligned}$$

could hardly be realized in biological neural systems. One problem is that the C update needs to evaluate $C(n)z(n)$, but $z(n)$ is not an *input* to $C(n)$ in the z update but the *outcome* of applying C. The input to C is instead the state $r(n) = \tanh(Wz(n) + Dz(n) + b)$. In order to have both computations carried out by the same C, it seems that biologically hardly feasible schemes of installing two weight-sharing copies of C would be required. Another problem is that the update of C is nonlocal: the information needed for updating a "synapse" C_{ij} (that is, an element of C) is not entirely contained in the presynaptic or postsynaptic signals available at this synapse.

Here I propose an architecture which solves these problems, and which I think has a natural biological "feel". The basic idea is to (i) randomly expand the

reservoir state r into a (much) higher-dimensional *random feature space*, (ii) carry out the conceptor operations in that random feature space, but in a simplified version that only uses scalar operations on individual state components, and (iii) project the conceptor-modulated high-dimensional feature space state back to the reservoir by another random projection. The reservoir-conceptor loop is replaced by a two-stage loop, which first leads from the reservoir to the feature space (through connection weight vectors f_i, collected column-wise in a random neural projection matrix F), and then back to the reservoir through a likewise random set of backprojection weights G (Figure 35 **A**). The reservoir-internal connection weights W are replaced by the combination of F and G, and the original reservoir state x known from the basic matrix conceptor framework is split into a reservoir state vector r and a feature space state vector z with components $z_i = c_i f_i' r$. The conception weights c_i take over the role of conceptors. In full detail,

1. expand the N-dimensional reservoir state $r = \tanh(Wz + W^{\text{in}}p + b)$ into the M-dimensional random feature space by a random feature map $F' = (f_1, \ldots, f_M)'$ (a synaptic connection weight matrix of size $M \times N$) by computing the M-dimensional feature vector $F'r$,
2. multiply each of the M feature projections $f_i' r$ with an adaptive *conception weight* c_i to get a conceptor-weighted feature state $z = \text{diag}(c)\, F' r$, where the *conception vector* $c = (c_1, \ldots, c_M)'$ is made of the conception weights,
3. project z back to the reservoir by a random $N \times M$ backprojection matrix $\tilde{G}$, closing the loop.

Since both W and $\tilde{G}$ are random, they can be joined in a single random map $G = W\tilde{G}$. This leads to the following consolidated state update cycle of a *random feature conception* (RFC) architecture:

$$r(n+1) = \tanh(G\, z(n) + W^{\text{in}} p(n) + b), \tag{76}$$

$$z(n+1) = \text{diag}(c(n))\, F'\, r(n+1), \tag{77}$$

where $r(n) \in \mathbb{R}^N$ and $z(n), c(n) \in \mathbb{R}^M$.

From a biological modeling perspective there exist a number of concrete candidate mechanisms by which the mathematical operation of multiplying-in the conception weights could conceivably be realized. I will discuss these later and for the time being remain on this abstract mathematical level of description.

The conception vector $c(n)$ is adapted online and element-wise in a way that is analog to the adaptation of matrix autoconceptors given in Definition 5. Per each element c_i of c, the adaptation aims at minimizing the objective function

$$E[(z_i - c_i z_i)^2] + \alpha^{-2} c_i^2, \tag{78}$$

which leads to fixed point solutions satisfying

$$c_i = E[z_i^2](E[z_i^2] + \alpha^{-2})^{-1} \tag{79}$$

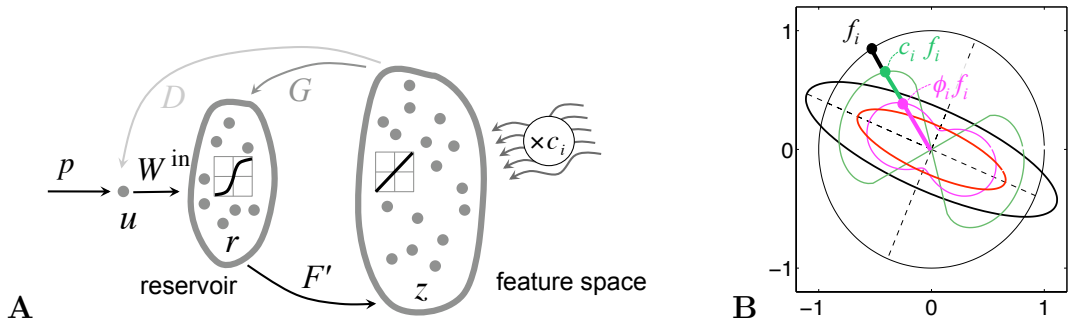

Figure 35: An alternative conceptor architecture aiming at greater biological plausibility. **A** Schematic of random feature space architecture. The reservoir state r is projected by a feature map F' into a higher-dimensional feature space with states z, from where it is back-projected by G into the reservoir. The conceptor dynamics is realized by unit-wise multiplying conception weights c_i into $f_i' r$ to obtain the z state. The input unit u is fed by external input p or by learnt input simulation weights D. **B** Basic idea (schematic). Black ellipse: reservoir state r correlation matrix R. Magenta dumbbell: scaling sample points f_i from the unit sphere by their mean squared projection on reservoir states. Green dumbbell: feature vectors f_i scaled by auto-adapted conception weights c_i. Red ellipse: the resulting virtual conceptor C_F. For detail see text.

and a stochastic gradient descent online adaptation rule

$$c_i(n+1) = c_i(n) + \lambda_i \left(z_i^2(n) - c_i(n)\, z_i^2(n) - \alpha^{-2}\, c_i(n) \right), \tag{80}$$

where $i = 1, \ldots, M$, λ_i is an adaptation rate, and z_i is the i-the component of z. In computer simulations one will implement this adaptation not element-wise but in an obvious vectorized fashion.

If (80) converges, the converged fixed point is either $c_i = 0$, which always is a possible and stable solution, or it is of the form

$$c_i = 1/2 + \sqrt{(\alpha^2 \phi_i - 4)/4\alpha^2 \phi_i}, \tag{81}$$

which is another possible stable solution provided that $\alpha^2 \phi_i - 4 > 0$. In this formula, ϕ_i denotes the expectation $\phi_i = E_r[(f_i' r)^2]$, the mean energy of the feature signal $f_i' r$. These possible values of stable solutions can be derived in a similar way as was done for the singular values of autoconceptive matrix C in Section 3.13.4, but the derivation is by far simpler (because it can be done element-wise for each c_i and thus entails only scalars, not matrices) and is left as an exercise. Like the singular values of stable autoconceptors C, the possible stable value range for conception weights obtainable through (80) is thus $\{0\} \cup (1/2,\, 1)$.

Some geometric properties of random feature conceptors are illustrated in Figure 35 **B**. The black ellipse represents the state correlation matrix $R = E[rr']$ of a hypothetical 2-dimensional reservoir. The random feature vectors f_i are assumed

to have unit norm in this schematic and therefore sample from the surface of the unit sphere. The magenta-colored dumbbell-shaped surface represents the weigthing of the random feature vectors f_i by the mean energies $\phi_i = E[(f_i' r)^2]$ of the feature signals $f_i' r$. Under the autoconception adaptation they give rise to conception weights c_i according to (81) (green dumbbell surface). For values $\alpha^2 \phi_i - 4 < 0$ one obtains $c_i = 0$, which shows up in the illustration as the wedge-shaped indentation in the green curve. The red ellipse renders the virtual conceptor C_F (see below) which results from the random feature conception weights.

Two properties of this RFC architecture are worth pointing out.

First, conceptor matrices C for an N-dimensional reservoir have $N(N+1)/2$ degrees of freedom. If, using conception vectors c instead, one wishes to attain a performance level of pattern reconstruction accuracy that is comparable to what can be achieved with conceptor matrices C, one would expect that M should be in the order of $N(N+1)/2$. At any rate, this is an indication that M should be significantly larger than N. In the simulations below I used $N = 100, M = 500$, which worked robustly well. In contrast, trying $M = 100$ (not documented), while likewise yielding good accuracies, resulted in systems that were rather sensitive to parameter settings.

Second, the individual adaptation rates λ_i can be chosen much larger than the global adaptation rate λ used for matrix conceptors, without putting stability at risk. The reason is that the original adpatation rate λ in the stochastic gradient descent formula for matrix conceptors given in Definition 5 is constrained by the highest local curvature in the gradient landscape, which leads to slow convergence in the directions of lower curvature. This is a notorious general characteristic of multidimensional gradient descent optimization, see for instance [28]. This problem becomes irrelevant for the individual c_i updates in (80). In the simulations presented below, I could safely select the λ_i as large as 0.5, whereas when I was using the original conceptor matrix autoadaption rules, $\lambda = 0.01$ was often the fastest rate possible. If adaptive individual adaptation rates λ_i would be implemented (not explored), very fast convergence of (80) should become feasible.

Geometry of feature-based conceptors. Before I report on simulation experiments, it may be helpful to contrast geometrical properties of the RFC architecture and with the geometry of matrix autoconceptors.

For the sake of discussion, I split the backprojection $N \times M$ matrix G in (76) into a product $G = W F$ where the "virtual" reservoir weight matrix $W := GF^\dagger$ has size $N \times N$. That is, I consider a system $z(n+1) = F \,\mathrm{diag}(c(n))\, F' \tanh(W z(n))$ equivalent to (76) and (77), where c is updated according to (80). For the sake of simplicity I omit input terms and bias in this discussion. The map $F \circ \mathrm{diag}(c) \circ F' : \mathbb{R}^N \to \mathbb{R}^N$ then plugs into the place that the conceptor matrix C held in the conceptor systems $z(n+1) = C \tanh(W z(n))$ discussed in previous sections. The question I want to explore is how $F \circ \mathrm{diag}(c) \circ F'$ compares to C in geometrical terms. A conceptor matrix C has an SVD $C = USU'$, where

U is orthonormal. In order to make the two systems directly comparable, I assume that all feature vectors f_i in F have unit norm. Then $C_F = \|F\|_2^{-2}\, F \circ \mathrm{diag}(c) \circ F'$ is positive semidefinite with 2-norm less or equal to 1, in other words it is an $N \times N$ conceptor matrix.

Now furthermore assume that the adaptation (80) has converged. The adaptation loop (76, 77, 80) is then a stationary process and the expectations $\phi_i = E_r[(f_i'\, r)^2]$ are well-defined. Note that these expectations can equivalently be written as $\phi_i = f_i'\, R\, f_i$, where $R = E[rr']$. According to what I remarked earlier, after convergence to a stable fixed point solution we have, for all $1 \leq i \leq M$,

$$c_i \in \begin{cases} \{0\}, & \text{if } \alpha^2\phi_i - 4 \leq 0, \\ \{0, 1/2 + \sqrt{(\alpha^2\phi_i - 4)/4\alpha^2\phi_i}\}, & \text{if } \alpha^2\phi_i - 4 > 0. \end{cases} \tag{82}$$

Again for the sake of discussion I restrict my considerations to converged solutions where all c_i that *can* be nonzero (that is, $\alpha^2\phi_i - 4 > 0$) are indeed nonzero.

It would be desirable to have an analytical result which gives the SVD of the $N \times N$ conceptor $C_F = \|F\|_2^{-2}\, F \circ \mathrm{diag}(c) \circ F'$ under these assumptions. Unfortunately this analysis appears to be involved and at this point I cannot deliver it. In order to still obtain some insight into the geometry of C_F, I computed a number of such matrices numerically and compared them to matrix-based autoconceptors C that were derived from the same assumed stationary reservoir state process. The outcome is displayed in Figure 36.

Concretely, these numerical investigations were set up as follows. The reservoir dimension was chosen as $N = 2$ to admit plotting. The number of features was $M = 200$. The feature vectors f_i were chosen as $(\cos(i\, 2\,\pi/M), \sin(i\, 2\,\pi/M))'$ (where $i = 1, \ldots, M$), that is, the unit vector $(1\ 0)'$ rotated in increments of $(i/M)\, 2\,\pi$. This choice mirrors a situation where a very large number of f_i would be randomly sampled; this would likewise result in an essentially uniform coverage of the unit circle. The conception weights c_i (and hence C_F) are determined by the reservoir state correlation matrix $R = E[rr']$. The same holds for autoconceptor matrices C. For an exploration of the C_F versus C geometries, I thus systematically varied $R = U\Sigma U'$. The principal directions U were randomly chosen and remained the same through all variations. The singular values $\Sigma = \mathrm{diag}(\sigma_1\ \sigma_2)$ were chosen as $\sigma_1 \equiv 10, \sigma_2 \in \{0, 1, 5\}$, which gave three versions of R. The aperture α was selected in three variants as $\alpha \in \{1, 2, 3\}$, which altogether resulted in nine (R, α) combinations.

For each of these combinations, conception weights c_i were computed via (82), from which C_F were obtained. Each of these maps the unit circle on an ellipse, plotted in Figure 36 in red. The values of the c_i are represented in the figure as the dumbbell-shaped curve (green) connecting the vectors $c_i\, f_i$. The wedge-shaped constriction to zero in some of these curves corresponds to angular values of f_i where $c_i = 0$.

For comparison, for each of the same (R, α) combinations also an autoconceptor matrix C was computed using the results from Section 3.13.4. We saw

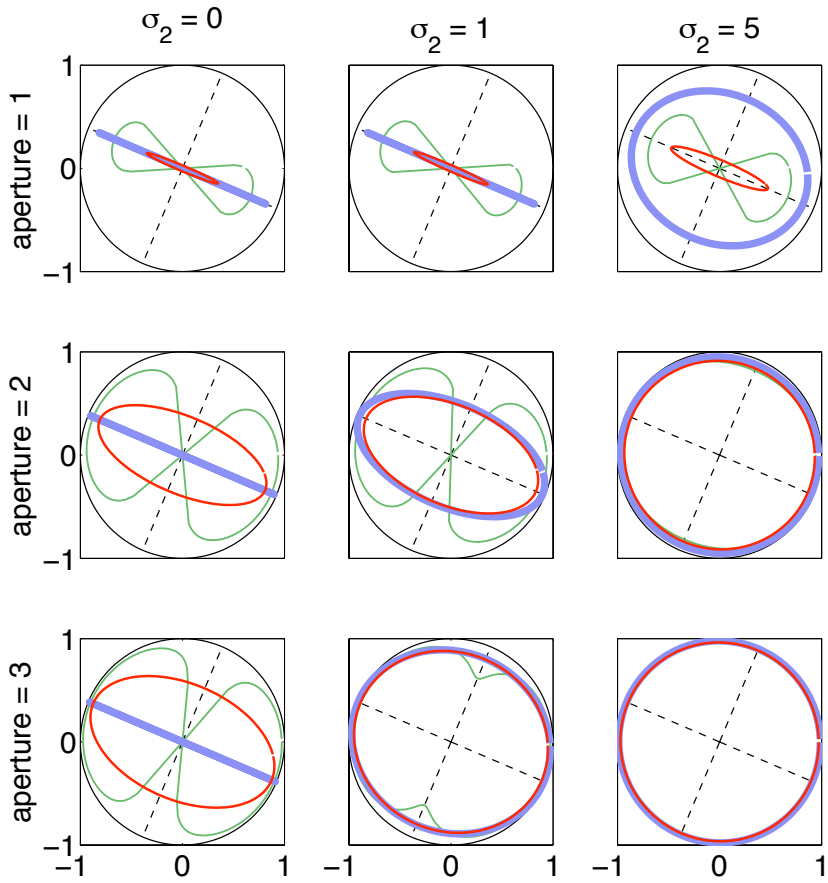

Figure 36: Comparing matrix-based autoconceptors (bold blue ellipses) with feature-based autoconceptors C_F (red ellipses). Broken lines mark principal directions of the reservoir state correlation matrix R. Each panel corresponds to a particular combination of aperture and the second singular value σ_2 of R. The dumbbell-shaped surfaces (green line) represent the values of the conception weights c_i. For explanation see text.

on that occasion that nonzero singular values of C are not uniquely determined by R; they are merely constrained by certain interlacing bounds. To break this indeterminacy, I selected those C that had the maximal admissible singular values. According to (73), this means that the singular values of C were set to $s_i = 1/2 + \sqrt{(\alpha^2\sigma_i - 4)/4\alpha^2\sigma_i}$ (where $i = 1, 2$) provided the root argument was positive, else $s_i = 0$.

Here are the main findings that can be collected from Figure 36:

1. The principal directions of C_F, C and R coincide. The fact that C_F and R have the same orientation can also be shown analytically, but the argument that I have found is (too) involved and not given here. This orientation of C_F hinges on the circumstance that the f_i were chosen to uniformly sample the unit sphere.

2. The C_F ellipses are all non-degenerate, that is, C_F has no zero singular values (although many of the c_i may be zero as becomes apparent in the wedge constrictions in the dumbbell-shaped representation of these values). In particular, the C_F are also non-degenerate in cases where the matrix autoconceptors C are (panels in left column and center top panel). The finding that C_F has no zero singular values can be regarded as a disadvantage compared to matrix autoconceptors, because it implies that no signal direction in the N-dimensional reservoir signal space can be completely suppressed by C_F. However, in the M-dimensional feature signal space, we do have nulled directions. Since the experiments reported below exhibit good stability properties in pattern reconstruction, it appears that this "purging" of signals in the feature space segment of the complete reservoir-feature loop is effective enough.

3. Call the ratio of the largest over the smallest singular value of C_F or C the *sharpness* of a conceptor (also known as eigenvalue spread in the signal processing literature). Then sometimes C_F is sharper than C, and sometimes the reverse is true. If sharpness is considered a desirable feature of concepors (which I think it often is), then there is no universal advantage of C over C_F or vice versa.

System initialization and loading patterns: generic description. Returning from this inspection of geometrical properties to the system (76) – (80), I proceed to describe the initial network creation and pattern loading procedure in generic terms. Like with the matrix conceptor systems considered earlier in this report, there are two variants which are the analogs of (i) recomputing the reservoir weight matrix W^*, as in Section 3.3, as opposed to (ii) training an additional input simulation matrix D, as in Section 3.11. In the basic experiments reported below I found that both work equally well. Here I document the second option. A readout weight vector W^{out} is likewise computed during loading. Let K target patterns p^j be given ($j = 1, \ldots, K$), which are to be loaded. Here is an outline:

Network creation. A random feature map F, random input weights W^{in}, a random bias vector b, and a random backprojection matrix G^* are generated. F and G^* are suitably scaled such that the combined $N \times N$ map $G^* F'$ attains a prescribed spectral radius. This spectral radius is a crucial system parameter and plays the same role as the spectral radius in reservoir computing in general (see for instance [106, 110]). All conception weights are initialized to $c_i^j = 1$, that is, $\text{diag}(c^j) = I_{M \times M}$.

Conception weight adaptation. The system is driven with each pattern p^j in turn for n_{adapt} steps (discarding an initial washout), while c^j is being adapted

per

$$\begin{aligned} z^j(n+1) &= \mathrm{diag}(c^j(n))\, F' \tanh(G^*\, z^j(n) + W^{\mathrm{in}} p^j(n) + b), \\ c_i^j(n+1) &= c_i^j(n) + \lambda_i \left(z_i^j(n)^2 - c_i^j(n)\, z_i^j(n)^2 - \alpha^{-2}\, c_i^j(n) \right), \end{aligned}$$

leading to conception vectors c^j at the end of this period.

State harvesting for computing D and W^{out}, and for recomputing G. The conception vectors c^j obtained from the previous step are kept fixed, and for each pattern p^j the input-driven system $r^j(n) = \tanh(G^*\, z^j(n) + W^{\mathrm{in}} p^j(n) + b)$; $z^j(n+1) = \mathrm{diag}(c^j)\, F'\, r^j(n)$ is run for n_{harvest} time steps, collecting states $r^j(n)$ and $z^j(n)$.

Computing weights. The $N \times M$ input simulation matrix D is computed by solving the regularized linear regression

$$D = \mathrm{argmin}_{\tilde{D}} \sum_{n,j} \| W^{\mathrm{in}} p^j(n) - \tilde{D} z^j(n-1) \|^2 + \beta_D^2 \, \| \tilde{D} \|_{\mathrm{fro}}^2 \qquad (83)$$

where β_D is a suitably chosen Tychonov regularizer. This means that the autonomous system update $z^j(n+1) = \mathrm{diag}(c^j)\, F' \tanh(G^* z^j(n) + W^{\mathrm{in}}\, D\, z^j(n) + b)$ should be able to simulate input-driven updates $z^j(n) = \mathrm{diag}(c^j)\, F' \tanh(G^*\, z^j(n) + W^{\mathrm{in}} p^j(n) + b)$. W^{out} is similarly computed by solving

$$W^{\mathrm{out}} = \mathrm{argmin}_{\tilde{W}} \sum_{n,j} \| p^j(n) - \tilde{W} r^j(n) \|^2 + \beta_{W^{\mathrm{out}}}^2 \, \| \tilde{W} \|^2.$$

Optionally one may also recompute G^* by solving the trivial regularized linear regression

$$G = \mathrm{argmin}_{\tilde{G}} \sum_{n,j} \| G^* z^j(n) - \tilde{G} z^j(n) \|^2 + \beta_G^2 \, \| \tilde{G} \|_{\mathrm{fro}}^2.$$

for a suitably chosen Tychonov regularizer β_G. While G^* and G should behave virtually identically on the training inputs, the average absolute size of entries in G will be (typically much) smaller than the original weights in G^* as a result of the regularization. Such regularized auto-adaptations have been found to be beneficial in pattern-generating recurrent neural networks [90], and in the experiments to be reported presently I took advantage of this scheme.

The feature vectors f_i that make up F can optionally be normalized such that they all have the same norm. In my experiments this was not found to have a noticeable effect.

If a stored pattern p^j is to be retrieved, the only item that needs to be changed is the conception vector c^j. This vector can either be obtained by re-activating that c^j which was adapted during the loading (which implies that it needs to be stored in some way). Alternatively, it can be obtained by autoadaptation without being previously stored, as in Sections 3.13.1 – 3.13.4. I now describe two simulation studies which demonstrate how this scheme functions (simulation detail documented in Section 4.6). The first study uses stored conception vectors, the second demonstrates autoconceptive adaptation.

Example 1: pattern retrieval with stored conception vectors c^j. This simulation re-used the $N = 100$ reservoir from Sections 3.2 *ff.* and the four driver patterns (two irrational-period sines, two very similar 5-periodic random patterns). The results are displayed in Figure 37.

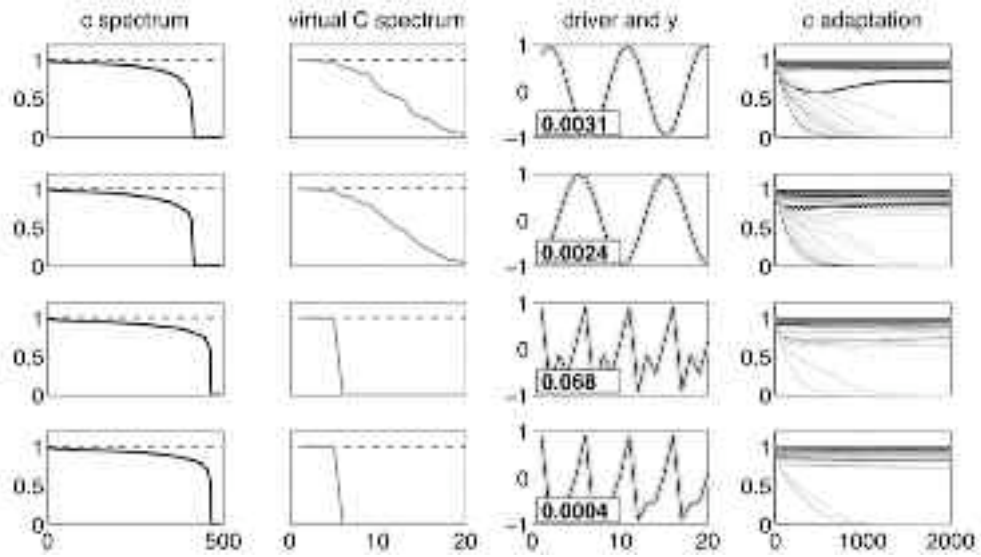

Figure 37: Using stored random feature coded conceptors in a replication of the basic pattern retrieval experiment from Section 3.4, with $M = 500$ random feature vectors f_i. First column: sorted conception vectors c^j. Second column: spectra of virtual conceptors C_F. Third column: reconstructed patterns (bold light gray) and original patterns (thin black) after phase alignment. NRMSEs are given in insets. Last column: The adaptation of c^j during the 2000 step runs carried out in parallel to the loading process. 50 of 500 traces are shown. For explanation see text.

The loading procedure followed the generic scheme described above (details in Section 4), with $n_{\text{adapt}} = 2000, n_{\text{harvest}} = 400$, $\lambda_i = 0.5$, $\beta_G^2 = \beta_D^2 = 0.01$ and $\beta_{W^{\text{out}}}^2 = 1$. The aperture was set to $\alpha = 8$. The left column in Figure 37 shows the resulting c^j spectra, and the right column shows the evolution of c^j during this adaptation. Notice that a considerable portion of the conception weights evolved toward zero, and that none ended in the range (0 1/2), in agreement with theory.

For additional insight into the dynamics of this system I also computed "virtual" matrix conceptors C_F^j by $R^j = E[(r^j)'r^j]$, $C_F^j = R^j\,(R^j + \alpha^{-2})^{-1}$ (second column). The singular value spectrum of C_F^j reveals that the autocorrelation spectra of r^j signals in RFC systems is almost identical to the singular value spectra obtained with matrix conceptors on earlier occasions (compare Figure 14).

The settings of matrix scalings and aperture were quite robust; variations in a range of about $\pm 50\%$ about the chosen values preserved stability and accuracy of pattern recall (detail in Section 4.6).

For testing the recall of pattern p^j, the loaded system was run using the update routine

$$\begin{aligned} r^j(n) &= \tanh(G\,z^j(n) + W^{\text{in}}\,D\,z^j(n) + b), \\ y^j(n) &= W^{\text{out}}\,r^j(n), \\ z^j(n+1) &= \text{diag}(c^j)\,F'\,r^j(n), \end{aligned}$$

starting from a random starting state $z^j(0)$ which was sampled from the normal distribution, scaled by $1/2$. After a washout of 200 steps, the reconstructed pattern y^j was recorded for 500 steps and compared to a 20-step segment of the target pattern p^j. The second column in Figure 37 shows an overlay of y^j with p^j and gives the NRMSEs. The reconstruction is of a similar quality as was found in Section 3.4 where full conceptor matrices C were used.

Example 2: content-addressed pattern retrieval. For a demonstration of content-addressed recall similar to the studies reported in Section 3.13.3, I reused the $M = 500$ system described above. Reservoir scaling parameters and the loading procedure were identical except that conception vectors c^j were not stored. Results are collected in Figure 38. The cue and recall procedure for a pattern p^j was carried out as follows:

1. Starting from a random reservoir state, the loaded reservoir was driven with the cue pattern for a washout time of 200 steps by $z^j(n+1) = F'\,\tanh(G\,z^j(n) + W^{\text{in}}\,p^j(n) + b)$.

2. Then, for a cue period of 800 steps, the system was updated with c^j adaptation by

$$\begin{aligned} z^j(n+1) &= \text{diag}(c^j(n))\,F'\,\tanh(G\,z^j(n) + W^{\text{in}}\,p^j(n) + b), \\ c_i^j(n+1) &= c_i^j(n) + \lambda_i\left(z_i^j(n)^2 - c_i^j(n)\,z_i^j(n)^2 - \alpha^{-2}\,c_i^j(n)\right), \quad (1 \le i \le M) \end{aligned}$$

   starting from an all-ones c^j, with an adaptation rate $\lambda_i = 0.5$ for all i. At the end of this period, a conception vector $c^{j,\text{cue}}$ was obtained.

3. To measure the quality of $c^{j,\mathrm{cue}}$, a separate run of 500 steps without c adapation was done using

$$\begin{aligned} r^j(n) &= \tanh(G\, z^j(n) + W^{\mathrm{in}}\, D\, z^j(n) + b), \\ y^j(n) &= W^{\mathrm{out}}\, r^j(n), \\ z^j(n+1) &= \mathrm{diag}(c^{j,\mathrm{cue}})\, F'\, r^j(n) \end{aligned}$$

obtaining a pattern reconstruction $y^j(n)$. This was phase-aligned with the original pattern p^j and an NRMSE was computed (Figure 38, third column).

4. The recall run was resumed after the cueing period and continued for another 10,000 steps in auto-adaptation mode, using

$$\begin{aligned} z^j(n+1) &= \mathrm{diag}(c^j(n))\, F'\, \tanh(G\, z^j(n) + W^{\mathrm{in}}\, D\, z^j(n) + b), \\ c_i^j(n+1) &= c_i^j(n) + \lambda_i \left(z_i^j(n)^2 - c_i^j(n)\, z_i^j(n)^2 - \alpha^{-2}\, c_i^j(n) \right), \quad (1 \le i \le K) \end{aligned}$$

leading to a final $c^{j,\mathrm{adapted}}$ at the end of this period.

5. Another quality measurement run was done identical to the post-cue measurement run, using $c^{j,\mathrm{adapted}}$ (NRMSE results in Figure 38, third column).

Like in the matrix-C-based content-addressing experiments from Section 3.13.3, the recall quality directly after the cue further improved during the autoconceptive adaption afterwards, except for the first pattern. Pending a more detailed investigation, this may be attributed to the "maturation" of the c^j during autoadaption which reveals itself in the convergence of a number of c_i^j to zero during autoadaptation (first and last column in Figure 38). We have seen similar effects in Section 3.13.3.

An obvious difference to those earlier experiments is that the cueing period is much longer now (800 versus 15 – 30 steps). This is owed to the circumstance that now the conceptor adaptation during cueing started from an all-ones c^j, whereas in Section 3.13.3 it was started from a zero C. In the latter case, singular values of C had to grow away from zero toward one during cueing, whereas here they had to sink away from one toward zero. The effects of this mirror situation are not symmetrical. In an "immature" post-cue conceptor matrix C started from a zero C, all the singular values which eventually should converge to zero are already at zero at start time and remain there. Conversely, the post-cue feature projection weights $c^{j,\mathrm{cue}}$, which *should* eventually become zero, have not come close to this destination even after the 800 cue steps that were allotted here (left panels in Figure 38). This tail of "immature" nonzero elements in $c^{j,\mathrm{cue}}$ leads to an insufficient filtering-out of reservoir state components which do not belong to the target pattern dynamics.

The development of the c_i^j during the autonomous post-cue adapation is not monotonous (right panels). Some of these weights meander for a while before they

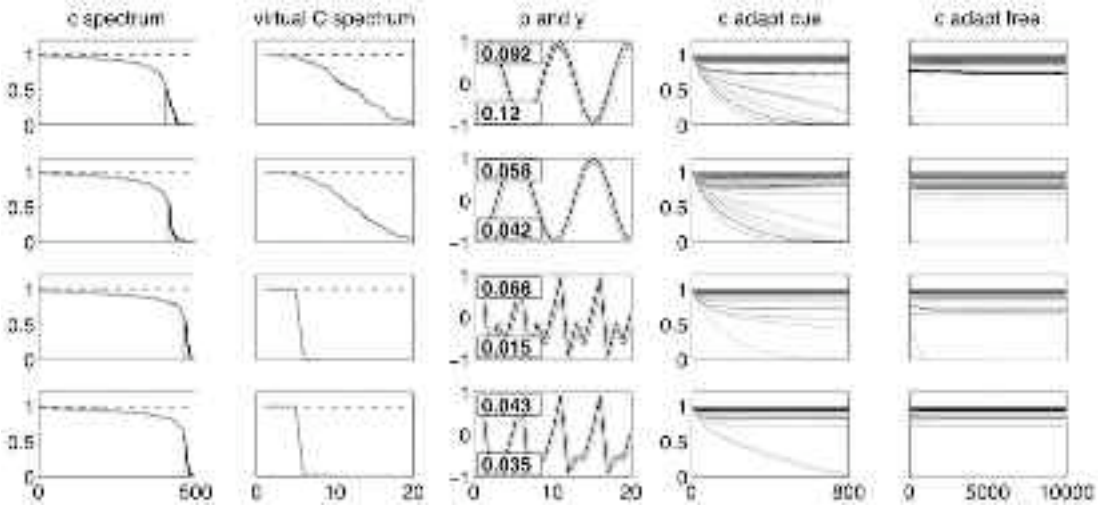

Figure 38: Content-addressed recall using RFC conceptors with $M = 500$ feature vectors f_i. First column: sorted feature projection weight vectors c^j after the cue phase (black) and after 10,000 steps of autoadaptation (gray). Second column: spectra of virtual conceptors C_F after cue (black) and at the end of autonomous adaptation (gray). Both spectra are almost identical. Third column: reconstructed patterns (bold light gray: after cue, bold dark gray: after autoadaptation; the latter are mostly covered by the former) and original patterns (thin black). NRMSEs are given in insets (top: after cue, bottom: after autoadaptation). Fourth column: The adaptation of c^j during the cueing period. Last column: same, during the 10000 autoadaptation steps. 50 of 500 traces are shown. Note the different timescales in column 4 versus column 5. For explanation see text.

settle to what appear stable final values. This is due to the transient nonlinear reservoir–c^j interactions which remain to be mathematically analyzed.

A potentially important advantage of using random feature conceptors c rather than matrix conceptors C in machine learning applications is the faster convergence of the former in online adaptation scenarios. While an dedicated comparison of convergence properties between c and C conceptors remains to be done, one may naturally expect that stochastic gradient descent works more efficiently for random feature conceptors than for matrix conceptors, because the gradient can be followed individually for each coordinate c_i, unencumbered by the second-order curvature interactions which notoriously slow down simple gradient descent in multidimensional systems. This is one of the reasons why in the complex hierarchical signal filtering architecture to be presented below in Section 3.15 I opted for random feature conceptors.

Algebraic and logical rules for conception weights. The various definitions and rules for aperture adaptation, Boolean operations, and abstraction introduced previously for matrix conceptors directly carry over to random feature conceptor.

The new definitions and rules are simpler than for conceptor matrices because they all apply to the individual, scalar conception weights. I present these items without detailed derivations (easy exercises). In the following, let $c = (c_1, \ldots, c_M)', b = (b_1, \ldots, b_M)'$ be two conception weight vectors.

Aperture adaptation (compare Definition 3) becomes

Definition 6

$$\begin{aligned}
\varphi(c_i, \gamma) &:= c_i/(c_i + \gamma^{-2}(1 - c_i)) \quad \textit{for } 0 < \gamma < \infty, \\
\varphi(c_i, 0) &:= \begin{cases} 0 & \textit{if } c_i < 1, \\ 1 & \textit{if } c_i = 1, \end{cases} \\
\varphi(c_i, \infty) &:= \begin{cases} 1 & \textit{if } c_i > 0, \\ 0 & \textit{if } c_i = 0. \end{cases}
\end{aligned}$$

Transferring the matrix-based definition of Boolean operations (Definition 4) to conception weight vectors leads to the following laws:

Definition 7

$$\begin{aligned}
\neg\, c_i &:= 1 - c_i, \\
c_i \wedge b_i &:= \begin{cases} c_i b_i/(c_i + b_i - c_i b_i) & \textit{if not} \quad c_i = b_i = 0, \\ 0 & \textit{if} \quad c_i = b_i = 0, \end{cases} \\
c_i \vee b_i &:= \begin{cases} (c_i + b_i - 2c_i b_i)/(1 - c_i b_i) & \textit{if not} \quad c_i = b_i = 1, \\ 1 & \textit{if} \quad c_i = b_i = 1. \end{cases}
\end{aligned}$$

The matrix-conceptor properties connecting aperture adaptation with Boolen operations (Proposition 10) and the logic laws (Propositions 11, 12) remain valid after the obvious modifications of notation.

We define $c \leq b$ if for all $i = 1, \ldots, M$ it holds that $c_i \leq b_i$. The main elements of Proposition 14 turn into

Proposition 17 *Let $a = (a_1, \ldots, a_M)', b = (b_1, \ldots, b_M)'$ be conception weight vectors. Then the following facts hold.*

1. *If $b \leq a$, then $b = a \wedge c$, where c is the conception weight vector with entries*

$$c_i = \begin{cases} 0 & \textit{if} \quad b_i = 0, \\ (b_i^{-1} - a_i^{-1} + 1)^{-1} & \textit{if} \quad b_i > 0. \end{cases}$$

2. *If $a \leq b$, then $b = a \vee c$, where c is the conception weight vector with entries*

$$c_i = \begin{cases} 1 & \textit{if} \quad b_i = 1, \\ 1 - ((1 - b_i)^{-1} - (1 - a_i)^{-1} + 1)^{-1} & \textit{if} \quad b_i < 1. \end{cases}$$

3. *If $a \wedge c = b$, then $b \leq a$.*

4. If $a \vee c = b$, then $a \leq b$.

Note that all of these definitions and rules can be considered as restrictions of the matrix conceptor items on the special case of diagonal conceptor matrices. The diagonal elements of such diagonal conceptor matrices can be identified with conception weights.

Aspects of biological plausibility. "Biological plausibility" is a vague term inviting abuse. Theoretical neuroscientists develop mathematical or computational models of neural systems which range from fine-grained compartment models of single neurons to abstract flowchart models of cognitive processes. Assessing the methodological role of formal models in neuroscience is a complex and sometimes controversial issue [2, 36]. When I speak of biological plausibility in connection with conceptor models, I do not claim to offer a blueprint that can be directly mapped to biological systems. All that I want to achieve in this section is to show that conceptor systems may be conceived which do *not* have characteristics that are decidedly *not* biologically feasible. In particular, (all) I wanted is a conceptor system variant which can be implemented without state memorizing or weight copying, and which only needs locally available information for its computational operations. In the remainder of this section I explain how these design goals may be satisfied by RFC conceptor architectures.

I will discuss only the adaptation of conception weights and the learning of the input simulation weights D, leaving cueing mechanisms aside. The latter was implemented in the second examples above in an ad-hoc way just to set the stage and would require a separate treatment under the premises of biological plausibility.

In my discussion I will continue to use the discrete-time update dynamics that was used throughout this report. Biological systems are not updated according to a globally clocked cycle, so this clearly departs from biology. Yet, even a synchronous-update discrete-time model can offer relevant insight. The critical issues that I want to illuminate – namely, no state/weight copying and locality of computations – are independent of choosing a discrete or continuous time setting.

I first consider the adaptation of the input simulation weights D. In situated, life-long learning systems one may assume that at given point in (life-)time, some version of D is already present and active, reflecting the system's learning history up to that point. In the content-addressable memory example above, at the end of the cueing period there was an abrupt switch from driving the system with external input to an autonomous dynamics using the system's own input simulation via D. Such binary instantaneous switching can hardly be expected from biological systems. It seems more adequate to consider a gradual blending between the input-driven and the autonomous input simulation mode, as per

$$\begin{aligned} z^j(n+1) &= \\ &= \mathrm{diag}(c^j(n))\, F' \tanh\left(G\, z^j(n) + W^{\mathrm{in}} \left(\tau(n)\, D\, z^j(n) + (1-\tau(n))\, p(n)\right) + b\right), \end{aligned} \tag{84}$$

where a mixing between the two modes is mediated by a "slide ruler" parameter τ which may range between 0 and 1 (a blending of this kind will be used in the architecture in Section 3.15).

As a side remark, I mention that when one considers comprehensive neural architectures, the question of negotiating between an input-driven and an autonomous processing mode arises quite generically. A point in case are "Bayesian brain" models of pattern recognition and control, which currently receive much attention [33, 16]. In those models, a neural processing layer is driven both from "lower" (input-related) layers and from "higher" layers which autonomously generate predictions. Both influences are merged in the target layer by some neural implementation of Bayes' rule. Other approaches that I would like to point out in this context are layered restricted Boltzmann machines [47], which likewise can be regarded as a neural implementation of Bayes' rule; hierarchical neural field models of object recognition [111] which are based on Arathorn's "map seeking circuit" model of combining bottom-up and top-down inputs to a neural processing layer [35]; and mixture of experts models for motor control (for example [107]) where a "responsibility" signal comparable in its function to the τ parameter negotiates a blending of different control signals.

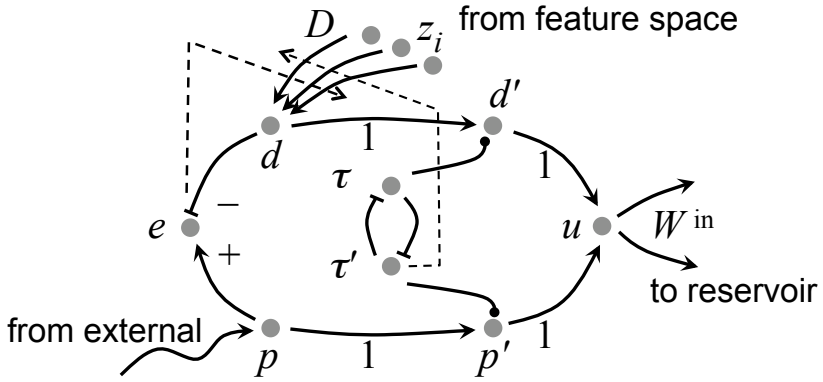

Figure 39: An abstract circuit which would implement the τ negotiation between driving a reservoir with external input p versus with simulated input Dz. Abstract neurons are marked by filled gray circles. Connections that solely copy a neural state forward are marked with "1". Connections marked $-\bullet$ refer to multiplicative modulation. Connections that are inhibitory by their nature are represented by $\dashv$. Broken arrows indicate a controlling influence on the weight adaptation of D. For explanation see text.

Returning to (84), the mathematical formula could be implemented in an abstract neural circuit as drawn in Figure 39. Explanation of this diagram: gray dots represent abstract firing-rate neurons (biologically realized by individual neurons or collectives). All neurons are linear. Activation of neuron d: simulated input $d(n) = Dz(n)$; of p: external driver $p(n)$. Neuron e maintains the value of the

"error" $p(n) - d(n)$. d and p project their activation values to d' and p', whose activations are multiplicatively modulated by the activations of neurons τ and τ'. The latter maintain the values of τ and $\tau' = 1 - \tau$ from (84). The activations $d'(n) = \tau(n)\, Dz(n)$ and $p'(n) = (1 - \tau(n))\, p(n)$ are additively combined in u, which finally feeds to the reservoir through W^{in}.

For a multiplicative modulation of neuronal activity a number of biological mechanisms have been proposed, for example [92, 14]. The abstract model given here is not committed to a specific such mechanism. Likewise I do not further specify the biological mechanism which balances between τ and τ', maintaining a relationship $\tau' = 1 - \tau$; it seems natural to see this as a suitable version of mutual inhibition.

An in-depth discussion by which mechanisms and for which purposes τ is administered is beyond the scope of this report. Many scenarios are conceivable. For the specific purpose of content-addressable memory recall, the setting considered in this section, a natural option to regulate τ would be to identify it with the (0-1-normalized and time-averaged) error signal e. In the architecture presented in Section 3.15 below, regulating τ assumes a key role and will be guided by novel principles.

The sole point that I want to make is that this abstract architecture (or similar ones) requires only local information for the adaptation/learning of D. Consider a synaptic connection D_i from a feature neuron z_i to d. The learning objective (83) can be achieved, for instance, by the stochastic gradient descent mechanism

$$D_i(n+1) = D_i(n) + \lambda \left((p(n) - D_i(n) z_i(n))\, z_i(n) - \alpha^{-2}\, D_i(n) \right), \qquad (85)$$

where the "error" $p(n) - D_i(n) z_i(n)$ is available in the activity of the e neuron. The learning rate λ could be fixed, but a more suggestive option would be to scale it by $\tau'(n) = 1 - \tau(n)$, as indicated in the diagram. That is, D_i would be adapted with an efficacy proportional to the degree that the system is currently being externally driven.

I now turn to the action and the adaptation of the conception weights, stated in mathematical terms in equations (77) and (80). There are a number of possibilities to implement these formulae in a model expressed on the level of abstract firing-rate neurons. I inspect three of them. They are sketched in Figure 40.

The simplest model (Figure 40 **A**) represents the quantity $z_i(n) = c_i(n)\, f_i'\, r(n)$ by the activation of a single neuron φ_i. It receives synaptic input $f_i'\, r(n)$ through connections f_i and feeds to the reservoir (or to an input gating circuit as discussed above) through the single synaptic connection D_i. The weighting of $f_i'\, r(n)$ with the factor c_i is effected by some self-regulated modulation of synaptic gain. Taking into account that c_i changes on a slower timescale than $f_i'\, r(n)$, the information needed to adapt the strength c_i of this modulation (80) is a moving average of the neuron's own activation energy $z_i^2(n)$ and the current synaptic gain $c_i(n)$, which are characteristics of the neuron φ_i itself and thus are trivially locally available.

In the next model (Figure 40 **B**), there is a division of labor between a neuron φ_i which again represents $c_i(n)\, f_i'\, r(n)$ and a preceding neuron ζ_i which represents

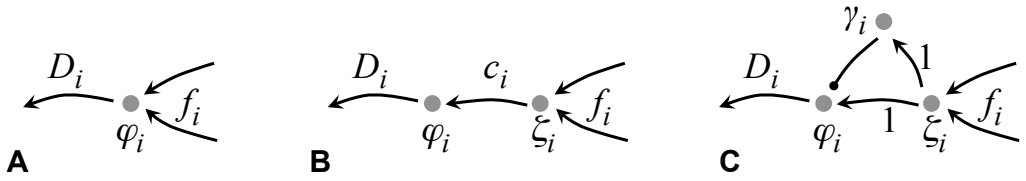

Figure 40: Three candidate neural implementations of conception weight mechanisms. In each diagram, φ_i is an abstract neuron whose activation is $\varphi(n) = z_i(n) = c_i(n)\, f_i'\, r(n)$. In **B** and **C**, ζ_i has activation $f_i'\, r(n)$. In **C**, γ_i has activation c_i. For explanation see text.

$f_i'\, r(n)$. The latter feeds into the former through a single synaptic connection weighted by c_i. The adaptation of the synaptic strength c_i here is based on the (locally time-averaged) squared activity of the postsynaptic neuron φ_i, which again is information locally available at the synaptic link c_i.

Finally, the most involved circuit offered in Figure 40 **C** delegates the representation of c_i to a separate neuron γ_i. Like in the second model, a neuron ζ_i which represents $f_i'\, r(n)$ feeds to φ_i, this time copying its own activation through a unit connection. The γ_i neuron multiplicatively modulates φ_i by its activation c_i. Like in the D adaptation proposal described in Figure 39, I do not commit to a specific biological mechanism for such a multiplicative modulation. The information needed to adapt the activation c_i of neuron γ_i according to (80) is, besides c_i itself, the quantity $z_i = c_i(n)\, f_i'\, r(n)$. The latter is represented in φ_i which is postsynaptic from the perspective of γ_i and therefore not directly accessible. However, the input $f_i'\, r(n)$ from neuron ζ_i is available at γ_i, from which the quantity $z_i = c_i\, f_i'\, r(n)$ can be inferred by neuron γ_i. The neuron γ_i thus needs to instantiate an intricate activation dynamics which combines local temporal averaging of $(f_i'\, r(n))^2$ with an execution of (80). A potential benefit of this third neural circuit over the preceding two is that a representation of c_i by a neural activation can presumably be biologically adapted on a faster timescale than the neuron auto-modulation in system **A** or the synaptic strength adaptation in **B**.

When I first considered content-addressable memories in this report (Section 3.13.3), an important motivation for doing so was that storing entire conceptor matrices C for later use in retrieval is hardly an option for biological systems. This may be different for conception vectors: it indeed becomes possible to "store" conceptors without having to store network-sized objects. Staying with the notation used in Figure 40: a single neuron γ^j might suffice to represent and "store" a conception vector c^j associated with a pattern p^j. The neuron γ^j would project to all φ_i neurons whose states correspond to the signals z_i, with synaptic connection weights c_i^j, and effecting a multiplicative modulation of the activation of the φ_i neurons proportional to these connection weights. I am not in a position to judge whether this is really an option in natural brains. For applications in machine

learning however, using stored conception vectors c^j in conjunction with RFC systems may be a relevant alternative to using stored matrix conceptors, because vectors c^j can be stored much more cheaply in computer systems than matrices.

A speculative outlook. I allow myself to indulge in a brief speculation of how RFC conceptor systems might come to the surface – literally – in mammalian brains. The idea is to interpret the activations of (groups of) neurons in the neocortical sheet as representing conception factors c_i or z_i values, in one of the versions shown in Figure 40 or some other concrete realization of RFC conceptors. The "reservoir" part of RFC systems might be found in deeper brain structures. When some patches of the neocortical sheet are activated and others not (revealed for instance through fMRI imaging or electro-sensitive dyes), this may then be interpreted as a specific c^j vector being active. In geometrical terms, the surface of the hyperellipsoid of the "virtual" conceptor would be mapped to the neocortical sheet. Since this implies a reduction of dimension from a hypothetical reservoir dimension N to the 2-dimensional cortical surface, a dimension folding as in self-organizing feature maps [58, 77, 34] would be necessary. What a cognitive scientist would call an "activation of a concept" would find its neural expression in such an activation of a dimensionally folded ellipsoid pertaining to a "virtual" conceptor C_F in the cortical sheet. An intriguing further step down speculation road is to think about Boolean operations on concepts as being neurally realized through the conceptor operations described in Sections 3.9 – 3.12. All of this is still too vague. Still, some aspects of this picture have already been explored in some detail in other contexts. Specifically, the series of neural models for processing serial cognitive tasks in primate brains developed by Dominey et al. [19, 20] combine a reservoir dynamics located in striatal nuclei with cortical context-providing activation patterns which shares some characteristics with the speculations offered here.

3.15 A Hierarchical Filtering and Classification Architecture

A reservoir equipped with some conceptor mechanism does not by itself serve a purpose. If this computational-dynamical principle is to be made useful for practical purposes like prediction, classification, control or others, or if it is to be used in cognitive systems modeling, conceptor-reservoir modules need to be integrated into more comprehensive architectures. These architectures take care of which data are fed to the conceptor modules, where their output is channelled, how apertures are adapted, and everything else that is needed to manage a conceptor module for the overall system purpose. In this section I present a particular architecture for the purpose of combined signal denoising and classification as an example. This (still simple) example introduces a number of features which may be of more general use when conceptor modules are integrated into architectures:

Arranging conceptor systems in bidirectional hierarchies: a higher concep-

tor module is fed from a lower one by the output of the latter (bottom-up data flow), while at the same time the higher module co-determines the conceptors associated with the lower one (top-down "conceptional bias" control).

Neural instantiations of individual conceptors: Using random feature conceptors, it becomes possible to economically store and address individual conceptors.

Self-regulating balance between perception and action modes: a conceptor-reservoir module is made to run in any mixture of two fundamental processing modes, (i) being passively driven by external input and (ii) actively generating an output pattern. The balance between these modes is autonomously steered by a criterion that arises naturally in hierarchical conceptor systems.

A personal remark: the first and last of these three items constituted the original research questions which ultimately guided me to conceptors.

The task. The input to the system is a timeseries made of alternating sections of the four patterns $p^1, \ldots, p^4$ used variously before in this report: two sines of irrational period lengths, and two slightly differing 5-periodic patterns. This signal is corrupted by strong Gaussian i.i.d. noise (signal-to-noise ratio = 0.5) – see bottom panels in Fig. 42. The task is to classify which of the four patterns is currently active in the input stream, and generate a clean version of it. This task is a simple instance of the generic task "classify and clean a signal that intermittently comes from different, but familiar, sources".

Architecture. The basic idea is to stack copies of a reservoir-conceptor loop, giving a hierarchy of such modules (compare Figure 41). Here I present an example with three layers, having essentially identical modules $\mathcal{M}_{[1]}$ on the lowest, $\mathcal{M}_{[2]}$ on the middle, and $\mathcal{M}_{[3]}$ on the highest layer (I use subscript square brackets $[l]$ to denote levels in the hierarchy).

Each module is a reservoir-conceptor loop. The conceptor is implemented here through the M-dimensional feature space expansion described in the previous section, where a high-dimensional conception weight vector c is multiplied into the feature state (as in Figure 35). At exploitation time the state update equations are

$$\begin{aligned} u_{[l]}(n+1) &= (1 - \tau_{[l-1,l]}(n))\, y_{[l-1]}(n+1) + \tau_{[l-1,l]}(n)\, D\, z_{[l]}(n), \\ r_{[l]}(n+1) &= \tanh(G\, z_{[l]}(n) + W^{\text{in}}\, u_{[l]}(n+1) + b), \\ z_{[l]}(n+1) &= c_{[l]}(n) .* F' r_{[l]}(n+1), \\ y_{[l]}(n+1) &= W^{\text{out}}\, r_{[l]}(n+1), \end{aligned}$$

where $u_{[l]}$ is the effective signal input to module $\mathcal{M}_{[l]}$, $y_{[l]}$ is the output signal from that module, and the τ are mixing parameters which play a crucial role here and will be detailed later. In addition to these fast timescale state updates there are several online adaptation processes, to be described later, which adapt τ's and c's

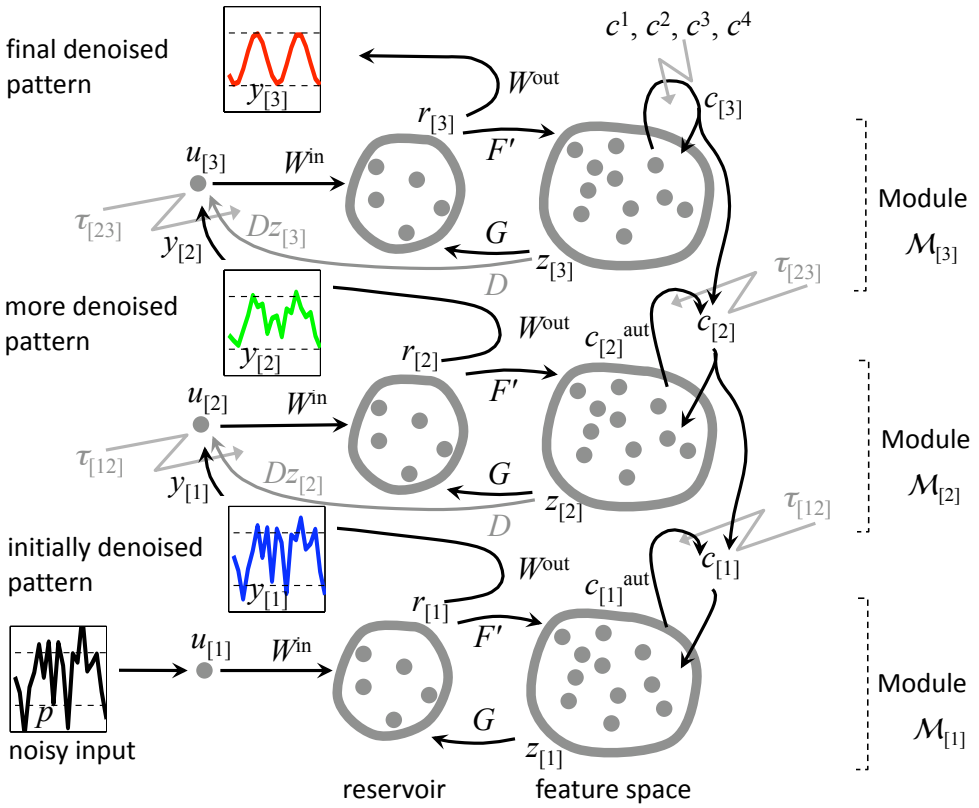

Figure 41: Schematic of 3-layer architecture for signal filtering and classification. For explanation see text.

on slower timescales. The weight matrices $D, G, F, W^{\text{in}}, W^{\text{out}}$ are identical in all modules. F, W^{in} are created randomly at design time and remain unchanged. D and W^{out} are trained on samples of clean "prototype" patterns in an initial pattern loading procedure. G is first created randomly as G^* and then is regularized using white noise (all detail in Section 4.7).

The effective input signal $u_{[l]}(n+1)$ to $\mathcal{M}_{[l]}$ is thus a mixture mediated by a "trust" variable $\tau_{[l-1,l]}$ of a module-external external input $y_{[l-1]}(n+1)$ and a module-internal input simulation signal $D\,z_{[l]}(n)$. On the bottom layer, $\tau_{[01]} \equiv 0$ and $y_{[0]}(n) = p(n)$, that is, this layer has no self-feedback input simulation and is entirely driven by the external input signal $p(n)$. Higher modules receive the output $y_{[l-1]}(n+1)$ of the respective lower module as their external input. Both input mix components $y_{[l-1]}$ and $D\,z_{[l]}$ represent partially denoised versions of the external pattern input. The component $y_{[l-1]}$ from the module below will typically be noisier than the component $Dz_{[l]}$ that is cycled back within the module, because each module is supposed to de-noise the signal further in its internal reservoir-conceptor feedback loop. If $\tau_{[l-1,l]}$ were to be 1, the module would be running in an autonomous pattern generation mode and would be expected to re-generate a very clean version of a stored pattern – which might however be a wrong one. If

$\tau_{[l-1,l]}$ were to be 0, the module would be running in an entirely externally driven mode, with no "cleaning" in effect. It is crucial for the success of this system that these mixing weights $\tau_{[l-1,l]}$ are appropriately set. They reflect a "trust" of the system in its current hypothesis about the type of the driving pattern p, hence I call them *trust* variables. I mention at this point that when the external input p changes from one pattern type to another, the trust variables must quickly decrease in order to temporarily admit the architecture to be in an altogether more input-driven, and less self-generating, mode. All in all this constitutes a bottom-up flow of information, whereby the raw input p is cleaned stage-wise with an amount of cleaning determined by the current trust of the system that it is applying the right conceptor to effect the cleaning.

The output signals $y_{[l]}$ of the three modules are computed from the reservoir states $r_{[l]}$ by output weights W^{out}, which are the same on all layers. These output weights are initially trained in the standard supervised way of reservoir computing to recover the input signal given to the reservoir from the reservoir state. The 3-rd layer output $y_{[3]}$ also is the ultimate output of the entire architecture and should give a largely denoised version of the external driver p.

Besides this bottom-up flow of information there is a top-down flow of information. This top-down pathway affects the conception weight vectors $c_{[l]}$ which are applied in each module. The guiding idea here is that on the highest layer ($l = 3$ in our example), the conceptor $c_{[3]}$ is of the form

$$c_{[3]}(n) = \bigvee_{j=1,\ldots,4} \varphi(c^j, \gamma^j(n)), \tag{86}$$

where $c^1, \ldots, c^4$ are *prototype* conception weight vectors corresponding to the four training patterns. These prototype vectors are computed and stored at training time. In words, at the highest layer the conception weight vector is constrained to be a disjunction of aperture-adapted versions of the prototype conception weight vectors. Imposing this constraint on the highest layer can be regarded as inserting a qualitative bias in the ensuing classification and denoising process. Adapting $c_{[3]}(n)$ amounts to adjusting the aperture adaptation factors $\gamma^j(n)$.

At any time during processing external input, the current composition of $c_{[3]}$ as a γ^j-weighted disjunction of the four prototypes reflects the system's current *hypothesis* about the type of the current external input. This hypothesis is stagewise passed downwards through the lower layers, again mediated by the trust variables.

This top-down pathway is realized as follows. Assume that $c_{[3]}(n)$ has been computed. In each of the two modules below ($l = 1, 2$), an (auto-)conception weight vector $c^{\text{aut}}_{[l]}(n)$ is computed by a module-internal execution of the standard autoconception adaptation described in the previous section (Equation (80)). To arrive at the effective conception weight vector $c_{[l]}(n)$, this $c^{\text{aut}}_{[l]}(n)$ is then blended with the current conception weight vector $c_{[l+1]}(n)$ from the next higher layer, again using the respective trust variable as mixing coefficient:

$$c_{[l]}(n) = (1 - \tau_{[l,l+1]}(n))\, c^{\text{aut}}_{[l]}(n) + \tau_{[l,l+1]}(n)\, c_{[l+1]}(n). \tag{87}$$

In the demo task reported here, the raw input p comes from either of four sources $p^1, \ldots, p^4$ (the familiar two sines and 5-periodic patterns), with additive noise. These four patterns are initially stored in each of the modules $\mathcal{M}_{[l]}$ by training input simulation weights D, as described in Section 3.14. The same D is used in all layers.

In the exploitation phase (after patterns have been loaded into D, output weights have been learnt, and prototype conceptors c^j have been learnt), the architecture is driven with a long input sequence composed of intermittent periods where the current input is chosen from the patterns p^j in turn. While being driven with p^j, the system must autonomously determine which of the four stored pattern is currently driving it, assign trusts to this judgement, and accordingly tune the degree of how strongly the overall processing mode is autonomously generative (high degree of cleaning, high danger of "hallucinating") vs. passively input-driven (weak cleaning, reliable coupling to external driver).

Summing up, the overall functioning of the trained architecture is governed by two pathways of information flow,

- a bottom-up pathway where the external noisy input p is successively denoised,
- a top-down pathway where hypotheses about the current pattern type, expressed in terms of conception weight vectors, are passed downwards,

and by two online adaptation processes,

- adjusting the trust variables $\tau_{[l-1,l]}$, and
- adjusting the conception weight vector $c_{[l]}$ in the top module.

I now describe the two adaptation processes in more detail.

Adapting the trust variables. Before I enter technicalities I want to emphasize that here we are confronted with a fundamental problem of information processing in situated intelligent agents ("SIA": animals, humans, robots). A SIA continuously has to "make sense" of incoming sensor data, by matching them to the agent's learnt/stored concepts. This is a multi-faceted task, which appears in many different instantiations which likely require specialized processing strategies. Examples include online speech understanding, navigation, or visual scene interpretation. For the sake of this discussion I will lump them all together and call them "online data interpretation" (ODI) tasks. ODI tasks variously will involve subtasks like de-noising, figure-ground separation, temporal segmentation, attention control, or novelty detection. The demo architecture described in this section only addresses de-noising and temporal segmentation. In many cognitive architectures in the literature, ODI tasks are addressed by maintaining an online representation of a "current best" interpretation of the input data. This representation is generated in "higher" levels of a processing hierarchy and is used in a top-down fashion to assist lower levels, for instance by way of providing statistical

bias or predictions (discussion in [16]). This top-down information then tunes the processing in lower levels in some way that enables them to extract from their respective bottom-up input specific features while suppressing others – generally speaking, by making them selective. An inherent problem in such architectures is that the agent must not grow overly confident in its top-down pre-conditioning of lower processing layers. In the extreme case of relying entirely on the current interpretation of data (instead of on the input data themselves), the SIA will be hallucinating. Conversely, the SIA will perform poorly when it relies entirely on the input data: then it will not "understand" much of it, becoming unfocussed and overwhelmed by noise. A good example is semantic speech or text understanding. Linguistic research has suggested that fast *forward inferences* are involved which predispose the SIA to interpret next input in terms of a current representation of semantic context (for instance, [96]). As long as this current interpretation is appropriate, it enables fast semantic processing of new input; but when it is inappropriate, it sends the SIA on erroneous tracks which linguists call "garden paths". Generally speaking, for robust ODI an agent should maintain a reliable measure of the degree of trust that the SIA has in its current high-level interpretation. When the trust level is high, the SIA will heavily tune lower levels by higher-level interpretations (top-down dominance), while when trust levels are low, it should operate in a bottom-up dominated mode.

Maintaining an adaptive measure of trust is thus a crucial objective for an SIA. In Bayesian architectures (including Kalman filter observers in control engineering), such a measure of trust is directly provided by the posterior probability $p(\text{interpretation} \mid \text{data})$. A drawback here is that a number of potentially complex probability distributions have to be learnt beforehand and may need extensive training data, especially when prior hyperdistributions have to be learnt instead of being donated by an oracle. In mixture of predictive expert models (for instance [107]), competing interpretation models are evaluated online in parallel and are assigned relative trust levels according to how precisely they can predict the current input. A problem that I see here is computational cost, besides the biological implausibility of executing numerous predictors in parallel. In adaptive resonance theory [43], the role of a trust measure is filled by the ratio between the norm of a top-down pattern interpretation over the norm of an input pattern; the functional effects of this ratio for further processing depends on whether that ratio is less or greater than a certain "vigilance" parameter. Adaptive resonance theory however is primarily a static pattern processing architecture not designed for online processing of temporal data.

Returning to our demo architecture, here is how trust variables are computed. They are based on auxiliary quantities $\delta_{[l]}(n)$ which are computed within each module l. Intuitively, $\delta_{[l]}(n)$ measures the (temporally smoothed) discrepancy between the external input signal fed to the module and the self-generated, conceptor-cleaned version of it. For layers $l > 1$ the external input signal is the bottom-up passed output $y_{[l-1]}$ of the lower layer. The conceptor-cleaned, module-generated

version is the signal $Dz_{[l]}(n)$ extracted from the conception-weighted feature space signal $z_{[l]}(n) = c_{[l]}(n) .* F' r_{[l]}(n)$ by the input simulation weights D, where $r_{[l]}(n)$ is the reservoir state in layer l. Applying exponential smoothing with smoothing rate $\sigma < 1$, and normalizing by the likewise smoothed variance of $y_{[l-1]}$, gives update equations

$$\begin{aligned} \bar{y}_{[l-1]}(n+1) &= \sigma\, \bar{y}_{[l-1]}(n) + (1-\sigma)\, y_{[l-1]}(n+1), \text{ (running average)} \quad (88) \\ \overline{\text{var}}\, y_{[l-1]}(n+1) &= \\ & \sigma\, \overline{\text{var}}\, y_{[l-1]}(n+1) + (1-\sigma)\, (y_{[l-1]}(n+1) - \bar{y}_{[l-1]}(n+1))^2, \quad (89) \\ \delta_{[l]}(n+1) &= \sigma\, \delta_{[l]}(n) + (1-\sigma)\, \frac{(y_{[l-1]}(n+1) - Dz_{[l]}(n+1))^2}{\overline{\text{var}}\, y_{[l-1]}(n+1)} \quad (90) \end{aligned}$$

for the module-internal detected discrepancies $\delta_{[l]}$. In the bottom module $\mathcal{M}_{[1]}$, the same procedure is applied to obtain $\delta_{[1]}$ except that the module input is here the external driver $p(n)$ instead the output $y_{[l-1]}(n)$ from the level below.

From these three discrepancy signals $\delta_{[l]}(n)$ two trust variables $\tau_{[12]}, \tau_{[23]}$ are derived. The intended semantics of $\tau_{[l,l+1]}$ can be stated as "measuring the degree by which the discrepancy is reduced when going upwards from level l to level $l+1$". The rationale behind this is that when the currently active conception weights in modules l and $l+1$ are appropriate for the current drive entering module i from below (or from the outside when $l = 1$), the discrepancy should *decrease* when going from level l to level $l+1$, while if the the currently applied conception weights are the wrong ones, the discrepancy should *increase* when going upwards. The core of measuring trust is thus the difference $\delta_{[l]}(n) - \delta_{[l+1]}(n)$, or rather (since we want the same sensitivity across all levels of absolute values of δ) the difference $log(\delta_{[l]}(n)) - log(\delta_{[l+1]}(n))$. Normalizing this to a range of $(0, 1)$ by applying a logistic sigmoid with steepness $d_{[l,l+1]}$ finally gives

$$\tau_{[l,l+1]}(n) = \left(1 + \left(\frac{\delta_{[l+1]}(n)}{\delta_{[l]}(n)}\right)^{d_{[l,l+1]}}\right)^{-1}. \quad (91)$$

The steepness $d_{[l,l+1]}$ of the trust sigmoid is an important design parameter, which currently I set manually. Stated in intuitive terms it determines how "decisively" the system follows its own trust judgement. It could be rightfully called a "meta-trust" variable, and should itself be adaptive. As will become clear in the demonstrations below, large values of this *decisiveness* leads the system to make fast decisions regarding the type of the current driving input, at an increased risk of settling down prematurely on a wrong decision. Low values of $d_{[l,l+1]}$ allow the system to take more time for making a decision, consolidating information acquired over longer periods of possibly very noisy and only weakly pattern-characteristic input. My current view on the regulation of decisiveness is that it cannot be regulated on the sole basis of the information contained in input data, but reflects higher cognitive capacities (connected to mental attitudes like "doubt", "confidence", or even "stubbornness"...) which are intrinsically not entirely data-dependent.

Adapting the top-level conception weight vectors $c_{[l]}$. For clarity of notation I will omit the level index $[l]$ in what follows, assuming throughout $l = 3$. By equation (86), the effective conception weight vector used in the top module will be constrained to be a disjunction $c(n) = \bigvee_{j=1,\ldots,4} \varphi(c^j, \gamma^j(n))$, where the c^j are prototype conception weight vectors, computed at training time. Adapting $c(n)$ amounts to adjusting the apertures of the disjunctive components c^j via $\gamma^j(n)$. This is done indirectly.

The training of the prototype conception weights (and of the input simulation matrix D and of the readout weights W^{out}) is done with a single module that is driven by the clean patterns p^j. Details of the training procedure are given in the Experiments and Methods Section 4.7. The prototype conception weight vectors can be written as

$$c^j = E[(z^j)^{\cdot\wedge 2}] \mathbin{.*} (E[(z^j)^{\cdot\wedge 2}] + \alpha^{-2})^{\cdot\wedge -1},$$

where $z^j(n) = c^j \mathbin{.*} F' r^j(n)$ is the M-dimensional signal fed back from the feature space to the reservoir while the module is being driven with pattern j during training, and the aperture α is a design parameter. Technically, we do not actually store the c^j but their constitutents α and the corresponding mean signal energy vectors $E[(z^j)^{\cdot\wedge 2}]$, the latter of which are collected in an $M \times 4$ *prototype* matrix

$$P = (E[(z_i^j)^2])_{i=1,\ldots,M;\, j=1,\ldots,4}. \tag{92}$$

I return to the conceptor adaptation dynamics in the top module at exploitation time. Using results from previous sections, equation (86) can be re-written as

$$c(n) = \left(\sum_j (\gamma^j(n))^{\cdot\wedge 2} \mathbin{.*} E[(z^j)^{\cdot\wedge 2}] \right) \mathbin{.*} \left(\sum_j (\gamma^j(n))^{\cdot\wedge 2} \mathbin{.*} E[(z^j)^{\cdot\wedge 2}] + \alpha^{-2} \right)^{\cdot\wedge -1}, \tag{93}$$

where the $+\alpha^{-2}$ operation is applied component-wise to its argument vector. The strategy for adapting the factors $\gamma^j(n)$ is to minimize the loss function

$$\mathcal{L}\{\gamma^1, \ldots, \gamma^4\} = \| \sum_j (\gamma^j)^{\cdot\wedge 2} E[(z^j)^{\cdot\wedge 2}] - E[z^{\cdot\wedge 2}] \|^2, \tag{94}$$

where z is the feature space output signal $z(n) = c(n) \mathbin{.*} F' r(n)$ available during exploitation time in the top module. In words, the adaptation of c aims at finding a weighted disjunction of prototype vectors which optimally matches the currently observed mean energies of the z signal.

It is straightforward to derive a stochastic gradient descent adaptation rule for minimizing the loss (94). Let $\gamma = (\gamma^1, \ldots, \gamma^4)$ be the row vector made from the γ^j, and let $\cdot.^2$ denote element-wise squaring of a vector. Then

$$\gamma(n+1) = \gamma(n) + \lambda_\gamma \left(z(n+1)^{\cdot\wedge 2} - P\,(\gamma'(n))^{\cdot\wedge 2} \right)' P\,\mathrm{diag}(\gamma(n)) \tag{95}$$

implements the stochastic gradient of $\mathcal{L}$ with respect to γ, where λ_γ is an adaptation rate. In fact I do not use this formula as is, but add two helper mechanisms, effectively carrying out

$$\begin{aligned}\gamma^*(n+1) &= \gamma(n) + \\ &\lambda_\gamma \left(\left(z(n+1)^{\cdot\wedge 2} - P\left(\gamma'(n)\right)^{\cdot\wedge 2} \right)' \; P\,\mathrm{diag}(\gamma(n)) + d\left(1/2 - \gamma(n)\right) \right) \quad (96) \\ \gamma(n+1) &= \gamma^*(n+1)/\mathrm{sum}(\gamma^*(n+1)). \quad (97)\end{aligned}$$

The addition of the term $d\,(1/2-\gamma(n))$ pulls the γ^j away from the possible extremes 0 and 1 toward 1/2 with a *drift* force d, which is a design parameter. This is helpful to escape from extreme values (notice that if $\gamma^j(n) = 0$, then γ^j would forever remain trapped at that value in the absence of the drift force). The normalization (97) to a unit sum of the γ^j greatly reduces adaptation jitter. I found both amendments crucial for a reliable performance of the γ adaptation.

Given the $\gamma(n)$ vector, the top-module $c(n)$ is obtained by

$$c(n) = \left(P\left(\gamma'(n)\right)^{\cdot\wedge 2} \right) \; .* \; \left(P\left(\gamma'(n)\right)^{\cdot\wedge 2} + \alpha^{-2} \right)^{\cdot\wedge -1}.$$

Simulation Experiment 1: Online Classification and De-Noising. Please consult Figure 42 for a graphical display of this experiment. Details (training, initalization, parameter settings) are provided in the Experiments and Methods section 4.7.

The trained 3-layer architecture was driven with a 16,000 step input signal composed of four blocks of 4000 steps each. In these blocks, the input was generated from the patterns p^1, p^3, p^2, p^4 in turn (black lines in bottom row of Figure 42), with additive Gaussian noise scaled such that a signal-to-noise ratio of 1/2 was obtained (red lines in bottom row of Figure 42). The 3-layer architecture was run for the 16,000 steps without external intervention.

The evolution of the four γ^j weights in the top layer represent the system's classification hypotheses concerning the type of the current driver (top row in Figure 42). In all four blocks, the correct decision is reached after an initial "rethinking" episode. The trust variable $\tau_{[23]}$ quickly approaches 1 after taking a drop at the block beginnings (green line in fourth row of Figure). This drop allows the system to partially pass through the external driver signal up to the top module, de-stabilizing the hypothesis established in the preceding block. The trust variable $\tau_{[12]}$ (blue line in fourth row) oscillates more irregularly but also takes its steepest drops at the block beginnings.

For diagnostic purposes, $\gamma^j_{[l]}$ weights were also computed on the lower layers $l = 2, 3$, using (96) and (97). These quantities, which were not entering the system's processing, are indicators of the "classification belief states" in lower modules (second and third row in the Figure). A stagewise consolidation of these hypotheses can be observed as one passes upwards through the layers.

The four patterns fall in two natural classes, "sinewaves" and "period-5". Inspecting the top-layer γ^j it can be seen that within each block, the two hypothesis

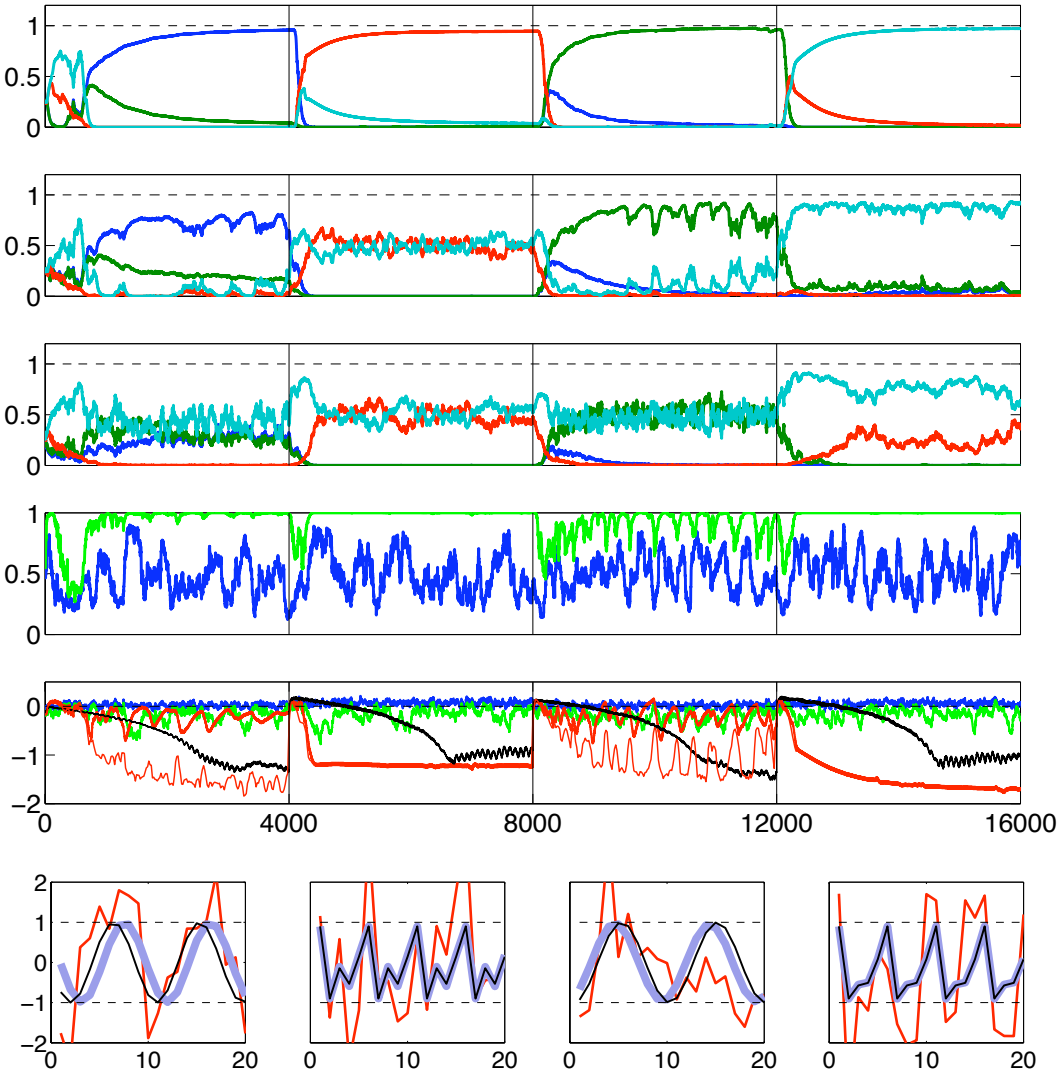

Figure 42: Denoising and classification of prototype patterns. Noisy patterns were given as input in the order p^1, p^3, p^2, p^4 for 4000 timesteps each. Top row: evolution of γ^1 (blue), γ^2 (green), γ^3 (red), γ^4 (cyan) in top layer module. Rows 2 and 3: same in modules 2 and 1. Fourth row: trust variables $\tau_{[12]}$ (blue) and $\tau_{[23]}$ (green). Fifth row: NRMSEs for reconstructed signals $y_{[1]}$ (blue), $y_{[2]}$ (green) and $y_{[3]}$ (red). Black line shows the linear filter reference NRMSE. Thin red line: NRMSE of phase-aligned $y_{[3]}$. The plotting scale is logarithmic base 10. Bottom: pattern reconstruction snapshots from the last 20 timesteps in each pattern presentation block, showing the noisy input (red), the layer-3 output $y_{[3]}$ (thick gray) and the clean signal (thin black). For explanation see text.

indicators associated with the "wrong" class are quickly suppressed to almost zero, while the two indicators of the current driver's class quickly dominate the picture (summing to close to 1) but take a while to level out to their relative final values. Since the top-level $c(n)$ is formally a γ^j-weighted disjunction of the four prototype c^j conception vectors, what happens in the first block (for instance) can also be rephrased as, "after the initial re-thinking, the system is confident that the current driver is p^1 OR p^3, while it is quite sure that it is NOT (p^2 OR p^4)". Another way to look at the same phenomena is to say, "it is easy for the system to quickly decide between the two classes, but it takes more time to distinguish the rather similar patterns within each class".

The fifth row in Figure 42 shows the log10 NRMSE (running smoothed average) of the three module outputs $y_{[l]}(n)$ with respect to a clean version of the driver (thick lines; blue = $y_{[1]}$, green = $y_{[2]}$, red = $y_{[3]}$). For the 5-periodic patterns (blocks 2 and 4) there is a large increase in accuracy from $y_{[1]}$ to $y_{[2]}$ to $y_{[3]}$. For the sinewave patterns this is not the case, especially not in the first block. The reason is that the re-generated sines $y_{[2]}$ and $y_{[3]}$ are not perfectly phase-aligned to the clean version of the driver. This has to be expected because such relative phase shifts are typical for coupled oscillator systems; each module can be regarded as an oscillator. After optimal phase-alignment (details in the Experiments and Methods section), the top-level sine re-generation matches the clean driver very accurately (thin red line). The 5-periodic signal behaves differently in this respect. Mathematically, an 5-periodic discrete-time dynamical system attractor is not an oscillation but a fixed point of the 5-fold iterated map, not admitting anything like a gradual phase shift.

As a baseline reference, I also trained a linear transversal filter (Wiener filter, see [28] for a textbook treatment) on the task of predicting the next clean input value (details in the Experiments and Methods section). The length of this filter was 2600, matching the number of trained parameters in the conceptor architecture (P has $500 * 4$ learnt parameters, D has 500, W^{out} has 100). The smoothed log10 NRMSE of this linear predictor is plotted as a black line. It naturally can reach its best prediction levels only after 2600 steps in each block, much more slowly than the conceptor architecture. Furthermore, the ultimate accuracy is inferior for all four patterns.

Simulation Experiment 2: Tracking Signal Morphs. Our architecture can be characterized as a signal cleaning-and-interpretation systems which guides itselft by allowing top-down hypotheses to make lower processing layers selective. An inherent problem in such systems is that that they may erroneously lock themselves on false hypotheses. Top-down hypotheses are self-reinforcing to a certain degree because they cause lower layers to filter out data components that do not agree with the hypothesis – which is the essence of de-noising after all.

In order to test how our architecture fares with respect to this "self-locking fallacy", I re-ran the simulation with an input sequence that was organized as a linear morph from p^1 to p^2 in the first 4000 steps (linearly ramping up the sine

frequency), then in the next block back to p^1; this was followed by a morph from p^3 to p^4 and back again. The task now is to keep track of the morph mixture in the top-level γ^j. This is a greatly more difficult task than the previous one because the system does not have to just decide between 4 patterns, but has to keep track of minute changes in relative mixtures. The signal-to-noise ratio of the external input was kept at 0.5. The outcome reveals an interesting qualitative difference in how the system copes with the sine morph as opposed to the 5-periodic morph. As can be seen in Figure 43, the highest-layer hypothesis indicators γ^j can track the frequency morph of the sines (albeit with a lag), but get caught in a constant hypothesis for the 5-period morph.

This once again illustrates that irrational-period sines are treated qualitatively differently from integer-periodic signals in conceptor systems. I cannot offer a mathematical analysis, only an intuition. In the sinewave tracking, the overall architecture can be described as a chain of three coupled oscillators, where the bottom oscillator is externally driven by a frequency-ramping sine. In such a driven chain of coupled oscillators, one can expect either chaos, the occurrence of natural harmonics, or frequency-locking across the elements of the chain. Chaos and harmonics are ruled out in our architecture because the prototype conceptors and the loading of two related basic sines prevent it. Only frequency-locking remains as an option, which is indeed what we find. The 5-periodic morph cannot benefit from this oscillation entrainment. The minute differences in shape between the two involved prototypes do not stand out strongly enough from the noise background to induce a noticable decline in the trust variable $\tau_{[23]}$: once established, a single hypothesis persists.

On a side note it is interesting to notice that the linear filter that was used as a baseline cannot at all cope with the frequency sweep, but for the 5-periodic morph it performs as well as in the previous simulation. Both effects can easily be deduced from the nature of such filters.

Only when I used a much cleaner input signal (signal-to-noise ratio of 10), and after the decisiveness d was reduced to 0.5, it became possible for the system to also track the 5-period pattern morph, albeit less precisely than it could track the sines (not shown).

Variants and Extensions. When I first experimented with architectures of the kind proposed above, I computed the module-internal conception weight vectors $c^{\text{aut}}_{[l]}$ (compare Equation (87)) on the two lower levels not via the autoconception mechanism, but in a way that was similar to how I computed the top-level conception weight vector $c_{[3]}$, that is, optimizing a fit to a disjunction of the four prototypes. Abstractly speaking, this meant that a powerful piece of prior information, namely of knowing that the driver was one of the four prototypes, was inserted in all processing layers. This led to a better system performance than what I reported above (especially, faster decisions in the sense of faster convergence of the $\gamma_{[3]}$). However I subsequently renounced this "trick" because the differences in performance were only slight, and from a cognitive modelling per-

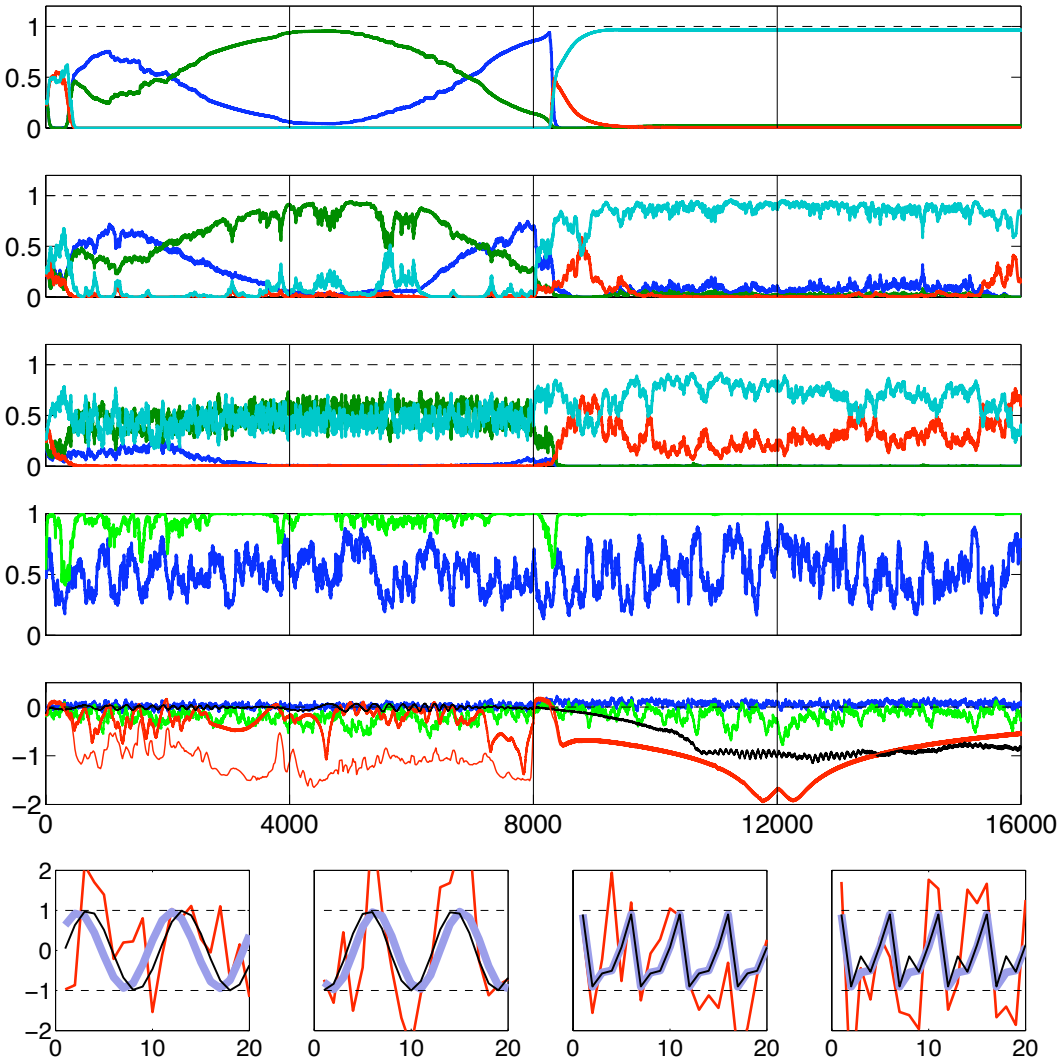

Figure 43: Morph-tracking. First 4000 steps: morphing from p^1 to p^2, back again in next 4000 steps; steps 8000 – 16,000: morphing from p^3 to p^4 and back again. Figure layout same as in Figure 42.

spective I found it more appealing to insert such a valuable prior only in the top layer (motto: the retina does not conceptually understand what it sees).

Inspecting again the top row in Figure 42, one finds fast initial decision between the alternatives "pattern 1 or 2" versus "pattern 3 or 4", followed by a much slower differentation within these two classes. This suggests architecture variants where all layers are informed by priors of the kind as in Equation (86), that is, the local conceptor on a layer is constrained to an aperture-weighted disjunction of a finite number of prototype conceptors. However, the number of prototype conceptors would shrink as one goes upwards in the hierarchy. The reduction in number would be effected by merging several distinct prototype conception vectors $c^{j_1}, \ldots, c^{j_k}$ in layer l into a single prototype vector $c^j = \bigvee\{c^{j_1}, \ldots, c^{j_k}\}$. In terms of classical AI

knowledge representation formalisms this would mean to implement an abstraction hierarchy. A further refinement that suggests itself would be to install a top-down processing pathway by which the current hypothesis on layer $l+1$ selects which finer-grained disjunction of prototypes on layer l is chosen. For instance, when $c_{[l+1]}(n) = c^j$ and $c^j = \bigvee\{c^{j_1}, \ldots, c^{j_k}\}$, then the conception weight vector $c_{[l]}(n)$ is constrained to be of the form $\bigvee_{i=1,\ldots,k} \varphi(c^{j_i}, \gamma^{j_i}_{[l]}(n))$. This remains for future work.

The architecture presented above is replete with ad-hoc design decisions. Numerous details could have been realized differently. There is no unified theory which could inform a system designer what are the "right" design decisions. A complete SIA architecture must provide a plethora of dynamical mechanisms for learning, adaptation, stabilization, control, attention and so forth, and each of them in multiple versions tailored to different subtasks and temporal scales. I do not see even a theoretical possibility for an overarching, principled theory which could afford us with rigorous design principles for all of these. The hierarchical conceptor architecture presented here is far from realizing a complete SIA system, but repercussions of that under-constrainedness of design already show.

Discussion. Hierarchical neural learning architectures for pattern recognition have been proposed in many variants (examples: [66, 38, 33, 47, 35, 111]), albeit almost always for static patterns. The only example of hierarchical neural architectures for temporal pattern recognition that I am aware of are the localist-connectionistic SHRUTI networks for text understanding [96]. Inherently temporal hierarchical pattern classification is however realized in standard hidden-Markov-model (HMM) based models for speech recognition.

There is one common characteristic across all of these hierarchical recognition systems (neural or otherwise, static or temporal). This shared trait is that when one goes upward through the processing layers, increasingly "global" or "coarse-grained" or "compound" features are extracted (for instance, local edge detection in early visual processing leading through several stages to object recognition in the highest layer). While the concrete nature of this layer-wise integration of information differs between approaches, at any rate there is change of represented categories across layers. For the sake of discussion, let me refer to this as the *feature integration* principle.

From the point of view of logic-based knowldege representation, another important trait is shared by hierarchical pattern recognition systems: abstraction. The desired highest-level output is a class labelling of the input data. The recognition architecture has to be able to *generalize* from the particular input instance. This abstraction function is not explicitly implemented in the layers of standard neural (or HMM) recognizers. In rule-based decision-tree classification systems (textbook: [75]), which can also be regarded as hierarchical recognition systems, the hierarchical levels however directly implement a series of class abstractions. I will refer to the abstraction aspect of pattern recognition as the *categorical abstraction* principle.

The conceptor-based architecture presented in this section implements categorical abstraction through the γ variables in the highest layer. They yield (graded) class judgements similar to what is delivered by the class indicator variables in the top layers of typical neural pattern recognizers.

The conceptor-based architecture is different from the typical neural recognition systems in that it does not implement the feature integration principle. As one progresses upwards through the layers, always the same dynamic item is represented, namely, the current periodic pattern, albeit in increasingly denoised versions. I will call this the *pattern integrity* principle. The pattern integrity principle is inherently conflicting with the feature integration principle.

By decades of theoretical research and successful pattern recognition applications, we have become accustomed to the feature integration principle. I want to argue that the pattern integrity principle has some cognitive plausibility and should be considered when one designs SIA architectures.

Consider the example of listening to a familiar piece of music from a CD player in a noisy party environment. The listener is capable of two things. Firstly, s/he can classify the piece of music, for instance by naming the title or the band. This corresponds to performing categorical abstraction, and this is what standard pattern recognition architectures would aim for. But secondly, the listener can also overtly sing (or whistle or hum) along with the melody, or s/he can mentally entrain to the melody. This overt or covert accompaniment has strong de-noising characteristics – party talk fragments are filtered out in the "mental tracing" of the melody. Furthermore, the mental trace is temporally entrained to the source signal, and captures much temporal and dynamical detail (single-note-level of accuracy, stressed versus unstressed beats, etc). That is an indication of pattern integrity.

Another example of pattern integrity: viewing the face of a conversation partner during a person-to-person meeting, say Anne meeting Tom. Throughout the conversation, Anne knows that the visual impression of Tom's face is indeed *Tom*'s face: categorical abstraction is happening. But also, just like a listener to noise-overlaid music can trace online the clean melody, Anne maintains a "clean video" representation of Tom's face as it undergoes a succession of facial expressions and head motions. Anne experiences more of Tom's face than just the top-level abstraction that it is Tom's; and this online experience is entrained to Anne's visual input stream.

Pattern integrity could be said to be realized in standard hierarchical neural architectures to the extent that they are *generative*. Generative models afford of mechanisms by which example instances of a recognition class can be actively produced by the system. Prime examples are the architectures of adaptive resonance theory [43], the Boltzmann Machine [3] and the Restricted Boltzmann Machine / Deep Belief Networks [47]. In systems of this type, the reconstruction of pattern instances occurs (only) in the input layer, which can be made to "confabulate" or "hallucinate" (both terms are used as technical terms in the concerned literature)

pattern instances when primed with the right bias from higher layers.

Projecting such architectures to the human brain (a daring enterprise) and returning to the two examples above, this would correspond to re-generating melodies in the early auditory cortex or facial expressions in the early visual cortex (or even in the retina). But I do not find this a convincing model of what happens in a human brain. Certainly I am not a neuroscientist and not qualified to make scientific claims here. My doubts rest on introspection (forbidden! I know) and on a computational argument. Introspection: when I am mentally humming along with a melody at a party, I still do *hear* the partytalk – I dare say my early auditory modules keep on being excited by the entire auditory input signal. I don't feel like I was hallucinating a clean version of the piece of music, making up an auditory reality that consists only of clean music. But I do not *listen* to the talk noise, I listen only to the music components of the auditory signal. The reconstruction of a clean version of the music happens – as far as I can trust my introspection – "higher up" in my brain's hierarchy, closer to the quarters where consciously controllable cognition resides. The computational argument: generative models, such as the ones mentioned, cannot (in their current versions at least) generate clean versions of noisy input patterns while the input is presented. They either produce a high-level classification response while being exposed to input (bottom-up processing mode), or they generate patterns in their lowest layer while being primed to a particular class in their highest layer (top-down mode). They can't do both at the same time. But humans can: while being exposed to input, a cleaned-up version of the input is being maintained. Furthermore, humans (and the conceptor architecture) can operate in an online-entrained mode when driven by temporal data, while almost all existing recognition architectures in machine learning are designed for static patterns.

Unfortunately I cannot offer a clear definition of "pattern integrity". An aspect of pattern integrity that I find important if not defining is that some temporal and spatial detail of a recognized pattern is preserved across processing layers. Even at the highest layers, a "complete" representation of the pattern should be available. This seems to agree with cognitive theories positing that humans represent concepts by *prototypes*, and more specifically, by *exemplars* (critical discussion [63]). However, these cognitive theories relate to empirical findings on human classification of static, not temporal, patterns. I am aware that I am vague. One reason for this is that we lack a scientific terminology, mathematical models, and standard sets of examples for discussing phenomena connected with pattern integrity. All I can bring to the table at this point is just a new architecture that has some extravagant processing characteristics. This is, I hope, relevant, but it is premature to connect this in any detail to empirical cognitive phenomena.

3.16 Toward a Formal Marriage of Dynamics with Logic

In this subsection I assume a basic acquaintance of the reader with Boolean and first-order predicate logic.

So far, I have established that conceptor matrices can be combined with (almost) Boolean operations, and can be ordered by (a version of) abstraction. In this subsection I explore a way to extend these observations into a formal "conceptor logic".

Before I describe the formal apparatus, I will comment on how I will be understanding the notions of "concept" and "logic". Such a preliminary clarification is necessary because these two terms are interpreted quite differently in different contexts.

"Concepts" in the cognitive sciences. I start with a quote from a recent survey on research on concepts and categories in the cognitive sciences [74]: *"The concept of concepts is difficult to define, but no one doubts that concepts are fundamental to mental life and human communication. Cognitive scientists generally agree that a concept is a mental representation that picks out a set of entities, or a category. That is, concepts* refer*, and what they refer to are categories. It is also commonly assumed that category membership is not arbitrary but rather a principled matter. What goes into a category belongs there by virtue of some law-like regularities. But beyond these sparse facts, the concept CONCEPT is up for grabs."* Within this research tradition, one early strand [85, 18] posited that the overall organization of a human's conceptual representations, his/her *semantic memory*, can be formally well captured by AI representation formalisms called *semantic networks* in later years. In semantic network formalisms, concepts are ordered in abstraction hierarchies, where a more abstract concept refers to a more comprehensive category. In subsequent research this formally clear-cut way of defining and organizing concepts largely dissolved under the impact of multi-faceted empirical findings. Among other things, it turned out that human concepts are graded, adaptive, and depend on features which evolve by learning. Such findings led to a diversity of enriched models of human concepts and their organization (my favourites: [62, 22, 64]), and many fundamental questions remain controversial. Still, across all diversity and dispute, the basic conception of concepts spelled out in the initial quote remains largely intact, namely that concepts are mental representations of categories, and categories are defined *extensionally* as a set of "entities". The nature of these entities is however "up to grabs". For instance, the concept named "Blue" might be referring to the set of blue physical objects, to a set of wavelengths of light, or to a set of sensory experiences, depending on the epistemic approach that is taken.

"Concepts" in logic formalisms. I first note that the word "concept" is not commonly used in logics. However, it is quite clear what elements of logical systems are equated with concepts when such systems are employed as models of semantic memory in cognitive science, or as knowledge representation frameworks in AI. There is a large variety of logic formalisms, but almost all of them employ typed

symbols, specifically unary predicate symbols, relation symbols of higher arity, constant symbols, and function symbols. In the model-theoretic view on logic, such symbols become extensionally *interpreted* by sets of elements defined over the domain set of a set-theoretic model. Unary predicate symbols become interpreted by sets of elements; n-ary relation symbols become interpreted by n-tuples of such elements; function symbols by sets of argument-value pairs; constant symbols by individual elements. A logic *theory* uses a fixed set of such symbols called the theory's *signature*. Within a theory, the interpretation of the signature symbols becomes constrained by the axioms of the theory. In AI knowledge representation systems, this set of axioms can be very large, forming a *world model* and *situation model* (sometimes called "T-Box" and "A-Box"). In the parlance of logic-oriented AI, the extension of unary predicate symbols are often called *classes* instead of "categories".

In AI applications, the world model is often implemented in the structure of a *semantic network* [67], where the classes are represented by nodes labelled by predicate symbols. These nodes are arranged in a hierarchy with more abstract class nodes in higher levels. This allows the computer program to exploit inheritance of properties and relations down the hierarchy, reducing storage requirements and directly enabling many elementary inferences. Class nodes in semantic networks can be laterally linked by relation links, which are labelled by relation symbols. At the bottom of such a hierarchy one may locate *individual* nodes labelled by constant symbols. A cognitive scientist employing such a semantic network representation would consider the class nodes, individual nodes, and relation links as computer implementations or formal models of class concepts, individual concepts, and relational concepts, respectively. Also, semantic network specification languages are sometimes called *concept description languages* in AI programming. On this background, I will understand the symbols contained in a logic signature as names of concepts.

Furthermore, a logical *expression* $\varphi[x_1, \ldots, x_n]$ containing n free (first-order) variables can be interpreted by the set of all n-tuples satisfying this expression. $\varphi[x_1, \ldots, x_n]$ thus defines an n-ary relation. For example, $\varphi[x] = \textsc{Fruit}(x) \wedge \textsc{Yellow}(x) \wedge \textsc{Longish}(x) \wedge \textsc{Curved}(x)$ would represent a class (seems to be the class of bananas). Quite generally, logical expressions formed according to the syntax of a logic formalism can build representations of new concepts from given ones.

There is an important difference between how "concepts" are viewed in cognitive modeling versus logic-based AI. In the latter field, concepts are typically *named* by symbols, and the formal treatment of semantics is based on a reference relationship between the symbols of a signature and their interpretations. However, even in logic-based knowledge representation formalisms there can be un-named concepts which are formally represented as logic expressions with free variables, as for instance the banana formula above. In cognitive science, concepts are not primarily or necessarily named, although a concept can be optionally la-

belled with a name. Cognitive modeling can deal with conceptual systems that have not a single symbol, for instance when modeling animal cognition. By contrast, AI-style logic modeling typically is strongly relying on symbols (the only exception being mathematical theories built on the empty signature; this is of interest only for intra-mathematical investigations).

Remarks on "Logics". In writing this paragraph, I follow the leads of the PhD thesis [86] of Florian Rabe which gives a comprehensive and illuminating account of today's world of formal logic research. The field of mathematical logics has grown and diversified enormously in the last three decades. While formal logic historically has been developed within and for pure mathematics, much of this recent boost was driven by demands from theoretical computer science, AI, and semantic web technologies. This has led to a cosmos populated by a multitude of "logics" which sometimes differ from each other even in basic premises of what, actually, qualifies a formal system as a "logic". In turn, this situation has led to *meta-logical* research, where one develops formal *logical frameworks* in order to systematically categorize and compare different logics.

Among the existing such logical frameworks, I choose the framework of *institutions* [37], because it has been devised as an abstraction of model-theoretic accounts of logics, which allows me to connect quite directly to concepts, categories, and the semantic reference link between these two. Put briefly, a formal system qualifies as a logic within this framework if it can be formulated as an institution. The framework of institutions is quite general: all logics used in AI, linguistics and theoretical cognitive sciences can be characterized as institutions.

The framework of institutions uses tools from category theory. In this section I do not assume that the reader is familiar with category theory, and therefore will give only an intuitive account of how "conceptor logic" can be cast as an institution. A full categorical treatment is given in Section 3.17.

An institution is made of three main components, familiar from the model theory of standard logics like first-order predicate logic:

1. a collection **Sign** of *signatures*, where each signature Σ is a set of *symbols*,

2. for each signature Σ, a set $Sen(\Sigma)$ of Σ-*sentences* that can be formed using the symbols of Σ,

3. again for each signature Σ, a collection $Mod(\Sigma)$ of Σ-*models*, where a Σ-model is a mathematical structure in which the symbols from Σ are interpreted.

Furthermore, for every signature Σ there is a *model relation* $\models_\Sigma \subseteq Mod(\Sigma) \times Sen(\Sigma)$. For a Σ-model m and a Σ-sentence χ, we write infix notation $m \models_\Sigma \chi$ for $(m, \chi) \in \models_\Sigma$, and say "$m$ is a model of χ", with the understanding that the sentence χ makes a true statement about m.

The relationsships between the main elements of an institution can be visualized as in Figure 44.

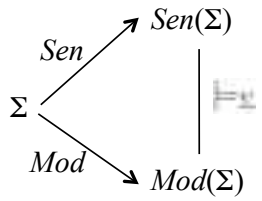

Figure 44: How the elements of an institution relate to each other. For explanation see text.

The full definition of an institution includes a mechanism for symbol re-naming. The intuitive picture is the following. If a mathematician or an AI engineer writes down a set of axioms, expressed as sentences in a logic, the choice of symbols should be of no concern whatsoever. As Hilbert allegedly put it, the mathematical theory of geometry should remain intact if instead of "points, lines, surfaces" one would speak of "tables, chairs, beer mugs". In the framework of institutions this is reflected by formalizing how a signature Σ may be transformed into another signature Σ' by a *signature morphism* $\phi : \Sigma \to \Sigma'$, and how signature morphisms are extended to sentences (by re-naming symbols in a sentence according to the signature morphism) and to models (by interpreting the re-named symbols by the same elements of a model that were previously used for interpreting the original symbols). Then, if m', χ' denote the re-named model m and sentence χ, an institution essentially demands that $m \models_\Sigma \chi$ if and only if $m' \models_{\Sigma'} \chi'$.

For example, first-order logic (FOL) can be cast as an institution by taking for **Sign** the class of all FOL signatures, that is the class of all sets containing typed predicate, relation, function and constant symbols; *Sen* maps a signature Σ to the set of all closed (that is, having no free variables) Σ-expressions (usually called *sentences*); *Mod* assigns to each signature the class of all set-theoretic Σ-structures; and $\models$ is the satisfaction relation of FOL (also called model relation). For another example, Boolean logic can be interpreted as an institution in several ways, for instance by declaring **Sign** as the class of all totally ordered countable sets (the elements of which would be seen as Boolean variables); for each signature Σ of Boolean variables, $Sen(\Sigma)$ is the set of all Boolean expressions $\varphi[X_{i_1}, \ldots, X_{i_n}]$ over Σ and $Mod(\Sigma)$ is the set of all truth value assignments $\tau : \Sigma \to \{T, F\}$ to the Boolean variables in Σ; and $\tau \models_\Sigma \varphi$ if φ evaluates to T under the assignment τ.

In an institution, one can define *logical entailment* between Σ-sentences in the familiar way, by declaring that χ logically entails χ' (where χ, χ' are Σ-sentences) if and only if for all Σ-models m it holds that $m \models_\Sigma \chi$ implies $m \models_\Sigma \chi'$. By a standard abuse of notation, this is also written as $\chi \models_\Sigma \chi'$ or $\chi \models \chi'$.

I will sketch two entirely different approaches to define a "conceptor logic". The first follows in the footsteps of familiar logics. Conceptors can be named by arbitrary symbols, sentences are built by an inductive procedure which specifies how more complex sentences can be constructed from simpler ones by similar syntax rules as in first-order logic, and models are designated as certain mathematical

structures built up from named conceptors. This leads to a logic that essentially represents a version of first-order logic constrained to conceptor domains. It would be a logic useful for mathematicians to investigate "logical" characteristics of conceptor mathematics, especially whether there are complete calculi that allow one to systematically prove all true facts concerning conceptors. I call such logics *extrinsic conceptor logics*. Extrinsic conceptor logics are tools for mathematicians to reason *about* conceptors. A particular extrinsic conceptor logic as an institution is detailed in Section 3.17.

The other approach aims at a conceptor logic that, instead of being a tool for mathematicians to reason *about* conceptors, is a model of how a situated intelligent agent does "logical reasoning" *with* conceptors. I call this *intrinsic conceptor logic* (ICL). An ICL has a number of unconventional properties:

- An ICL should function as a model of a situated agent's conceptor-based information processing. Agents are bound to differ widely in their structure and their concrete lifetime learning histories. Therefore I do not attempt to design a general "fits-all-agents" ICL. Instead, for every single, concrete agent life history there will be an ICL, *the* private ICL of *that* agent life.

- An agent with a personal learning history is bound to develop its private "logic" over time. The ICL of an agent life thus becomes a dynamical system in its own right. The framework of institutions was not intended by its designers to model temporally evolving objects. Specifying an institution such that it can be considered a dynamical system leads to some particularly unconventional characteristics of an agent life ICL. Specifically, signatures become time-varying objects, and signature morphisms (recall that these model the "renaming" of symbols) are used to capture the temporal evolution of signatures.

An agent's lifetime ICL is formalized differently according to whether the agent is based on matrix conceptors or random feature conceptors. Here I work out only the second case.

In the following outline I use the concrete three-layer de-noising and classification architecture from Section 3.15 as a reference example to fill the abstract components of ICL with life. Even more concretely, I use the specific "lifetime history" of the 16000-step adaptation run illustrated in Figure 42 as demonstration example. For simplicity I will refer to that particular de-noising and classification architecture run as "DCA".

Here is a simplified sketch of the main components of an agent's lifetime ICL (full treatment in Section 3.17):

1. An ICL is designed to model a particular agent lifetime history. A specification of such an ICL requires that a formal model of such an *agent life* is available beforehand. The core part of an agent life model $\mathcal{AL}$ is a set of

m conceptor adaptation sequences $\{a_1(n), \ldots, a_m(n)\}$, where each $a_i(n)$ is an M-dimensional conception weight vector. It is up to the modeler's discretion which conceptors in a modeled agent become included in the agent life model $\mathcal{AL}$. In the DCA example I choose the four prototype conception weight vectors $c^1, \ldots, c^4$ and the two auto-adapted $c^{\text{aut}}_{[l]}$ on layers $l = 1, 2$. In this example, the core constitutent of the agent life model $\mathcal{AL}$ is thus the set of $m = 6$ conceptor adaptation trajectories $c^1(n), \ldots, c^4(n), c^{\text{aut}}_{[1]}(n), c^{\text{aut}}_{[2]}(n)$, where $1 \leq n \leq 16000$. The first four trajectories $c^1(n), \ldots, c^4(n)$ are constant over time because these prototype conceptors are not adapted; the last two evolve over time. Another part of an agent life model is the *lifetime* T, which is just the interval of timepoints n for which the adaptation sequences $a_i(n)$ are defined. In the DCA example, $T = (1, 2, \ldots, 16000)$.

2. A signature is a finite non-empty set $\Sigma^{(n)} = \{A^{(n)}_1, \ldots, A^{(n)}_m\}$ of m time-indexed symbols A_i. For every $n \in T$ there is a signature $\Sigma^{(n)}$.

   DCA example: In the ICL of this example agent life, the collection **Sign** of signatures is made of 16000 signatures $\{C^{(n)}_1, \ldots, C^{(n)}_4, A^{(n)}_1, A^{(n)}_2\}$ containing six symbols each, with the understanding that the first four symbols refer to the prototype conceptors $c^1(n), \ldots, c^4(n)$ and the last two refer to the auto-adapted conceptors $c^{\text{aut}}_{[1]}(n), c^{\text{aut}}_{[2]}(n)$.

3. For every pair $\Sigma^{(n+k)}, \Sigma^{(n)}$ of signatures, where $k \geq 0$, there is a signature morphism $\phi^{(n+k,n)} : \Sigma^{(n+k)} \to \Sigma^{(n)}$ which maps $A^{(n+k)}_i$ to $A^{(n)}_i$. These signature morphisms introduce a time arrow into **Sign**. This time arrow "points backwards", leading from later times $n + k$ to earlier times n. There is a good reason for this backward direction. Logic is all about describing facts. In a historically evolving system, facts $\chi^{(n+k)}$ established at some later time $n + k$ can be explained in terms of facts $\zeta^{(n)}$ at preceding times n, but not vice versa. Motto: "the future can be explained in terms of the past, but the past cannot be reduced to facts from the future". Signature morphisms are a technical vehicle to re-formulate descriptions of facts. They must point backwards in time in order to allow facts at later times to become re-expressed in terms of facts stated for earlier times. Figure 45 illustrates the signatures and their morphisms in an ICL.

4. Given a signature $\Sigma^{(n)} = \{A^{(n)}_1, \ldots, A^{(n)}_m\}$, the set of sentences $Sen(\Sigma^{(n)})$ which can be expressed with the symbols of this signature is the set of syntactic expressions defined inductively by the following rules (incomplete, full treatment in next subsection):

   (a) $A^{(n)}_1, \ldots, A^{(n)}_m$ are sentences in $Sen(\Sigma^{(n)})$.

   (b) For $k \geq 0$ such that $n, n + k \in T$, for $A^{(n)}_i \in \Sigma^{(n)}$, $\delta^{(n)}_k A^{(n)}_i$ is in $Sen(\Sigma^{(n)})$.

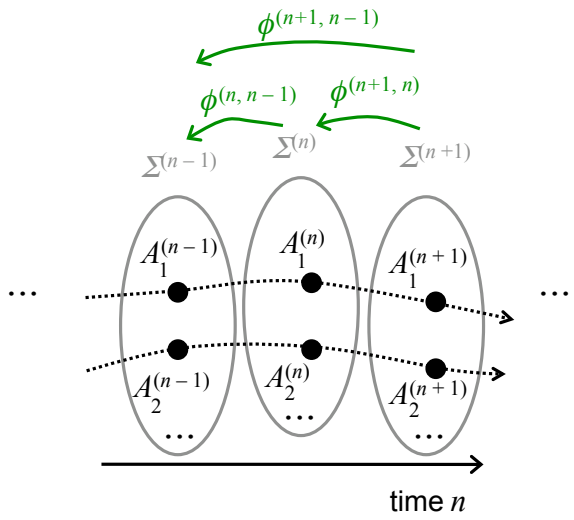

Figure 45: Signatures and their morphisms in an ICL (schematic). For explanation see text.

(c) If $\zeta, \xi \in Sen(\Sigma^{(n)})$, then $(\zeta \vee \xi), (\zeta \wedge \xi), \neg\zeta \in Sen(\Sigma^{(n)})$.

(d) If $\zeta \in Sen(\Sigma^{(n)})$, then $\varphi(\zeta, \gamma) \in Sen(\Sigma^{(n)})$ for every $\gamma \in [0, \infty]$ (this captures aperture adaptation).

(e) If $\zeta, \xi \in Sen(\Sigma^{(n)})$ and $0 \leq b \leq 1$, then $\beta_b(\zeta, \xi) \in Sen(\Sigma^{(n)})$ (this will take care of linear blends $b\zeta + (1-b)\xi$).

In words, sentences express how new conceptors can be built from existing ones by Boolean operations, aperture adaptation, and linear blends. The "seed" set for these inductive constructions is provided by the conceptors that can be directly identified by the symbols in $\Sigma^{(n)}$.

The sentences of form $\delta_k^{(n)} A_i^{(n)}$ deserve a special comment. The operators $\delta_k^{(n)}$ are time evolution operators. A sentence $\delta_k^{(n)} A_i^{(n)}$ will be made to refer to the conceptor version $a_i(n+k)$ at time $n+k$ which has evolved from $a_i(n)$.

5. For every time n, the set $Mod(\Sigma^{(n)})$ of $\Sigma^{(n)}$-models is the set $\mathbf{Z}$ of M-dimensional nonnegative vectors.

   Remarks: (i) The idea for these models is that they represent mean energy vectors $E[z^{\cdot\wedge\cdot,2}]$ of feature space states. (ii) The set of models $Mod(\Sigma^{(n)})$ is the same for every signature $\Sigma^{(n)}$.

   DCA example: Such feature space signal energy vectors occur at various places in the DCA, for instance in Equations (92), (93), and conception weight vectors which appear in the DCA evolution are all defined or adapted in one way or other on the basis of such feature space signal energy vectors.

6. Every $\Sigma^{(n)}$-sentence χ is associated with a concrete conception weight vector $\iota(\chi)$ by means of the following inductive definition:

   (a) $\iota(A_i^{(n)}) = a_i(n)$.

   (b) $\iota(\delta_k^{(n)} A_i^{(n)}) = a_i(n+k)$.

   (c) Case $\chi = (\zeta \vee \xi)$: $\iota(\chi) = \iota(\zeta) \vee \iota(\xi)$ (compare Definition 7).

   (d) Case $\chi = (\zeta \wedge \xi)$: $\iota(\chi) = \iota(\zeta) \wedge \iota(\xi)$.

   (e) Case $\chi = \neg\zeta$: $\iota(\chi) = \neg\iota(\zeta)$.

   (f) Case $\chi = \varphi(\zeta, \gamma)$: $\iota(\chi) = \varphi(\iota(\zeta), \gamma)$ (compare Definition 6).

   (g) Case $\chi = \beta_b(\zeta, \xi)$: $\iota(\chi) = b\,\iota(\zeta) + (1-b)\iota(\xi)$.

   Remark: This statement of the interpretation operator ι is suggestive only. The rigorous definition (given in the next section) involves additional non-trivial mechanisms to establish the connection between the symbol $A_i^{(n)}$ and the concrete conceptor version $a_i(n)$ in the agent life. Here I simply appeal to the reader's understanding that symbol A_i refers to object a_i.

7. For $z^{\cdot\wedge 2} \in \mathbf{Z}$ and $\chi \in Sen(\Sigma^{(n)})$, the model relationship is defined by

$$z^{\cdot\wedge 2} \models_{\Sigma^{(n)}} \chi \quad \text{iff} \quad z^{\cdot\wedge 2} .* (z^{\cdot\wedge 2} + 1)^{\cdot\wedge -1} \leq \iota(\chi). \tag{98}$$

   Remark: This definition in essence just repeats how a conception weight vector is derived from a feature space signal energy vector.

When all category-theoretical details are filled in which I have omitted here, one obtains a formal definition of an institution which represents *the* ICL of an agent life $\mathcal{AL}$. It can be shown that in an ICL, for all $\zeta, \xi \in Sen(\Sigma^{(n)})$ it holds that

$$\zeta \models_{\Sigma^{(n)}} \xi \quad \text{iff} \quad \iota(\zeta) \leq \iota(\xi).$$

By virtue of this fact, logical entailment becomes *decidable* in an ICL: if one wishes to determine whether ξ is implied by ζ, one can effectively compute the vectors $\iota(\zeta), \iota(\xi)$ and then check in constant time whether $\iota(\zeta) \leq \iota(\xi)$.

Returning to the DCA example (with lifetime history shown in Figure 42), its ICL identiefies over time the four prototype conceptors $c^1, \ldots, c^4$ and the two auto-adapted conceptors $c_{[1]}^{\text{auto}}, c_{[2]}^{\text{auto}}$ by temporally evolving symbols $\{C_1^{(n)}, \ldots, C_4^{(n)}, A_1^{(n)}, A_2^{(n)}\}$. All other conceptors that are computed in this architecture can be defined in terms of these six ones. For instance, the top-level conceptor $c_{[3]}(n)$ can be expressed in terms of the identifiable four prototype conceptors by $c_{[3]}(n) = \bigvee_{j=1,\ldots,4} \varphi(c^j, \gamma^j(n))$ by combining the operations of disjunction and aperture adaptation. In ICL syntax this construction would be expressible by a $\Sigma^{(n)}$ sentence, for instance by

$$(((\varphi(C_1^{(n)}, \gamma^1(n)) \vee \varphi(C_2^{(n)}, \gamma^2(n))) \vee \varphi(C_3^{(n)}, \gamma^3(n))) \vee \varphi(C_4^{(n)}, \gamma^4(n))).$$

A typical adaptation objective of a conception vector $c(n)$ occurring in an agent life is to minimize a loss of the form (see Definition 78)

$$E_z[\|z - c(n) .* z\|^2] + \alpha^{-2} \|c(n)\|^2,$$

or equivalently, the objective is to converge to

$$c(n) = E[\alpha^2 z^{\cdot\wedge 2}] .* (E[\alpha^2 z^{\cdot\wedge 2}] + 1)^{\cdot\wedge -1}.$$

This can be re-expressed in ICL terminology as "adapt $c(n)$ such that $\alpha^2 E[z^{\cdot\wedge 2}] \models_{\Sigma^{(n)}} \chi_{c(n)}$, and such that not $z^{\cdot\wedge 2} \models_{\Sigma^{(n)}} \chi_{c(n)}$ for any $z^{\cdot\wedge 2} > \alpha^2 E[z^{\cdot\wedge 2}]$" (here $\chi_{c(n)}$ is an adhoc notation for an ICL sentence $\chi_{c(n)} \in Sen(\Sigma^{(n)})$ specifying $c(n)$). In more abstract terms, the typical adaptation of random feature conceptors can be understood as an attempt to converge toward the conceptor that is maximally $\models$-specific under a certain constraint.

Discussion. I started this section by a rehearsal of how the notion of "concept" is understood in cognitive science and logic-based AI formalisms. According to this understanding, a concept *refers* to a category (terminology of cognitive science); or a class symbol or logical expression with free variables is *interpreted by* its set-theoretical extension (logic terminology). Usually, but not necessarily, the concepts/logical expressions are regarded as belonging to an "ontological" domain that is different from the domain of their respective referents. For instance, consider a human maintaining a concept named cow in his/her mind. Then many cognitive scientists would identifiy the category that is referred to by this concept with the some set of physical cows. Similarly, an AI expert system set up as a farm management system would contain a symbol cow in its signature, and this symbol would be deemed to refer to a collection of physical cows. In both cases, the concept / symbolic expression cow is ontologically different from a set of physical cows. However, both in cognitive science and AI, concepts / symbolic expressions are sometimes brought together with their referents much more closely. In some perspectives taken in cognitive science, concepts are posited to refer to other mental items, for instance to sensory perceptions. In most current AI proof calculi ("inference engines"), models of symbolic expressions are created which are assembled not from external physical objects but from symbolic expressions ("Herbrand universe" constructions). Symbols from a signature Σ then refer to sets of Σ-terms. In sum, fixing the ontological nature of referents is ultimately left to the modeling scientist in cognitive science or AI.

In contrast, ICL is committed to one particular view on the semantic relationship: $\Sigma^{(n)}$-sentences are always describing conception weight vectors, and refer to neural activation energy vectors $z^{\cdot\wedge 2}$. In the case of matrix conceptor based agents, $\Sigma^{(n)}$-sentences describe conceptor matrices and refer to neural activation correlation matrices R by the following variant of (98):

$$R \models_{\Sigma^{(n)}} \chi \quad \text{iff} \quad R(R+I)^{-1} \leq \iota(\chi). \tag{99}$$

In Figure 46 I try to visualize this difference between the classical, extensional view on symbols and their referents, and the view adopted by ICL. This figure contrasts how classical logicians and cognitive scientists would usually model an agent's representation of farm livestock, as opposed to how ICL renders that situation. The semantic relation is here established between the physical world on the one side, and symbols and logical expressions on the other side. The world is idealized as a set of individuals (individual animals in this example), and symbols for concepts (predicate symbols in logic) are semantically interpreted by sets of individuals. In the farmlife example, a logician might introduce a symbol lifestock which would denote the set of all economically relevant animals grown in farms, and one might introduce another symbol poultry to denote the subset of all feathered such animals. The operator that creates "meaning" for concept symbols is the grouping of individuals into sets (the bold "{ }" in Figure 46).

With conceptors, the semantic relation connects neural activity patterns triggered by perceiving animals on the one side, with conceptors acting on neural dynamics on the other side. The core operator that creates meaning is the condensation of the incoming data into a neural activation energy pattern $z^{\cdot\wedge 2}$ (or correlation matrix R for matrix conceptors) from which conceptors are generated via the fundamental construction $c = E[z^{\cdot\wedge 2}] \,.*\, (E[z^{\cdot\wedge 2}]+1)^{\cdot\wedge -1}$ or $C = R(R+I)^{-1}$ (Figure 46 depicts the latter case).

ICL, as presented here, cannot claim to be a model of all "logical reasoning" in a neural agent. Specifically, humans sometimes engage in reasoning activities which are very similar to how syntactic logic calculi are executed in automated theorem proving. Such activities include the build-up and traversal of search trees, creating and testing hypotheses, variable binding and renaming, and more. A standard example is the step-by-step exploration of move options done by a human chess novice. ICL is not designed to capture such conscious combinatorial logical reasoning. Rather, ICL is intended to capture the automated aspects of neural information processing of a situated agent, where incoming (sensor) information is immediately transformed into perceptions and maybe situation representations in a tight dynamical coupling with the external driving signals.

The material presented in this and the next section is purely theoretical and offers no computational add-on benefits over the material presented in earlier sections. There are three reasons why nonetheless I invested the effort of defininig ICLs:

- By casting conceptor logic rigorously as an institution, I wanted to substantiate my claim that conceptors are "logical" in nature, beyond a mere appeal to the intuition that anything admitting Boolean operations is logic.

- The institutional definition given here provides a consistent formal picture of the semantics of conceptors. A conceptor c identified by an ICL sentence χ_c "means" neural activation energy vectors $z^{\cdot\wedge 2}$. Conceptors and their meanings are both neural objects of the same mathematical format, M-

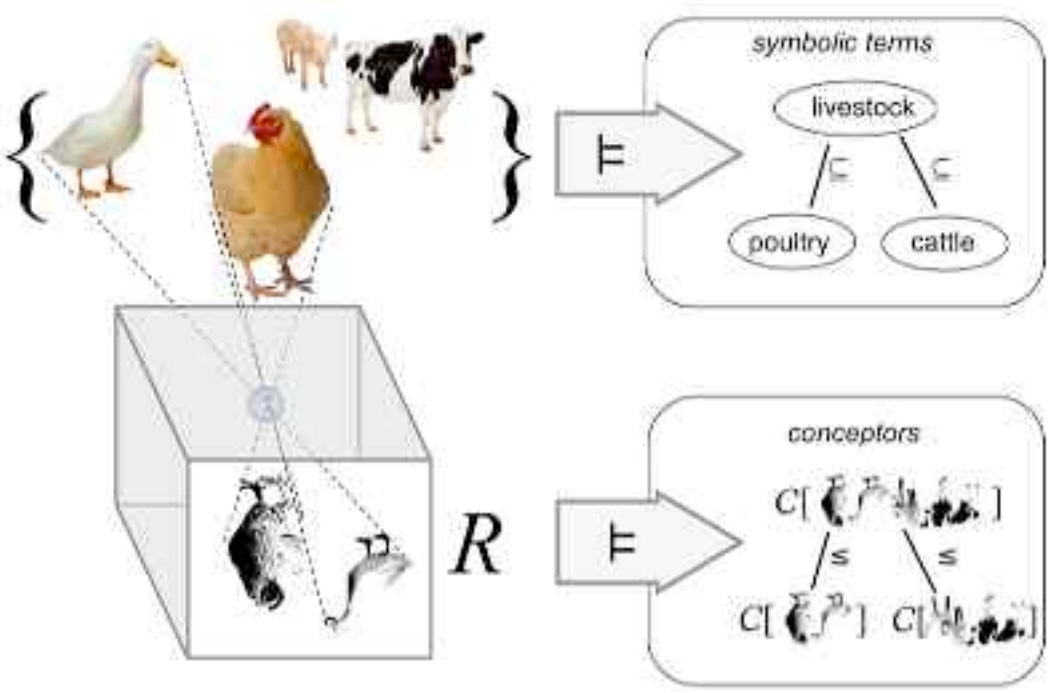

Figure 46: Contrasting the extensional semantics of classical knowledge representation formalisms (upper half of graphics) with the system-internal neurodynamical semantics of conceptors (lower half). In both modeling approaches, abstraction hierarchies of "concepts" arise. For explanation see text.

dimensional nonnegative vectors. Having a clear view on this circumstance helps to relate conceptors to the notions of concepts and their referents, which are so far from being fully understood in the cognitive sciences.

- Some of the design ideas that went into casting ICLs as institutions may be of more general interest for mathematical logic research. Specifically, making signatures to evolve over time – and hence, turn an institution into a dynamical system – might be found a mechanism worth considering in scenarios, unconnected with conceptor theory or neural networks, where one wants to analyse complex dynamical systems by means of formal logics.

3.17 Conceptor Logic as Institutions: Category-Theoretical Detail

In this section I provide a formal specification of conceptor logic as an institution. This section addresses only readers with a dedicated interest in formal logic. I assume that the reader is familiar with the institution framework for representing logics (introduced in [37] and explained in much more detail in Section 2 in [86]) and with basic elements of category theory. I first repeat almost verbatim the

categorical definition of an institution from [37].

Definition 8 *An* institution $\mathcal{I}$ *consists of*

1. *a category* **Sign**, *whose objects* Σ *are called* signatures *and whose arrows are called* signature morphisms,
2. *a functor* $Sen : \mathbf{Sign} \to \mathbf{Set}$, *giving for each signature a set whose elements are called* sentences *over that signature,*
3. *a functor* $Mod : \mathbf{Sign} \to \mathbf{Cat}^{\mathrm{op}}$, *giving for each signature* Σ *a category* $Mod(\Sigma)$ *whose objects are called* Σ-models, *and whose arrows are called* Σ-*(model)* morphisms, *and*
4. *a relation* $\models_\Sigma \subseteq Mod(\Sigma) \times Sen(\Sigma)$ *for each* $\Sigma \in \mathbf{Sign}$, *called* Σ-satisfaction,

such that for each morphism $\phi : \Sigma_1 \to \Sigma_2$ *in* **Sign**, *the* Satisfaction Condition

$$m_2 \models_{\Sigma_2} Sen(\phi)(\chi_1) \quad \textit{iff} \quad Mod(\phi)(m_2) \models_{\Sigma_1} \chi_1 \tag{100}$$

holds for each $m_2 \in Mod(\Sigma_2)$ *and each* $\chi_1 \in Sen(\Sigma_1)$.

The interrelations of these items are visualized in Figure 47.

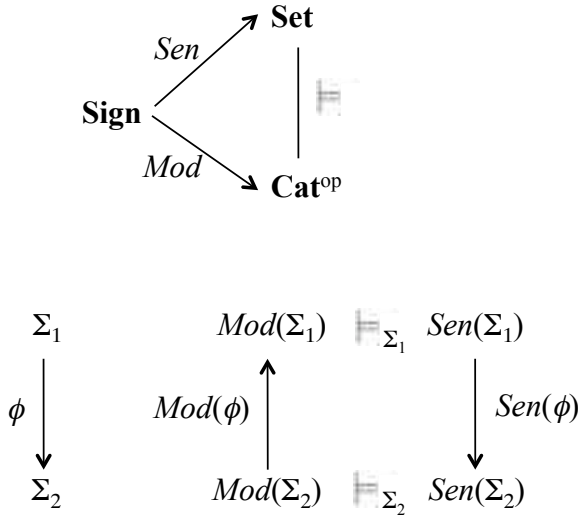

Figure 47: Relationships between the constituents of an institution (redrawn from [37]).

Remarks:

1. The morphisms in **Sign** are the categorical model of re-naming the symbols in a logic. The essence of the entire apparatus given in Definition 8 is to capture the condition that the model-theoretic semantics of a logic is invariant to renamings of symbols, or, as Goguen and Burstall state it, "Truth is invariant under change of notation".

2. The intuition behind Σ-model-morphisms, that is, maps $\mu : I_1^\Sigma \to I_2^\Sigma$, where I_1^Σ, I_2^Σ are two Σ-models, is that μ is an embedding of I_1^Σ in I_2^Σ. If we take first-order logic as an example, with I_1^Σ, I_2^Σ being two Σ-structures, then $\mu : I_1^\Sigma \to I_2^\Sigma$ would be a map from the domain of I_1^Σ to the domain of I_2^Σ which preserves functional and relational relationships specified under the interpretations of I_1^Σ and I_2^Σ.

3. In their original 1992 paper [37], the authors show how a number of standard logics can be represented as institutions. In the time that has passed since then, institutions have become an important "workhorse" for software specification in computer science and for semantic knowledge management systems in AI, especially for managing mathematical knowledge, and several families of programming toolboxes have been built on institutions (overview in [86]). Alongside with the model-theoretic spirit of institutions, this proven usefulness of institutions has motivated me to adopt them as a logical framework for conceptor logic.

Logical entailment between sentences is defined in institutions in the traditional way:

Definition 9 *Let $\chi_1, \chi_2 \in Sen(\Sigma)$. Then χ_1* entails *χ_2, written $\chi_1 \models_\Sigma \chi_2$, if for all $m \in Ob(Mod(\Sigma))$ it holds that $m \models_\Sigma \chi_1 \ \to m \models_\Sigma \chi_2$.*

Institutions are flexible and offering many ways for defining logics. I will frame two entirely different kinds of conceptor logics. The first kind follows the intuitions behind the familiar first-order predicate logic, and should function as a formal tool for mathematicians to reason about (and with) conceptors. Since it looks at conceptors "from the outside" I will call it *extrinsic conceptor logic* (ECL). Although ECL follows the footsteps of familiar logics in many respects, in some aspects it deviates from tradition. The other kind aims at modeling the "logical" operations that an intelligent neural agent can perform whose "brain" implements conceptors. I find this the more interesting formalization; certainly it is the more exotic one. I will call it *intrinsic conceptor logic* (ICL).

Extrinsic conceptor logic. I first give an intuitive outline. I treat only the case of matrix-based conceptors. An ECL concerns conceptors of a fixed dimension N and their logical interrelationships, so one should more precisely speak of N-dimensional ECL. I assume some N is fixed. Sentences of ECL should enable a mathematician to talk about conceptors in a similar way as familiar predicate logics allow a mathematician to describe facts about other mathematical objects. For example, "for all conceptors X, Y it holds that $X \wedge Y \leq X$ and $X \wedge Y \leq Y$" should be formalizable as an ECL sentence. A little notational hurdle arises because Boolean operations appear in two roles: as operators acting on conceptors (the "$\wedge$" in the sentence above), and as constitutents of the logic language (the "and" in that sentence). To keep these two roles notationally apart, I will use $\mathrm{AND}, \mathrm{OR}, \mathrm{NOT}$ (allowing infix notation) for the role as operators, and $\wedge, \vee, \neg$

for the logic language. The above sentence would then be formally written as "$\forall x \forall y\,(x \text{ AND } y \leq x) \wedge (x \text{ AND } y \leq y)$".

The definition of signatures and ECL-sentences in many respects follows standard customs (with significant simplifications to be explained after the definition) and is the same for any conceptor dimension N:

Definition 10 *Let* $Var = \{x_1, x_2, ...\}$ *be a fixed countable indexed set of* variables.

1. *(ECL-signatures) The objects (signatures) of* **Sign** *are all countable sets, whose elements are called* symbols. *For signatures* Σ_1, Σ_2*, the set of morphisms* $hom\,(\Sigma_1, \Sigma_2)$ *is the set of all functions* $\phi : \Sigma_1 \to \Sigma_2$.

2. *(ECL-terms) Given a signature* Σ*, the set of* Σ-terms *is defined inductively by*

   (a) *Every variable* x_i*, every symbol* $a \in \Sigma$*, and* I *is a* Σ*-term.*

   (b) *For* Σ*-terms* t_1, t_2 *and* $\gamma \in [0, \infty]$*, the following are* Σ*-terms:* NOT t_1, $(t_1 \text{ AND } t_2)$, $(t_1 \text{ OR } t_2)$, *and* $\varphi(t_1, \gamma)$.

3. *(ECL-expressions) Given a signature* Σ*, the set* $Exp(\Sigma)$ *of* Σ-expressions *is defined inductively by*

   (a) *If* t_1, t_2 *are* Σ*-terms, then* $t_1 \leq t_2$ *is a* Σ*-expression.*

   (b) *If* e_1, e_2 *are* Σ*-expressions, and* x_i *a variable, then the following are* Σ*-expressions:* $\neg e_1$, $(e_1 \wedge e_2)$, $(e_1 \vee e_2)$, $\forall x_i\, e_1$.

4. *(ECL-sentences) A* Σ*-expression that contains no free variables is a* Σ-sentence *(free occurrence of variables to be defined as usual, omitted here.)*

Given a Σ-morphism $\phi : \Sigma_1 \to \Sigma_2$, its image $Sen(\phi)$ under the functor Sen is the map which sends every Σ_1-sentence χ_1 to the Σ_2-sentence χ_2 obtained from χ_1 by replacing all occurrences of Σ_1 symbols in χ_1 by their images under ϕ. I omit the obvious inductive definition of this replacement construction.

Notes:

- ECL only has a single sort of symbols with arity 0, namely constant symbols (which will be made to refer to conceptors later). This renders the categorical treatment of ECL much simpler than it is for logics with sorted symbols of varying arities.

- The operator symbols NOT, AND, OR, the parametrized operation symbol $\varphi(\cdot, \gamma)$ and the relation symbol $\leq$ are not made part of signatures, but become universal elements in the construction of sentences.

The models of ECL are quite simple. For a signature Σ, the objects of $Mod(\Sigma)$ are the sets of Σ-indexed N-dimensional conceptor matrices

$$Ob(Mod(\Sigma)) = \{m \subset \mathcal{C}_{N\times N} \times \Sigma \mid \forall \sigma \in \Sigma\ \exists^{=1} C \in \mathcal{C}_{N\times N} : (C,\sigma) \in m\}$$

where $\mathcal{C}_{N\times N}$ is the set of all N-dimensional conceptor matrices. The objects of $Mod(\Sigma)$ are thus the graph sets of the functions from Σ to the set of N-dimensional conceptor matrices. The model morphisms of $Mod(\Sigma)$ are canonically given by the index-preserving maps

$$hom(\{(C_1,\sigma)\},\{(C_2,\sigma)\}) = \{\mu : \{(C_1,\sigma)\} \to \{(C_2,\sigma)\} \mid \mu : (C_1,\sigma) \mapsto (C_2,\sigma)\}.$$

Clearly, $hom(\{(C,\sigma)\},\{(C',\sigma)\})$ contains exactly one element.

Given a signature morphism $\Sigma_1 \xrightarrow{\phi} \Sigma_2$, then $Mod(\phi)$ is defined to be a map from $Mod(\Sigma_2)$ to $Mod(\Sigma_1)$ as follows. For a Σ_2-model $m_2 = \{(C_2,\sigma_2)\}$ let $[\![\sigma_2]\!]^{m_2}$ denote the interpretation of σ_2 in m_2, that is, $[\![\sigma_2]\!]^{m_2}$ is the conceptor matrix C_2 for which $(C_2,\sigma_2) \in m_2$. Then $Mod(\phi)$ assigns to to m_2 the Σ_1-model $m_1 = Mod(\phi)(m_2) = \{([\![\phi(\sigma_1)]\!]^{m_2},\sigma_1)\} \in Mod(\Sigma_1)$.

The model relations $\models_\Sigma$ are defined in the same way as in the familiar first-order logic. Omitting some detail, here is how:

Definition 11 *Preliminaries: A map $\beta : Var \to \mathcal{C}_{N\times N}$ is called a* variable assignment. *$\mathcal{B}$ is the set of all variable assignments. We denote by $\beta\frac{C}{x_i}$ the variable assignment that is identical to β except that x_i is mapped to C. A* Σ-interpretation *is a pair $\mathcal{I} = (m,\beta)$ consisting of a Σ-model m and a variable assignment β. By $\mathcal{I}\frac{C}{x_i}$ we denote the interpretation $(m,\beta\frac{C}{x_i})$. For a Σ-term t, the interpretation $\mathcal{I}(t) \in \mathcal{C}_{N\times N}$ is defined in the obvious way. Then $\models^*_\Sigma \subseteq (Mod(\Sigma)\times\mathcal{B}) \times Exp(\Sigma)$ is defined inductively by*

1. $\mathcal{I} \models^*_\Sigma t_1 \le t_2$ *iff* $\mathcal{I}(t_1) \le \mathcal{I}(t_2)$,
2. $\mathcal{I} \models^*_\Sigma \neg e$ *iff not* $\mathcal{I} \models^*_\Sigma e$,
3. $\mathcal{I} \models^*_\Sigma (e_1 \wedge e_2)$ *iff* $\mathcal{I} \models^*_\Sigma e_1$ *and* $\mathcal{I} \models^*_\Sigma e_2$,
4. $\mathcal{I} \models^*_\Sigma (e_1 \vee e_2)$ *iff* $\mathcal{I} \models^*_\Sigma e_1$ *or* $\mathcal{I} \models^*_\Sigma e_2$,
5. $\mathcal{I} \models^*_\Sigma \forall x_i\, e$ *iff for all* $C \in \mathcal{C}_{N\times N}$ *it holds that* $\mathcal{I}\frac{C}{x_i} \models^*_\Sigma e$.

$\models_\Sigma$ then is the restriction of $\models^*_\Sigma$ on sentences.

This completes the definition of ECL as an institution. The satisfaction condition obviously holds. While in many respects ECL follows the role model of first-order logic, the associated model theory is much more restricted in that only N-dimensional conceptors are admitted as interpretations of symbols. The natural next step would be to design calculi for ECL and investigate whether this logic is complete or even decidable. Clarity on this point would amount to an insight

in the computational tractability of knowledge representation based on matrix conceptors with Boolean and aperture adaptation opertors.

Intrinsic conceptor logic. I want to present ICL as a model of the "logics" which might unfold *inside* a neural agent. All constitutents of ICL should be realizable in terms of neurodynamical processes, giving a logic not for reasoning *about* conceptors, but *with* conceptors.

Taking the idea of placing "logics" *inside* an agent seriously has a number of consequences which lead quite far away from traditional intuitions about "logics":

- Different agents may have different logics. I will therefore not try to define a general ICL that would fit any neural agent. Instead every concrete agent with a concrete lifetime learning history will need his/her/its own individual conceptor logic. I will use the signal de-noising and classification architecture from Section 3.15 as an example "agent" and describe how an ICL can be formulated as an institution for this particular case. Some general design principles will however become clear from this case study.

- Conceptors are all about temporal processes, learning and adaptation. An agent's private ICL will have to possess an eminently dynamical character. Concepts will change their meaning over time in an agent. This "personal history dynamics" is quintessential for modeling an agent and should become reflected in making an ICL a dynamical object itself – as opposed to introducing time through descriptive syntactical elements in an otherwise static logic, like it is traditionally done by means of modal operators or axioms describing a timeline. In my proposal of ICLs, time enters the picture through the central constitutent of an institution, signature morphisms. These maps between signatures (all commanded by the same agent) will model time, and an agent's lifetime history of adaptation will be modeled by an evolution of signatures. Where the original core motif for casting logics as institutions was that "truth is invariant under change of notation" ([37]), the main point of ICLs could be contrasted as "concepts and their meaning change with time". The role of signature morphisms in ICLs is fundamentally different in ICLs compared to customary formalizations of logics. In the latter, signature changes should leave meaning invariant; in the former, adaptive changes in conceptors are reflected by temporally indexed changes in signature.

- Making an ICL private to an agent implies that the model relation $\models$ becomes agent-specific. An ICL cannot be specified as an abstract object in isolation. Before it can be defined, one first needs to have a formal model of a particular agent with a particular lifelong adaptation history.

In sum, an ICL (formalized as institution) itself becomes a dynamical system, defined relative to an existing (conceptor-based) neural agent with a particular adaptation history. The "state space" of an ICL will be the set of signatures. A "trajectory" of the temporal evolution of an ICL will essentially be a sequence of

signatures, enriched with information pertaining to forming sentences and models. For an illustration, assume that a neural agent adapts two random feature conceptors $a(n), b(n)$. These are named by two *temporally indexed* symbols $A^{(n)}, B^{(n)}$. A signature will be a timeslice of these, $\Sigma^{(n)} = \{A^{(n)}, B^{(n)}\}$. For every pair of integer timepoints $(n+k, n)$ (where $k \geq 0$) there will be a signature morphism $\phi^{(n+k,n)} : \Sigma^{(n+k)} \to \Sigma^{(n)}$. The (strong) reason why signature morphisms point backwards in time will become clear later. Figure 45 visualizes the components of this example. The dotted lines connecting the $A_i^{(n)}$ are suggestive graphical hints that the symbols $A_i^{(n)}$ all name the "same" conceptor a_i. How this "sameness of identity over time" can be captured in the institution formalism will become clear presently.

Formal definitions of ICLs will vary depending on what kind of conceptors are used (for instance, matrix or random feature based), or whether time is taken to be discrete or continuous. I give a definition for discrete-time, random feature conceptor based ICLs.

Because ICLs will be models of an agent's private logic which evolves over the agent's lifetime, the definition of an ICL is stated relative to an agent's lifetime conceptor adaptation history. The only property of such an agent that is needed for defining an ICL is the existence of temporally adapted conceptors owned by the agent. Putting this into a formal definition:

Definition 12 *An* agent life *(here: M-dimensional random feature conceptor based, discrete time) is a structure $\mathcal{AL} = (T, \Sigma, \iota_{\mathcal{AL}})$, where*

1. *$T \subseteq \mathbb{Z}$ is an interval (finite or infinite) of the integers, the* lifetime *of $\mathcal{AL}$,*
2. *$\Sigma = \{A_1, \ldots, A_m\}$ is a finite nonempty set of* conceptor identifiers*,*
3. *$\iota_{\mathcal{AL}} : \Sigma \times T \to [0,1]^M, (A_i, n) \mapsto a_i(n)$ assigns to every time point and conceptor identifier an* adaptation version *$a_i(n)$ of the conceptor identified by the symbol A_i.*

As an example of an agent consider the signal de-noising and classification architecture (DCA) presented in Section 3.15, with a "life" being the concrete 16000-step adaptation run illustrated in Figure 42. In this example, the lifetime is $T = \{1, \ldots, 16000\}$. I will identify by symbols the four prototype conceptors $c^1, \ldots, c^4$ and the two auto-adapted conceptors $c_{[1]}^{\text{auto}}, c_{[2]}^{\text{auto}}$. Accordingly I choose Σ to be $\{C_1, \ldots, C_4, A_1, A_2\}$. The map $\iota_{\mathcal{AL}}$ is constant in time for the four protype conceptors: $\iota_{\mathcal{AL}}(n, C_j) = c^j$ for all $n \in T, j = 1, \ldots, 4$. For the remaining two conceptors, $\iota_{\mathcal{AL}}(n, A_i) = c_{[i]}^{\text{auto}}(n)$.

The stage is now prepared to spell out the definition of an agent's lifetime ICL (for the case of an agent based on M-dimensional random feature conceptor and discrete time):

Definition 13 *The* intrinsic conceptor logic (ICL) of an agent life*$\mathcal{AL} = (T, \Sigma, \iota_{\mathcal{AL}})$ is an institution whose components obey the following conditions:*

1. *The objects (signatures) of* **Sign** *are the pairs* $\Sigma^{(n)} = (\{A_1^{(n)}, \ldots, A_m^{(n)}\}, \sigma^{(n)})$, *where* $n \in T$, *and* $\sigma^{(n)} : \Sigma \to \{A_1^{(n)}, \ldots, A_m^{(n)}\}$ *is a bijection.*

   DCA example: *The lifetime of this example is* $T = \{1, \ldots, 16000\}$. *A signature* $\Sigma^{(n)} = (\{C_1^{(n)}, \ldots, C_4^{(n)}, A_1^{(n)}, A_2^{(n)}\}, \sigma^{(n)})$ *at time* $n \in T$ *will later be employed to denote some of the conceptors in the DCA in their adapted versions at time* n. *These conceptor adaptation versions will thus become identifiable by symbols from* $\Sigma^{(n)}$. *For* $\sigma^{(n)}$ *I take the natural projection* $C_j \mapsto C_j^{(n)}, A_i \mapsto A_i^{(n)}$.

2. *For every* $n, n+k \in T$ *(where* $k \geq 0$*),* $\phi^{(n+k,n)} : \Sigma^{(n+k)} \to \Sigma^{(n)}, A_i^{(n+k)} \mapsto (\sigma^{(n)} \circ (\sigma^{(n+k)})^{-1})(A_i^{(n+k)})$ *is a morphism in* **Sign**. *There are no other morphisms in* **Sign** *besides these.* Remark: *At first sight this might seem unneccessarily complicated. Why not simply require* $\phi^{(n+k,n)} : A_i^{(n+k)} \mapsto A_i^{(n)}$*? The reason is that the set of symbols* $\{A_1^{(n)}, \ldots A_m^{(n)}\}$ *of* $\Sigma^{(n)}$ *is just that, a set of symbols. That over time* $A_i^{(n)}$ *should correspond to* $A_i^{(n+k)}$ *is only visually suggested to us, the mathematicians, by the chosen notation for these symbols, but by no means does it actually follow from that notation.*

3. $Sen(\Sigma^{(n)})$ *is inductively defined as follows:*

   (a) $A_1^{(n)}, \ldots, A_m^{(n)}$ *and* I *and* 0 *are sentences in* $Sen(\Sigma^{(n)})$.

   (b) *For* $k \geq 0$ *such that* $n, n+k \in T$, *for* $A_i^{(n)} \in \Sigma^{(n)}$, $\delta_k^{(n)} A_i^{(n)}$ *is in* $Sen(\Sigma^{(n)})$. Remark: *the* δ *operators capture the temporal adaptation of conceptors. The symbol* $A_i^{(n)}$ *will be used to denote a conceptor* $a_i(n)$ *in its adaptation version at time* n, *and the sentence* $\delta_k^{(n)} A_i^{(n)}$ *will be made to refer to* $a_i(n+k)$.

   (c) *If* $\zeta, \xi \in Sen(\Sigma^{(n)})$, *then* $(\zeta \vee \xi), (\zeta \wedge \xi), \neg\zeta \in Sen(\Sigma^{(n)})$. Remark: *unlike in ECL there is no need for a notational distinction between* $\wedge$ *and* AND *etc.*

   (d) *If* $\zeta \in Sen(\Sigma^{(n)})$, *then* $\varphi(\zeta, \gamma) \in Sen(\Sigma^{(n)})$ *for every* $\gamma \in [0, \infty]$ *(this captures aperture adaptation).*

   (e) *If* $\zeta, \xi \in Sen(\Sigma^{(n)})$ *and* $0 \leq b \leq 1$, *then* $\beta_b(\zeta, \xi) \in Sen(\Sigma^{(n)})$ *(this will take care of linear blends* $b\zeta + (1-b)\xi$*).*

   Remark: *Including* I *and* 0 *in the sentence syntax is a convenience item.* 0 *could be defined in terms of any* $A_i^{(n)}$ *by* $0 \stackrel{\wedge}{=} (\varphi(A_i^{(n)}, \infty) \wedge \neg\varphi(A_i^{(n)}, \infty))$, *and* I *by* $I \stackrel{\wedge}{=} \neg 0$. *Likewise,* $\vee$ *(or* $\wedge$*) could be dismissed because it can be expressed in terms of* $\wedge$ *and* $\neg$ *(*$\vee$ *and* $\neg$*, respectively).*

4. *For a signature morphism* $\phi^{(n+k,n)} : \Sigma^{(n+k)} \to \Sigma^{(n)}$, $Sen(\phi^{(n+k,n)}) : Sen(\Sigma^{(n+k)}) \to Sen(\Sigma^{(n)})$ *is the map defined inductively as follows:*

   (a) $Sen(\phi^{(n+k,n)}) : I \mapsto I, 0 \mapsto 0$.

(b) $Sen(\phi^{(n+k,n)}) : A_i^{(n+k)} \mapsto \delta_k^{(n)}\, \phi^{(n+k,n)}(A_i^{(n+k)})$. Remark 1: *When we use the natural projections $\sigma^{(n)} : A_i \mapsto A_i^{(n)}$, this rule could be more simply written as $Sen(\phi^{(n+k,n)}) : A_i^{(n+k)} \mapsto \delta_k^{(n)}\, A_i^{(n)}$.* Remark 2: *This is the pivotal point in this entire definition, and the point where the difference to customary views on logics comes to the surface most conspicuously. Usually signature morphisms act on sentences by simply re-naming all signature symbols that occur in a sentence. The structure of a sentence remains unaffected, in agreement with the motto "truth is invariant under change of notation". By contrast, here a signature symbol $A_i^{(n+k)}$ is replaced by an temporal change operator term $\delta_k^{(n)}\, A_i^{(n)}$, reflecting the new motto "meaning changes with time". The fact that $\phi^{(n+k,n)}$ leads from $A_i^{(n+k)}$ to $A_i^{(n)}$ establishes "sameness of identity over time" between $A_i^{(n+k)}$ and $A_i^{(n)}$. Usually one would formally express sameness of identity of some mathematical entity by using the same symbol to name that entity at different time points. Here different symbols are used, and thus another mechanism has to be found in order to establish that an entity named by different symbols at different times remains "the same". The dotted "identity" lines in Figure 45 are fixed by the signature morphisms $\phi^{(n+k,n)}$, not by using the same symbol over time.* Remark 3: *At this point it also becomes clear why the signature morphisms $\phi^{(n+k,n)} : \Sigma^{(n+k)} \to \Sigma^{(n)}$ lead backwards in time. A conceptor $a_i(n+k)$ in its time-$(n+k)$ version can be expressed in terms of the earlier version $a_i(n)$ with the aid of the temporal evolution operator δ, but in general an earlier version $a_i(n)$ cannot be expressed in terms of a later $a_i(n+k)$. This reflects the fact that, seen as a trajectory of an input-driven dynamical system, an agent life is (typically) irreversible. To put it into everyday language, "the future can be explained from the past, but not vice versa".*

(c) $Sen(\phi^{(n+k,n)}) : \delta_l^{(n+k)}\, A_i^{(n+k)} \mapsto \delta_{(k+l)}^{(n)}\, \phi^{(n+k,n)}(A_i^{(n+k)})$.

(d) *For $\zeta, \xi \in Sen(\Sigma^{(n+k)})$, put*

$$
\begin{array}{rcl}
Sen(\phi^{(n+k,n)}) : & & \\
(\zeta \vee \xi) & \mapsto & (Sen(\phi^{(n+k,n)})(\zeta) \vee Sen(\phi^{(n+k,n)})(\xi)), \\
(\zeta \wedge \xi) & \mapsto & (Sen(\phi^{(n+k,n)})(\zeta) \wedge Sen(\phi^{(n+k,n)})(\xi)), \\
\neg\zeta & \mapsto & \neg Sen(\phi^{(n+k,n)})(\zeta), \\
\varphi(\zeta,\gamma) & \mapsto & \varphi(Sen(\phi^{(n+k,n)})(\zeta), \gamma), \\
\beta_b(\zeta,\xi) & \mapsto & \beta_b(Sen(\phi^{(n+k,n)})(\zeta), Sen(\phi^{(n+k,n)})(\xi)).
\end{array}
$$

5. *For every signature $\Sigma^{(n)} \in$ **Sign**, $Mod(\Sigma^{(n)})$ is always the same category **Z** with objects all non-negative M-dimensional vectors $z^{\cdot\wedge 2}$. There are no model morphisms except the identity morphisms $z^{\cdot\wedge 2} \xrightarrow{id} z^{\cdot\wedge 2}$.*

6. *For every morphism* $\phi^{(n+k,n)} \in \mathbf{Sign}$, $Mod(\phi^{(n+k,n)})$ *is the identity morphism of* $\mathbf{Z}$.

7. *As a preparation for defining the model relationships* $\models_{\Sigma^{(n)}}$ *we assign by induction to every sentence* $\chi \in Sen(\Sigma^{(n)})$ *an* M*-dimensional conception weight vector* $\iota(\chi)$ *as follows:*

   (a) $\iota(I) = (1, \ldots, 1)'$ *and* $\iota(0) = (0, \ldots, 0)'$.

   (b) $\iota(A_i^{(n)}) = \iota_{\mathcal{AL}}(n, (\sigma^{(n)})^{-1} A_i^{(n)})$.

   (c) $\iota(\delta_k^{(n)} A_i^{(n)}) = \iota_{\mathcal{AL}}(n+k, (\sigma^{(n)})^{-1} A_i^{(n)})$.

   (d) *Case* $\chi = (\zeta \vee \xi)$: $\iota(\chi) = \iota(\zeta) \vee \iota(\xi)$ *(compare Definition 7).*

   (e) *Case* $\chi = (\zeta \wedge \xi)$: $\iota(\chi) = \iota(\zeta) \wedge \iota(\xi)$.

   (f) *Case* $\chi = \neg\zeta$: $\iota(\chi) = \neg\iota(\zeta)$.

   (g) *Case* $\chi = \varphi(\zeta, \gamma)$: $\iota(\chi) = \varphi(\iota(\zeta), \gamma)$ *(compare Definition 6).*

   (h) *Case* $\chi = \beta_b(\zeta, \xi)$: $\iota(\chi) = b\,\iota(\zeta) + (1-b)\iota(\xi)$.

8. *For* $z^{\cdot\wedge 2} \in \mathbf{Z} = Mod(\Sigma^{(n)})$ *and* $\chi \in Sen(\Sigma^{(n)})$, *the model relationship is defined by*

$$z^{\cdot\wedge 2} \models_{\Sigma^{(n)}} \chi \quad \textit{iff} \quad z^{\cdot\wedge 2} .* (z^{\cdot\wedge 2} + 1)^{\cdot\wedge -1} \leq \iota(\chi).$$

The satisfaction condition (100) requires that for $\chi^{(n+k)} \in Sen(\Sigma^{(n+k)})$ and $z^{\cdot\wedge 2} \in Mod(\Sigma^{(n)})$ it holds that

$$z^{\cdot\wedge 2} \models_{\Sigma^{(n)}} Sen(\phi^{(n+k,n)})(\chi^{(n+k)}) \text{ iff } Mod(\phi^{(n+k,n)})(z^{\cdot\wedge 2}) \models_{\Sigma^{(n+k)}} \chi^{(n+k)},$$

which is equivalent to

$$z^{\cdot\wedge 2} \models_{\Sigma^{(n)}} Sen(\phi^{(n+k,n)})(\chi^{(n+k)}) \text{ iff } z^{\cdot\wedge 2} \models_{\Sigma^{(n+k)}} \chi^{(n+k)} \tag{101}$$

because $Mod(\phi^{(n+k,n)})$ is the identity morphism on $\mathbf{Z}$. This follows directly from the following fact:

Lemma 1 $\iota\left(Sen(\phi^{(n+k,n)})(\chi^{(n+k)})\right) = \iota(\chi^{(n+k)})$,

which in turn can be established by an obvious induction on the structure of sentences, where the crucial steps are the cases (i) $\chi^{(n+k)} = A_i^{(n+k)}$ and (ii) $\chi^{(n+k)} = \delta_l^{(n+k)} A_i^{(n+k)}$.

In case (i), $\iota(Sen(\phi^{(n+k,n)})(A_i^{(n+k)})) = \iota(\delta_k^{(n)} \phi^{(n+k,n)}(A_i^{(n+k)})) = \iota(\delta_k^{(n)}(\sigma^{(n)} \circ (\sigma^{(n+k)})^{-1})(A_i^{(n+k)})) = \iota_{\mathcal{AL}}(n+k, (\sigma^{(n)}))^{-1}(\sigma^{(n)} \circ (\sigma^{(n+k)})^{-1})(A_i^{(n+k)})) = \iota_{\mathcal{AL}}(n+k, (\sigma^{(n+k)})^{-1}(A_i^{(n+k)})) = \iota(A_i^{(n+k)})$.

In case (ii), conclude $\iota(Sen(\phi^{(n+k,n)})(\delta_l^{(n+k)} A_i^{(n+k)})) = \iota(\delta_{(k+l)}^{(n)}\, \phi^{(n+k,n)}(A_i^{(n+k)})) = \iota(\delta_{(k+l)}^{(n)}\, (\sigma^{(n)} \circ (\sigma^{(n+k)})^{-1})(A_i^{(n+k)})) = \iota_{\mathcal{AL}}(n+k+l, (\sigma^{(n)}))^{-1}(\sigma^{(n)} \circ (\sigma^{(n+k)})^{-1})(A_i^{(n+k)})) = \iota_{\mathcal{AL}}(n+k+l, (\sigma^{(n+k)})^{-1}(A_i^{(n+k)})) = \iota(\delta_l^{(n+k)}\, A_i^{(n+k)})$.

An important difference between "traditional" logics and the ICL of an agent $\mathcal{AL}$ concerns different intuitions about semantics. Taking first-order logic as an example of a traditional logic, the "meaning" of a first-order sentence χ with signature Σ is the class of all of its models. Whether a set is a model of χ depends on how the symbols from Σ are extensionally interpreted over that set. First-order logic by itself does not prescribe how the symbols of a signature have to be interpreted over some domain. In contrast, an ICL is defined with respect to a concrete agent $\mathcal{AL}$, which in turn uniquely fixes how the symbols from an ICL signature $\Sigma^{(n)}$ *must* be interpreted – this is the essence of points *7. (a – c)* in Definition 13.

Logical entailment in an ICL coincides with abstraction of conceptors:

Proposition 18 *In an ICL, for all $\zeta, \xi \in Sen(\Sigma^{(n)})$ it holds that*

$$\zeta \models_{\Sigma^{(n)}} \xi \quad \textit{iff} \quad \iota(\zeta) \leq \iota(\xi). \tag{102}$$

The simple proof is given in Section 5.11. In an agent's lifetime ICL, for any sentence $\chi \in Sen(\Sigma^{(n)})$ the concrete interpretation $\iota(\chi) \in [0,1]^M$ can be effectively computed via the rules stated in Nr. *7* in Definition 13, provided one has access to the identifiable conceptors $a_i(n)$ in the agent's life. Since for two vectors $a, b \in [0,1]^M$ it can be effectively checked whether $a \leq b$, it is decidable whether $\iota(\zeta) \leq \iota(\xi)$. An ICL is therefore decidable.

Seen from a categorical point of view, an ICL is a particularly small-size institution. Since the (only) category in the image of *Mod* is a set, we can regard the functor *Mod* as having codomain $\mathbf{Set}^{\mathrm{op}}$ instead of $\mathbf{Cat}^{\mathrm{op}}$. Also the category **Sign** is small, that is, a set. Altogether, an ICL institution nowhere needs proper classes.

The ICL definition I gave here is very elementary. It could easily be augmented in various natural ways, for instance by admitting permutations and/or projections of conceptor vector components as model morphisms in $\mathbf{Z}$, or allowing conceptors of different dimensions in the makeup of an agent life. Likewise it is straightforward to spell out definitions for an agent life and its ICL for matrix-based conceptors. In the latter case, the referents of sentences are correlation matrices R, and the defining equation for the model relationship appears as

$$R \models_{\Sigma^{(n)}} \chi \quad \text{iff} \quad R(R+I)^{-1} \leq \iota(\chi).$$

In traditional logics and their applications, an important role is played by *calculi*. A calculus is a set of syntactic transformation rules operating on sentences (and expressions containing free variables) which allows one to derive purely syntactical proofs of logical entailment statements $\zeta \models \xi$. Fundamental properties of

familiar logics, completeness and decidability in particular, are defined in terms of calculi. While it may be possible and mathematically interesting to design syntactical calculi for ICLs, they are not needed because $\zeta \models \xi$ is decidable via the semantic equivalent $\iota(\zeta) \leq \iota(\xi)$. Furthermore, the ICL decision procedure is computationally very effective: only time $O(1)$ is needed (admitting a parallel execution) to determine whether some conception vector is at most as large as another in all components. This may help to explain why humans can so quickly carry out many concept subsumption judgements ("this looks like a cow to me").

3.18 Final Summary and Outlook

Abstracting from all technical detail, here is a summary account of the conceptor approach:

From neural dynamics to conceptors. Conceptors capture the shape of a neural state cloud by a positive semi-definite operator.

From conceptors to neural dynamics. Inserting a conceptor into a neurodynamical state update loop allows to select and stabilize a previously stored neural activation pattern.

Matrix conceptors are useful in machine learning applications and as mathematical tools for analysing patterns emerging in nonlinear neural dynamics.

Random feature conceptors are not biologically apriori implausible and can be neurally coded by single neurons.

Autoconceptor adaptation dynamics leads to content-addressable neural memories of dynamical patterns and to signal filtering and classification systems.

Boolean operations and the abstraction ordering on conceptors establish a bi-directional connection between logic and neural dynamics.

From static to dynamic models. Conceptors allow to understand and control dynamical patterns in scenarios that previously have been mostly restricted to static patterns. Specifically this concerns content-addressable memories, morphing, ordering concepts in logically structured abstraction hierarchies, and top-down hypothesis control in hierarchical architectures.

The study of conceptors is at an early stage. There are numerous natural directions for next steps in conceptor research:

Affine conceptor maps. Conceptors constrain reservoir states by applying a positive semi-definite map. This map is adapted to the "soft-bounded" linear subspace visited by a reservoir when it is driven through a pattern. If the pattern-driven reservoir states do not have zero mean, it seems natural to

first subtract the mean before applying the conceptor. If the mean is μ, this would result in an update loop of the form $x(n+1) = \mu + C(\tanh(...) - \mu)$. While this may be expected to improve control characteristics of pattern re-generation, it is not immediately clear how Boolean operations transfer to such affine conceptors which are characterized by pairs (C, μ).

Nonlinear conceptor filters. Even more generally, one might conceive of conceptors which take the form of nonlinear filters, for instance instantiated as feedforward neural networks. Such filters F could be trained on the same objective function as I used for conceptors, namely minimizing $E[\|z - F(z)\|^2] + \alpha^{-2}\|F\|^2$, where $\|F\|$ is a suitably chosen norm defined for the filter. Like with affine conceptor maps, the pattern specificity of such nonlinear conceptor filters would be greater than for our standard matrix conceptors, but again it is not clear how logical operations would extend to such filters.

Basic mathematical properties. From a mathematics viewpoint, there are some elementary questions about conceptors which should be better understood, for instance:

- What is the relationship between Boolean operations, aperture adaptation, and linear blending? in particular, can the latter be expressed in terms of the two former?
- Given a dimension N, what is the minimial number of "basis" conceptors such that the transitive closure under Boolean operations, aperture adaptation, and/or linear blending is the set of all N-dimensional conceptors?
- Are there normal forms for expressions which compose conceptors by Boolean operations, aperture adaptation, and/or linear blending?
- Find conceptor analogs of standard structure-building mathematical operations, especially products. These would be needed to design architectures with several conceptor modules (likely of different dimension) where the "soft subspace constraining" of the overall dynamics works on the total architecture state space. Presumably this leads to tensor variants of conceptors. This may turn out to be a challenge because a mathematical theory of "semi positive-definite tensors" seems to be only in its infancy (compare [84]).

Neural realization of Boolean operations. How can Boolean operations be implemented in biologically not impossible neural circuits?

Complete analysis of autoconceptor adaptation. The analysis of autoconceptor adaptation in Section 3.13.4 is preliminary and incomplete. It only characterizes certain aspects of fixed-point solutions of this adaptation dynamics but remains ignorant about the effects that the combined, nonlinear

conceptor-reservoir update dynamics may have when such a fixed-point solution is perturbed. Specifically, this reservoir-conceptor interaction will have to be taken into account in order to understand the dynamics in the center manifold of fixed-point solutions.

Applications. The usefulness of conceptors as a practical tool for machine learning and as a modeling tool in the cognitive and computational neurosciences will only be established by a suite of successful applications.

4 Documentation of Experiments and Methods

In this section I provide details of all simulation experiments reported in Sections 1 and 3.

4.1 General Set-Up, Initial Demonstrations (Section 1 and Section 3.2 - 3.4)

A reservoir with $N = 100$ neurons, plus one input unit and one output unit was created with a random input weight vector W^{in}, a random bias b and preliminary reservoir weights W^*, to be run according to the update equations

$$\begin{aligned} x(n+1) &= \tanh(W^* \, x(n) + W^{\text{in}} \, p(n+1) + b), \\ y(n) &= W^{\text{out}} \, x(n). \end{aligned} \tag{103}$$

Initially W^{out} was left undefined. The input weights were sampled from a normal distribution $\mathcal{N}(0,1)$ and then rescaled by a factor of 1.5. The bias was likewise sampled from $\mathcal{N}(0,1)$ and then rescaled by 0.2. The reservoir weight matrix W^* was first created as a sparse random matrix with an approximate density of 10%, then scaled to obtain a spectral radius (largest absolute eigenvalue) of 1.5. These scalings are typical in the field of reservoir computing [69] for networks to be employed in signal-generation tasks.

For each of the four driving signals p^j a training time series of length 1500 was generated. The reservoir was driven with these $p^j(n)$ in turn, starting each run from a zero initial reservoir state. This resulted in reservoir state responses $x^j(n)$. The first 500 steps were discarded in order to exclude data influenced by the arbitrary starting condition, leading to four 100-dimensional reservoir state time series of length $L = 1000$, which were recorded into four 100×1000 state collection matrices X^j, where $X^j(:,n) = x^j(n+500)$ $(j = 1, \ldots, 4)$. Likewise, the corresponding driver signals were recorded into four pattern collection (row) vectors P^j of size 1×1000. In addition to this, a version $\tilde{X}^j$ of X^j was built, identical to X^j except that it was delayed by one step: $\tilde{X}^j(:,n) = x^j(n+499)$. These collections were then concatenated to obtain $X = [X^1|X^2|X^3|X^4], \tilde{X} = [\tilde{X}^1|\tilde{X}^2|\tilde{X}^3|\tilde{X}^4], P = [P^1|P^2|P^3|P^4]$.

The "PC energy" plots in Figure 12 render the singular values of the correlation matrices $X^j(X^j)'/L$.

The output weights W^{out} were computed as the regularized Wiener-Hopf solution (also known as ridge regression, or Tychonov regularization)

$$W^{\text{out}} = ((XX' + \varrho^{\text{out}} I_{N\times N})^{-1} \, X \, P')', \tag{104}$$

where the regularizer ϱ^{out} was set to 0.01.

Loading: After loading, the reservoir weights W should lead to the approximate equality $Wx^j(n) \approx W^* \, x^j(n) + W^{\text{in}} \, p^j(n+1)$, across all patterns j, which leads to

the objective of minimizing the squared error $\epsilon^j(n+1) = ((\tanh^{-1}(x^j(n+1)) - b) - Wx^j(n))^2$, averaged over all four j and training time points. Writing B for the $100 \times (4 * 1000)$ matrix whose columns are all identical equal to b, this has the ridge regression solution

$$W = ((\tilde{X}\tilde{X}' + \varrho^{\mathrm{W}} I_{N\times N})^{-1}\ \tilde{X}\,(\tanh^{-1}(X) - B))', \tag{105}$$

where the regularizer ϱ^{W} was set to 0.0001. To assess the accuracy of the weight computations, the training normalized root mean square error (NRMSE) was computed. For the readout weights, the NRMSE between $y(n) = W^{\mathrm{out}}x(n)$ and the target P was 0.00068. For the reservoir weights, the average (over reservoir neurons i, times n and patterns j) NRMSE between $W(i,:)x^j(n)$ and the target $W^*(i,:)\,x^j(n) + W^{\mathrm{in}}(i)\,p^j(n+1)$ was 0.0011.

In order to determine the accuracy of fit between the original driving signals p^j and the network observation outputs $y^j(n) = W^{\mathrm{out}}\,C^j\,\tanh(W\,x(n-1)+b)$ in the conceptor-constrained autonomous runs, the driver signals and the y^j signals were first interpolated with cubic splines (oversampling by a factor of 20). Then a segment length 400 of the oversampled driver (corresponding to 20 timesteps before interpolation) was shifted over the oversampled y^j in search of a position of best fit. This is necessary to compensate for the indeterminate phaseshift between the driver data and the network outputs. The NRMSEs given in Figure 12 were calculated from the best-fit phaseshift position, and the optimally phase-shifted version of y^j was also used for the plot.

4.2 Aperture Adaptation (Sections 3.8.3 and 3.8.4)

Data generation. For the Rössler attractor, training time series were obtained from running simple Euler approximations of the following ODEs:

$$\begin{aligned} \dot{x} &= -(y+z) \\ \dot{y} &= x + a\,y \\ \dot{z} &= b + x\,z - c\,z, \end{aligned}$$

using parameters $a = b = 0.2, c = 8$. The evolution of this system was Euler approximated with stepsize $1/200$ and the resulting discrete time series was then subsampled by 150. The x and y coordinates were assembled in a 2-dimensional driving sequence, where each of the two channels was shifted/scaled to a range of $[0, 1]$. For the Lorenz attractor, the ODE

$$\begin{aligned} \dot{x} &= \sigma(y - x) \\ \dot{y} &= r\,x - y - x\,z \\ \dot{z} &= x\,y - b\,z \end{aligned}$$

with $\sigma = 10, r = 28, b = 8/3$ was Euler-approximated with stepsize 1/200 and subsequent subsampling by 15. The x and z coordinates were collected in a 2-dimensional driving sequence, again each channel normalized to a range of $[0, 1]$. The Mackey Glass timeseries was obtained from the delay differential equation

$$\dot{x}(t) = \frac{\beta\, x(t-\tau)}{1 + x(t-\tau)^n} - \gamma\, x(t)$$

with $\beta = 0.2, n = 10, \tau = 17, \gamma = 0.1$, a customary setting when this attractor is used in neural network demonstrations. An Euler approximation with stepsize 1/10 was used. To obtain a 2-dim timeseries that could be fed to the reservoir through the same two input channels as the other attractor data, pairs $x(t), x(t-\tau)$ were combined into 2-dim vectors. Again, these two signals were normalized to the $[0, 1]$ range. The Hénon attractor is governed by the iterated map

$$\begin{aligned} x(n+1) &= y(n) + 1 - a\, x(n) \\ y(n+1) &= b\, x(n), \end{aligned}$$

where I used $a = 1.4, b = 0.3$. The two components were filed into a 2-dim timeseries $(x(n), y(n))'$ with no further subsampling, and again normalization to a range of $[0, 1]$ in each component.

Reservoir setup. A 500-unit reservoir RNN was created with a normal distributed, 10%-sparse weight matrix W^* scaled to a spectral radius of 0.6. The bias vector b and input weights W^{in} (sized 400×2 for two input channels) were sampled from standard normal distribution and then scaled by 0.4 and 1.2, respectively. These scaling parameters were found by a (very coarse) manual optimization of the performance of the pattern storing process. The network size was chosen large enough to warrant a robust trainability of the four chaotic patterns. Repeated executions of the experiment with different randomly initialized weights (not documented) showed no significant differences.

Pattern storing. The W^* reservoir was driven, in turn, by 2500 timesteps of each of the four chaotic timeseries. The first 500 steps were discarded to account for initial reservoir state washout, and the remaining 4×2000 reservoir states were collected in a 500×8000 matrix X. From this, the new reservoir weights W were computed as in (105), with a regularizer $\varrho^{\text{W}} = 0.0001$. The readout weights were computed as in (104) with regularizer $\varrho^{\text{out}} = 0.01$. The average NRMSEs obtained for the reservoir and readout weights were 0.0082 and 0.013, respectively.

Computing conceptors. From each of the four $n = 2000$ step reservoir state sequences X recorded in the storing procedure, obtained from driving the reservoir with one of the four chaotic signals, a preliminary correlation matrix $\tilde{R} = XX'/2000$ and its SVD $U\tilde{S}U' = \tilde{R}$ were computed. This correlation matrix was then used to obtain a conceptor associated with the respective chaotic signal, using an aperture $\alpha = 1$. From these unit-aperture conceptors, versions with differing α were obtained through aperture adaptation per (16).

In passing I note that the overall stability and parameter robustness of this simulation can be much improved if small singular values in $\tilde{S}$ (for instance, with values smaller than 1e-06) are zeroed, obtaining a clipped S, from which a "cleaned-up" correlation matrix $R = USU'$ would be computed. This would lead to a range of well-working apertures spanning three orders of magnitude (not shown). I did not do this in the reported simulation in order to illustrate the effects of too large apertures; these effects would be partly suppressed when the spectrum of R is clipped.

Pattern retrieval and plotting. The loaded network was run using the conceptor-constrained update rule $x(n+1) = C\ \tanh(W\,x(n) + W^{\text{in}}\,p(n+1) + b)$ with various $C = \varphi(C_{\#}, \gamma_{\#,i})$ ($\# =$ R, L, MG, H, $i = 1, \ldots, 5$) for 800 steps each time, of which the first 100 were discarded to account for initial state washout. The delay embedding plots in Figure 18 were generated from the remaining 700 steps. Embedding delays of 2, 2, 3, 1 respectively were used for plotting the four attractors.

For each of the four 6-panel blocks in Figure 18, the five aperture adaptation factors $\gamma_{\#,i}$ were determined in the following way. First, by visual inspection, the middle $\gamma_{\#,3}$ was determined to fall in the trough center of the attenuation plot of Fig. 19 **A**. Then the remaining $\gamma_{\#,1}, \gamma_{\#,2}, \gamma_{\#,4}, \gamma_{\#,5}$ were set in a way that (i) the entire γ sequence was a geometrical progression, and (ii) that the plot obtained from the first $\gamma_{\#,1}$ was visually strongly corrupted.

4.3 Memory Management (Section 3.11)

For both reported simulation studies, the same basic reservoir was used: size $N = 100$, W^* sparse with approximately 10% nonzero entries, these sampled from a standard normal distribution and later rescaled to obtain a spectral radius of 1.5. The input weights W^{in} were non-sparse, sampled from a standard normal distribution and then scaled by a factor of 1.5. The bias vector b was sampled from a standard normal distribution and scaled by a factor of 0.25. The bias serves the important function to break the sign symmetry of the reservoir dynamics. Without it, in autonomous pattern generation runs a pattern p^j or its sign-reversed version $-p^j$ could be generated equally well, which is undesirable.

The driver patterns used in the first demonstration were either sinewaves with integer period lengths, or random periodic patterns scaled to a range of $[-0.9, 0.9]$. Period lengths were picked randomly between 3 and 15.

The drivers for the second demonstrations were the sinewaves $\sin(2{*}\pi{*}n/(3\sqrt{2}\cdot(1.1)^i))$ (where $i = 1, \ldots, 16$), leading to periods exponentially spread over the range $[4.24, 19.49]$. These signals were randomly permuted to yield the presentation order $j = \text{randperm}(i)$.

The lengths of pattern signals used for training were $L = 100$ in demo 1 and $L = 1000$ in demo 2, with additional washouts of the same lengths.

The readout weights W^{out} were trained by the generic procedure detailed in Section 4.1 after all patterns had been used to drive the network in turn for the

incremental storage, and thus having available all state collections X^j.

For testing, the conceptors C^j obtained during the storage procedure were used by running the reservoir via $x(n+1) = C^j \tanh(W^* x(n) + D x(n) + b)$ from random initial states. After a washout time, the network outputs $y^j(n)$ were recorded for durations of 200 and 400, respectively, in the two experiments. A 20-step portion of the original driver pattern was interpolated by a factor of 10 and shifted over a likewise interpolation-refined version of these outputs. The best matching shift position was used for plotting and NRMSE computation (such shift-search for a good fit is necessary because the autonomous runs are not phase-synchronized to the original drivers, and irrational-period sinewaves need interpolation because they may be phase-shifted by noninteger amounts).

4.4 Content-Addressable Memory (Section 3.13.3)

Network setup. Reservoir network matrices W were sampled sparsely (10% nonzero weights) from a normal distribution, then scaled to a spectral radius of 1.5 in all experiments of this section. Reservoir size was $N = 100$ for all period-4 experiments and the unrelated patterns experiment, and $N = 200$ for all experiments that used mixtures-of-sines patterns. For all experiments in the section, input weights W^{in} were randomly sampled from the standard normal distribution and rescaled by a factor of 1.5. The bias vector b was likewise sampled from the standard normal distribution and rescaled by 0.5. These scaling parameters had been determined by a coarse manual search for a well-working configuration when this suite experiments was set up. The experiments are remarkably insensitive to these parameters.

Storing patterns. The storage procedure was set up identically for all experiments in this section. The reservoir was driven by the k loading patterns in turn for $l = 50$ steps (period-4 and unrelated patterns) or $l = 500$ steps (mix-of-sines) time steps, plus a preceding 100 step initial washout. The observed network states $x(n)$ were concatenated into a $N \times kl$ sized state collection matrix X, and the one-step earlier states $x(n-1)$ into a matrix $\tilde{X}$ of same size. The driver pattern signals were concatenated into a kl sized row vector P. The readout weights W^{out} were then obtained by ridge regression via $W^{\text{out}} = ((XX' + 0.01\, I)^{-1}\, XP')'$, and D by $D = ((\tilde{X}\tilde{X}' + 0.001\, I)^{-1}\, \tilde{X}(W^{\text{in}}\, P)')'$.

Quality measurements. After the cueing, and at the end of the recall period (or at the ends of the three interim intervals for some of the experiments), the current conceptor C was tested for retrieval accuracy as follows. Starting from the current network state $x(n)$, the reservoir network was run for 550 steps, constrained by C. The first 50 steps served as washout and were discarded. The states x from the last 500 steps were transformed to patterns by applying W^{out}, yielding a 500-step pattern reconstruction. This was interpolated with cubic splines and then sampled at double resolution, leading to a 999-step pattern reconstruction $\tilde{y}$.

A 20-step template sample of the original pattern was similarly interpolated-

resampled and then passed over $\tilde{y}$, detecting the best-fitting position where the the NRMSE between the target template and $\tilde{y}$ was minimal (this shift-search accomodated for unknown phase shifts between the target template and $\tilde{y}$). This minimal NRMSE was returned as measurement result.

Irrational-period sines. This simulation was done exactly as the one before, using twelve irrational-period sines as reference patterns.

4.5 The Japanese Vowels Classification (Section 3.12)

Network setup. In each of the 50 trials, the weights in a fully connected, 10×10 reservoir weight matrix, a 12-dimensional input weight vector, a 10-dimensional bias vector, and a 10-dimensional start state were first sampled from a normal distribution, then rescaled to a spectral radius of 1.2 for W, and by factors of 0.2, 1, 1 for $W^{\text{in}}, b, x_{\text{start}}$ respectively.

Numerical determination of best aperture. To determine γ_i^+, the quantities $\|\varphi(\tilde{C}_i^+, 2^g)\|_{\text{fro}}^2$ were computed for $g = 0, 1, \ldots, 8$. These values were interpolated on a 0.01 raster with cubic splines, the support point $g_{\max}$ of the maximum of the interpolation curve was detected, returning $\gamma_i^+ = 2^{g\max}$.

Linear classifier training. The linear classifier that serves as a baseline comparison was designed in essentially the same way as the Echo State Networks based classifiers which in [57] yielded zero test misclassifications (when combining 1,000 such classifiers made from 4-unit networks) and 2 test misclassifications (when a single such classifier was based on a 1,000 unit reservoir), respectively. Thus, linear classifiers based on reservoir responses outperform all other reported methods on this benchmark and therefore provide a substantive baseline information.

In detail, the linear classifier was learnt from 270 training data pairs of the form (z, y_{teacher}), where the z were the same 88-dimensional vectors used for constructing conceptors, and the y_{teacher} were 9-dimensional, binary speaker indicator vectors with a "1" in the position of the speaker of z. The classifier consists in a 9×88 sized weight matrix V, and the cost function was the quadratic error $\|Vz - y_{\text{teacher}}\|^2$. The classifier weight matrix V which minimized this cost function on average over all training samples was computed by linear regression with Tychonov regularization, also known as ridge regression [106]. The Tychonov parameter which determines the degree of regularization was determined by a grid search over a 5-fold cross-validation on the training data. Across the 50 trials it was found to vary quite widely in a range between 0.0001 and 0.25; in a separate auxiliary investigation it was also found that the effect of variation of the regularizer within this range was very small and the training of the linear classifier can therefore be considered robust.

In testing, V was used to determine classification decisions by computing $y_{\text{test}} = Vz_{\text{test}}$ and opting for the index of the largest entry in y_{test} as the speaker.

4.6 Conceptor Dynamics Based on RFC Conceptors (Section 3.14)

Network setup. In both experiments reported in this section, the same reservoir made from $N = 100$ units was used. F and G were full matrices with entries first sampled from the standard normal distribution. Then they were both scaled by an identical factor a such that the product $a^2\,G\,F$ attained a spectral radius of 1.4. The input weight vector W^{in} was sampled from the standard normal distribution and then scaled by 1.2. The bias b was likewise sampled from the normal distribution and then scaled by 0.2.

These values were determined by coarse manual search, where the main guiding criterion was recall accuracy. The settings were rather robust in the first experiment which used stored c^j. The spectral radius could be *individually* varied from 0.6 to 1.45, the input weight scaling from 0.3 to 1.5, the bias scaling from 0.1 to 0.4, and the aperture from 3 to 8.5, while always keeping the final recall NRMSEs for all four patterns below 0.1. Furthermore, much larger *combined* variations of these scalings were also possible (not documented).

In the second experiment with content-addressed recall, the functional parameter range was much narrower. Individual parameter variation beyond $\pm 5\%$ was disruptive. Specifically, I observed a close inverse coupling between spectral radius and aperture: if one of the two was raised, the other had to be lowered.

Loading procedure. The loading procedure is described in some detail in the report text. The mean NRMSEs on training data was 0.00081 for recomputing G, 0.0011 for D, and 0.0029 for W^{out}. The mean absolute size of matrix elements in G was 0.021, about a third of the mean absolute size of elements of G^*.

Computing NRMSEs. The NRMSE comparison between the re-generated patterns at the end of the c adaptation and the original drivers was done in the same way as reported on earlier occasions (Section 4.4), that is, invoking spline interpolation of the comparison patterns and optimal phase-alignment.

4.7 Hierarchical Classification and Filtering Architecture (Section 3.15)

Module setup. The three modules are identical copies of each other. The reservoir had $N = 100$ units and the feature space had a dimension of $M = 500$. The input weight matrix W^{in} was sampled from the standard normal distribution and rescaled by a factor of 1.2. The reservoir-featurespace projection and backprojection matrices F, G^* (sized $N \times M$) were first sampled from the standard normal distribution, then linearly rescaled by a common factor such that the $N \times N$ matrix G^*F' (which functionally corresponds to an internal reservoir weight matrix) had a spectral radius of 1.4. The bias b was likewise sampled from the standard normal distribution and then scaled by 0.2.

A regularization procedure was then applied to G^* to give G as follows. The

preliminary module was driven per

$$z(n+1) = F' r(n), \quad r(n+1) = \tanh(G^* z(n+1) + W^{\text{in}} u(n) + b),$$

with an i.i.d. input signal $u(n)$ sampled uniformly from $[-1, 1]$, for 1600 steps (after discarding an initial washout). The values obtained for $z(n+1)$ were collected as columns in a $M \times 1600$ matrix Z. The final G was then computed by a ridge regression with a regularizer $a = 0.1$ by

$$G = ((ZZ' + aI)^{-1} Z (G^* Z)')'.$$

In words, G should behave as the initially sampled G^* in a randomly driven module, but do so with minimized weight sizes. This regularization was found to be important for a stable working of the final architecture.

Training. The input simulation weights D (size $1 \times M$) and W^{out} (size $1 \times N$) were trained by driving a single module with the four target patterns, as follows. The module was driven with clean $p^1, \ldots, p^4$ in turn, with auto-adaptation of conception weights c activated:

$$\begin{aligned} z(n+1) &= c(n) .* F' r(n), \\ r(n+1) &= \tanh(G z(n+1) + W^{\text{in}} p^j(n) + b), \\ c(n+1) &= c(n) + \lambda_c \left((z(n+1) - c(n) .* z(n+1)) .* z(n+1) - \alpha^{-2} c(n) \right), \end{aligned}$$

where the c adaptation rate was set to $\lambda_c = 0.5$, and an aperture $\alpha = 8$ was used. After discarding initial washouts in each of the four driving conditions (long enough for $c(n)$ to stabilize), 400 reservoir state vectors $r(n+1)$, 400 z vectors $z(n)$ and 400 input values $p^j(n)$ were collected for $j = 1, \ldots, 4$, and collected column-wise in matrices R (size $N \times 1600$), Z (size $M \times 1600$) and Q (size 1×1600), respectively. In addition, the 400-step submatrices Z^j of Z containing the z-responses of the module when driven with p^j were registered separately.

The output weights were then computed by ridge regression on the objective to recover $p^j(n)$ from $r(n+1)$ by

$$W^{\text{out}} = \left((RR' + aI)^{-1} RQ' \right)',$$

using a regularizer $a = 0.1$. In a similar way, the input simulation weights were obtained as

$$D = \left((ZZ' + aI)^{-1} ZQ' \right)'$$

with a regularizer $a = 0.1$ again. The training NRMSEs for W^{out} and D were 0.0018 and 0.0042, respectively.

The $M \times 4$ prototype matrix P was computed as follows. First, a preliminary version P^* was constructed whose j-the column vector was the mean of the element-wise squared column vectors in Z^j. The four column vectors of P^* were then normalized such that the norm of each of them was the mean of the norms

of columns in P^*. This gave P. This normalization is important for the performance of the architecture, because without it the optimization criterion (94) would systematically lead to smaller values for those γ^j that are associated with smaller-norm columns in P.

Baseline linear filter. The transversal filter that served as a baseline was a row vector w of size 2600. It was computed to minimize the loss function

$$\mathcal{L}(w) = \frac{1}{4} \sum_{j=1}^{4} E[p^j(n+1) - (p^j(n-2600+1), \ldots, p^j(n))^2] + a^2 \|w\|^2,$$

where $p^j(n)$ were clean versions of the four patterns. 400 timesteps per pattern were used for training, and a was set to 1.0. The setting of a was very robust. Changing a in either direction by factors of 100 changed the resulting test NRMSEs at levels below the plotting accuracy in Figure 42.

Parameter settings in testing. The adaptation rate λ_γ was set to 0.002 for the classification simulation and to 0.004 for the morph-tracking case study. The other global control parameters were identical in both simulations: trust smoothing rate $\sigma = 0.99$, decisiveness $d_{[12]} = d_{[23]} = 8$, drift $d = 0.01$, $c_{[l]}^{\text{aut}}$ adaptation rate $\lambda = 0.5$ (compare Equation (80); I used the same adaptation rate $\lambda_i \equiv \lambda$ for all of the 500 feature units).

Computing and plotting running NRMSE estimates. For the NRMSE plots in the fifth rows in Figures 42 and 43, a running estimate of the NRMSE between the module outputs $y_{[l]}$ and the clean input patterns p (unknown to the system) was computed as follows. A running estimate $\overline{\text{var}}\, p(n)$ of the variance of the clean pattern was maintained like it was done for $\overline{\text{var}}\, y_{[l]}(n)$ in (88) and (89), using an exponential smoothing rate of $\sigma = 0.95$. Then the running NRMSE was computed by another exponential smoothing per

$$\overline{\text{nrmse}}\, y_{[l]}(n+1) = \sigma\, \overline{\text{nrmse}}\, y_{[l]}(n) + (1-\sigma) \left(\frac{(p(n+1) - y_{[l]}(n+1))^2}{\overline{\text{var}}\, p(n+1)} \right)^{1/2}.$$

The running NRMSE for the baseline transversal filter were obtained in a similar fashion.

5 Proofs and Algorithms

5.1 Proof of Proposition 1 (Section 3.4)

Claim 1. We first re-write the minimization quantity, using $R = E[xx']$:

$$\begin{aligned} & E[\|x - Cx\|^2] + \alpha^{-2} \|C\|^2_{\text{fro}} = \\ &= E[\text{tr}\,(x'(I - C')(I - C)x)] + \alpha^{-2} \|C\|^2_{\text{fro}} \\ &= \text{tr}\,((I - C')(I - C)R + \alpha^{-2} C'C) \\ &= \text{tr}\,(R - C'R - CR + C'C(R + \alpha^{-2} I)) \\ &= \sum_{i=1,\ldots,N} e_i'(R - C'R - CR + C'C(R + \alpha^{-2} I))e_i. \end{aligned}$$

This quantity is quadratic in the parameters of C and non-negative. Because $\text{tr}\, C'C(R+\alpha^{-2} I) = \text{tr}\, C(R+\alpha^{-2} I)C'$ and $R+\alpha^{-2} I$ is positive definite, $\text{tr}\, C(R+\alpha^{-2} I)C'$ is positive definite in the N^2-dimensional space of C elements. Therefore $E[\|x - Cx\|^2] + \alpha^{-2} \|C\|^2_{\text{fro}}$ has a unique minimum in C space. To locate it we compute the derivative of the i-th component of this sum with respect to the entry $C(k, l) = C_{kl}$:

$$\begin{aligned} & \frac{\partial}{\partial C_{kl}} e_i'(R - C'R - CR + C'C(R + \alpha^{-2} I))e_i = \\ &= -2\, R_{kl} + \partial/\partial C_{kl} \sum_{j,a=1,\ldots,N} C_{ij}\, C_{ja}\, A_{ai} \\ &= -2\, R_{kl} + \sum_{a=1,\ldots,N} \partial/\partial C_{kl}\, C_{kj}\, C_{ka}\, A_{ai} \\ &= -2\, R_{kl} + \sum_{a=1,\ldots,N} (C_{ka}\, A_{al} + C_{ki}\, A_{li}) \\ &= -2\, R_{kl} + (CA)_{kl} + N\, C_{ki}\, A_{il}. \end{aligned}$$

where we used the abbreviation $A = R + \alpha^{-2} I$, observing in the last line that $A = A'$. Summing over i yields

$$\frac{\partial}{\partial C_{kl}} E[\|x - Cx\|^2] + \alpha^{-2} \|C\|^2_{fro} = -2\, N\, R_{kl} + 2N\, (CA)_{kl},$$

which in matrix form is

$$\frac{\partial}{\partial C_{kl}} E[\|x - Cx\|^2] + \alpha^{-2} \|C\|^2_{fro} = -2\, N\, R + 2N\, C(R + \alpha^{-2} I),$$

where we re-inserted the expression for A. Setting this to zero yields the claim *1.* stated in the proposition.

The subclaim that $R(R + \alpha^{-2}I)^{-1} = (R + \alpha^{-2}I)^{-1} R$ can be easily seen when R is written by its SVD $R = U\Sigma U'$: $U\Sigma U'(U\Sigma U' + \alpha^{-2}I)^{-1} = U\Sigma U'(U(\Sigma + \alpha^{-2}I)U')^{-1} = U\Sigma U'U(\Sigma+\alpha^{-2}I)^{-1}U' = U\Sigma(\Sigma+\alpha^{-2}I)^{-1}U' = U(\Sigma+\alpha^{-2}I)^{-1}\Sigma U' = \ldots = (R + \alpha^{-2}I)^{-1} R$. This also proves claim *2.*

Claims 3.–5. can be derived from the first claim by elementary arguments.

5.2 Proof of Proposition 6 (Section 3.9.3)

C_δ^{-1} can be written as $US_\delta^{-1}U'$, where the diagonal of S_δ^{-1} is $(s_1^{-1},\ldots,s_l^{-1},\delta^{-1},\ldots,\delta^{-1})'$. Putting $e=\delta^{-1}$ and $S_e := (s_1^{-1},\ldots,s_l^{-1},e,\ldots,e)'$, and similarly $T_e = (t_1^{-1},\ldots,t_m^{-1}, e,\ldots,e)'$, we can express the limit $\lim_{\delta\to 0}(C_\delta^{-1}+B_\delta^{-1}-I)^{-1}$ equivalently as $\lim_{e\to\infty}(US_eU'+VT_eV'-I)^{-1}$. Note that $US_eU'+VT_eV'-I$ is invertible for sufficiently large e. Let $U_{>l}$ be the $N\times(N-l)$ submatrix of U made from the last $N-l$ columns of U (spanning the null space of C), and let $V_{>m}$ be $N\times(N-m)$ submatrix of V made from the last $N-m$ columns of V. The $N\times N$ matrix $U_{>l}(U_{>l})'+V_{>m}(V_{>m})'$ is positive semidefinite. Let $W\Sigma W'$ be its SVD, with singular values in Σ in descending order. Noting that $C^\dagger = U\mathrm{diag}(s_1^{-1},\ldots,s_l^{-1},0,\ldots,0)'U'$, and $B^\dagger = V\mathrm{diag}(t_1^{-1},\ldots,t_m^{-1},0,\ldots,0)'V'$, we can rewrite

$$\begin{aligned}\lim_{\delta\to 0} C_\delta \wedge B_\delta &= \lim_{e\to\infty}(US_eU'+VT_eV'-I)^{-1}\\ &= \lim_{e\to\infty}(C^\dagger+B^\dagger+e\,W\Sigma W'-I)^{-1}\\ &= W\left(\lim_{e\to\infty}(W'C^\dagger W+W'B^\dagger W+e\Sigma-I)^{-1}\right)W'. \qquad (106)\end{aligned}$$

If Σ is invertible, clearly (106) evaluates to the zero matrix. We proceed to consider the case of non-invertible $\Sigma=\mathrm{diag}(\sigma_1,\ldots,\sigma_k,0,\ldots,0)$, where $0\le k<N$.

We derive two auxiliary claims. Let $W_{>k}$ be the $N\times(N-k)$ submatrix of W made from the last $N-k$ columns. *Claim 1:* the $(N-k)\times(N-k)$ matrix $A=(W_{>k})'\,(C^\dagger+B^\dagger-I)\,W_{>k}$ is invertible. We rewrite

$$A=(W_{>k})'C^\dagger W_{>k}+(W_{>k})'B^\dagger W_{>k}-I_{(N-k)\times(N-k)}, \qquad (107)$$

and analyse $(W_{>k})'C^\dagger W_{>k}$. It holds that $(W_{>k})'U_{>l}=0_{(N-k)\times(N-l)}$, because

$$\begin{aligned} W\begin{pmatrix}\sigma_1 & & & & \\ & \ddots & & & \\ & & \sigma_k & & \\ & & & 0 & \\ & & & & \ddots \\ & & & & & 0\end{pmatrix}W' &= U_{>l}(U_{>l})'+V_{>m}(V_{>m})'\\ \Longrightarrow \quad \begin{pmatrix}\sigma_1 & & & & \\ & \ddots & & & \\ & & \sigma_k & & \\ & & & 0 & \\ & & & & \ddots \\ & & & & & 0\end{pmatrix} &= W'U_{>l}(U_{>l})'W+W'V_{>m}(V_{>m})'W\\ \Longrightarrow \quad 0_{(N-k)\times(N-k)} &= (W_{>k})'U_{>l}(U_{>l})'W_{>k}+(W_{>k})'V_{>m}(V_{>m})'W_{>k}\\ \Longrightarrow \quad (W_{>k})'U_{>l} &= 0_{(N-k)\times(N-l)}. \qquad (108)\end{aligned}$$

Let $U_{\leq l}$ be the $N \times l$ submatrix of U made of the first l columns. Because of (108) it follows that

$$\begin{aligned}(W_{>k})' C^\dagger W_{>k} &= (W_{>k})' U \begin{pmatrix} s_1^{-1} & & & & \\ & \ddots & & & \\ & & s_l^{-1} & & \\ & & & 0 & \\ & & & & \ddots \\ & & & & & 0 \end{pmatrix} U' W_{>k} \\ &= (W_{>k})' U_{\leq l} \begin{pmatrix} s_1^{-1} & & \\ & \ddots & \\ & & s_l^{-1} \end{pmatrix} (U_{\leq l})' W_{>k} \end{aligned} \tag{109}$$

The rows of the $(N-k) \times l$ sized matrix $(W_{>k})' U_{\leq l}$ are orthonormal, which also follows from (108). The Cauchy interlacing theorem (stated in Section 3.13.4 in another context) then implies that all singular values of $(W_{>k})' C^\dagger W_{>k}$ are greater or equal to $\min\{s_1^{-1}, \ldots, s_l^{-1}\}$. Since all s_i are smaller or equal to 1, all singular values of $(W_{>k})' C^\dagger W_{>k}$ are greater or equal to 1. The singular values of the matrix $(W_{>k})' C^\dagger W_{>k} - 1/2\, I_{(N-k)\times(N-k)}$ are therefore greater or equal to $1/2$. Specifically, $(W_{>k})' C^\dagger W_{>k} - 1/2\, I_{(N-k)\times(N-k)}$ is positive definite. By a similar argument, $(W_{>k})' B^\dagger W_{>k} - 1/2\, I_{(N-k)\times(N-k)}$ is positive definite. The matrix A from Claim 1 is therefore revealed as the sum of two positive definite matrices, and hence is invertible.

Claim 2: If M is a symmetric $N \times N$ matrix, and the right lower principal submatrix $M_{>k>k} = M(k+1 : N, k+1 : N)$ is invertible, and $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_k, 0, \ldots, 0)$ with all $\sigma_i > 0$, then

$$\lim_{e \to \infty} (M + e\Sigma)^{-1} = \begin{pmatrix} 0 & 0 \\ 0 & M_{>k>k}^{-1} \end{pmatrix}. \tag{110}$$

We exploit the following elementary block representation of the inverse of a symmetric matrix (e.g. [9], fact 2.17.3):

$$\begin{pmatrix} X & Y \\ Y' & Z \end{pmatrix}^{-1} = \begin{pmatrix} V^{-1} & -V^{-1} Y Z^{-1} \\ -Z^{-1} Y' V^{-1} & Z^{-1} Y' V^{-1} Y Z^{-1} + Z^{-1} \end{pmatrix},$$

where Z is assumed to be invertible and $V = (X - Y Z^{-1} Y')$ is assumed to be invertible. Block-structuring $M + e\Sigma$ analogously to this representation, where $M_{>k>k}$ is identified with Z, then easily leads to the claim (110).

Applying Claims 1 and 2 to the limit expression $\lim_{e\to\infty}(W' C^\dagger W + W' B^\dagger W + e\Sigma - I)^{-1}$ in (106), where the matrix M in Claim 2 is identified with $W' C^\dagger W + W' B^\dagger W - I$ and the matrix Z from Claim 2 is identified with the matrix $A = (W_{>k})' (C^\dagger + B^\dagger - I) W_{>k}$ from Claim 1, yields

$$\lim_{e \to \infty} (W' C^\dagger W + W' B^\dagger W + e\Sigma - I)^{-1} = \begin{pmatrix} 0 & 0 \\ 0 & \left((W_{>k})' (C^\dagger + B^\dagger - I) W_{>k}\right)^{-1} \end{pmatrix}$$

which, combined with (106), leads to

$$\lim_{\delta\to 0} C_\delta \wedge B_\delta = W_{>k}\,\left((W_{>k})'\,(C^\dagger + B^\dagger - I)\,W_{>k}\right)^{-1}\,(W_{>k})'. \qquad (111)$$

$W_{>k}$ is an $N \times (N-k)$ size matrix whose columns are orthonormal. Its range is

$$\begin{aligned}
\mathcal{R}(W_{>k}) &= \mathcal{N}(W\Sigma W') \quad = \quad \mathcal{N}(U_{>l}(U_{>l})' + V_{>m}(V_{>m})') \\
&= \mathcal{N}((U_{>l}(U_{>l})') \cap \mathcal{N}(V_{>m}(V_{>m})') \quad = \quad \mathcal{N}((U_{>l})') \cap \mathcal{N}((V_{>m})') \\
&= \mathcal{R}(U_{>l})^\perp \cap \mathcal{R}(V_{>m})^\perp \quad = \quad \mathcal{N}(C)^\perp \cap \mathcal{N}(B)^\perp \\
&= \mathcal{R}(C') \cap \mathcal{R}(B') \quad = \quad \mathcal{R}(C) \cap \mathcal{R}(B), \qquad (112)
\end{aligned}$$

that is, $W_{>k}$ is a matrix whose columns form an orthonormal basis of $\mathcal{R}(C)\cap\mathcal{R}(B)$.

Let $\mathbf{B}_{\mathcal{R}(C)\cap\mathcal{R}(B)}$ be any matrix whose columns form an orthonormal basis of $\mathcal{R}(C)\cap\mathcal{R}(B)$. Then there exists a unique orthonormal matrix T of size $(N-k)\times(N-k)$ such that $\mathbf{B}_{\mathcal{R}(C)\cap\mathcal{R}(B)} = W_{>k}\,T$. It holds that

$$\begin{aligned}
&W_{>k}\,\left((W_{>k})'\,(C^\dagger + B^\dagger - I)\,W_{>k}\right)^{-1}\,(W_{>k})' = \\
&= W_{>k}T\,\left((W_{>k}T)'\,(C^\dagger + B^\dagger - I)\,W_{>k}T\right)^{-1}\,(W_{>k}T)' \\
&= \mathbf{B}_{\mathcal{R}(C)\cap\mathcal{R}(B)}\,\left(\mathbf{B}'_{\mathcal{R}(C)\cap\mathcal{R}(B)}\,(C^\dagger + B^\dagger - I)\,\mathbf{B}_{\mathcal{R}(C)\cap\mathcal{R}(B)}\right)^{-1}\,\mathbf{B}'_{\mathcal{R}(C)\cap\mathcal{R}(B)}, \qquad (113)
\end{aligned}$$

which gives the final form of the claim in the proposition.

For showing equivalence of (33) with (113), I exploit a fact known in matrix theory ([9], Fact 6.4.16): for two real matrices X, Y of sizes $n\times m, m\times l$ the following two condition are equivalent: (i) $(X\,Y)^\dagger = Y^\dagger\,X^\dagger$, and (ii) $\mathcal{R}(X'\,X\,Y) \subseteq \mathcal{R}(Y)$ and $\mathcal{R}(Y\,Y'\,X') \subseteq \mathcal{R}(X')$. Observing that $\mathbf{P}_{\mathcal{R}(C)\cap\mathcal{R}(B)} = \mathbf{B}_{\mathcal{R}(C)\cap\mathcal{R}(B)}\mathbf{B}'_{\mathcal{R}(C)\cap\mathcal{R}(B)}$ and that $\mathbf{B}^\dagger_{\mathcal{R}(C)\cap\mathcal{R}(B)} = \mathbf{B}'_{\mathcal{R}(C)\cap\mathcal{R}(B)}$, setting $X = \mathbf{B}_{\mathcal{R}(C)\cap\mathcal{R}(B)}$ and $Y = \mathbf{B}'_{\mathcal{R}(C)\cap\mathcal{R}(B)}\,(C^\dagger + B^\dagger - I)\mathbf{P}_{\mathcal{R}(C)\cap\mathcal{R}(B)}$ in (33) yields

$$\begin{aligned}
&\left(\mathbf{P}_{\mathcal{R}(C)\cap\mathcal{R}(B)}\,(C^\dagger + B^\dagger - I)\,\mathbf{P}_{\mathcal{R}(C)\cap\mathcal{R}(B)}\right)^{-1} = \\
&\qquad \left(\mathbf{B}'_{\mathcal{R}(C)\cap\mathcal{R}(B)}\,(C^\dagger + B^\dagger - I)\,\mathbf{P}_{\mathcal{R}(C)\cap\mathcal{R}(B)}\right)^\dagger\,\mathbf{B}'_{\mathcal{R}(C)\cap\mathcal{R}(B)},
\end{aligned}$$

where condition (ii) from the abovementioned fact is easily verified. In a second, entirely analog step one can pull apart $\left(\mathbf{B}'_{\mathcal{R}(C)\cap\mathcal{R}(B)}\,(C^\dagger + B^\dagger - I)\,\mathbf{P}_{\mathcal{R}(C)\cap\mathcal{R}(B)}\right)^\dagger$ into

$$\mathbf{B}_{\mathcal{R}(C)\cap\mathcal{R}(B)}\,\left(\mathbf{B}'_{\mathcal{R}(C)\cap\mathcal{R}(B)}\,(C^\dagger + B^\dagger - I)\,\mathbf{B}_{\mathcal{R}(C)\cap\mathcal{R}(B)}\right)^\dagger.$$

Algorithm for Computing $\mathbf{B}_{\mathcal{R}(C)\cap\mathcal{R}(B)}$. Re-using ideas from this proof, a basis matrix $\mathbf{B}_{\mathcal{R}(C)\cap\mathcal{R}(B)}$ can be computed as follows:

1. Compute the SVDs $C = U\mathrm{diag}(s_1,\ldots,s_l,0,\ldots,0)U'$ and $B = V\mathrm{diag}(t_1,\ldots,t_m, 0,\ldots,0)V'$.

2. Let $U_{>l}$ be the submatrix of U made from the last $N-l$ columns in U, and similarly let $V_{>m}$ consist of the last $N-m$ columns in V.

3. Compute the SVD $U_{>l}(U_{>l})' + V_{>m}(V_{>m})' = W\Sigma W'$, where $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_k, 0, \ldots, 0)$.

4. Let $W_{>k}$ be the submatrix of W consisting of the last $N-k$ columns of W. Then $\mathbf{B}_{\mathcal{R}(C)\cap\mathcal{R}(B)} = W_{>k}$.

5.3 Proof of Proposition 7 (Section 3.9.3)

.

By Definition 4, Proposition 6, and Equation (7),

$$\begin{aligned} C \vee B &= \neg(\neg C \wedge \neg B) = I - \lim_{\delta\downarrow 0}\left((\neg C)_\delta^{-1} + (\neg B)_\delta^{-1} - I\right)^{-1} \\ &= I - \lim_{\delta\downarrow 0}\left((I - C^{(\delta)})^{-1} + (I - B^{(\delta)})^{-1} - I\right)^{-1} \\ &= I - \lim_{\delta\downarrow 0}\left((I - R_C^{(\delta)}(R_C^{(\delta)} + I)^{-1})^{-1} + (I - R_B^{(\delta)}(R_B^{(\delta)} + I)^{-1} - I\right)^{-1}. \end{aligned}$$

It is easy to check that $(I - A(A+I)^{-1})^{-1} = I + A$ holds for any positive semidefinite matrix A. Therefore,

$$C \vee B = I - \lim_{\delta\downarrow 0}\left(R_C^{(\delta)} + R_B^{(\delta)} + I\right)^{-1}.$$

Furthermore, for positive semidefinite A, B it generally holds that $I - (A + B + I)^{-1} = (A + B)(A + B + I)^{-1}$, and hence

$$C \vee B = \lim_{\delta\downarrow 0}(R_C^{(\delta)} + R_B^{(\delta)})\,(R_C^{(\delta)} + R_B^{(\delta)} + I)^{-1}.$$

5.4 Proof of Proposition 8 (Section 3.9.3)

Using (32), Proposition 7 and that fact $A^{(\delta)} = R_A^{(\delta)}\,(R_A^{(\delta)} + I)^{-1}$ holds for any conceptor A (which entails $I - A^{(\delta)} = (R_A^{(\delta)} + I)^{-1}$), we derive the claim as follows:

$$\begin{aligned} C \wedge B &= \lim_{\delta\downarrow 0}\left(C_\delta^{-1} + B_\delta^{-1} - I\right)^{-1} \\ &= \lim_{\delta\downarrow 0}\left((\neg\neg C)_\delta^{-1} + (\neg\neg B)_\delta^{-1} - I\right)^{-1} \\ &= \lim_{\delta\downarrow 0}\left((\neg(\neg C)^{(\delta)})^{-1} + (\neg(\neg B)^{(\delta)})^{-1} - I\right)^{-1} \\ &= \lim_{\delta\downarrow 0}\left((I - (\neg C)^{(\delta)})^{-1} + (I - (\neg B)^{(\delta)})^{-1} - I\right)^{-1} \\ &= \lim_{\delta\downarrow 0}\left((R_{\neg C}^{(\delta)} + I) + (R_{\neg B}^{(\delta)}) + I) - I\right)^{-1} \\ &= \lim_{\delta\downarrow 0}(R_{\neg C}^{(\delta)} + R_{\neg B}^{(\delta)} + I)^{-1} \\ &= I - \lim_{\delta\downarrow 0}(I - (R_{\neg C}^{(\delta)} + R_{\neg B}^{(\delta)} + I)^{-1}) \\ &= \neg(\neg C \vee \neg B). \end{aligned}$$

5.5 Proof of Proposition 9 (Section 3.9.4)

Claims 1. – 3. are elementary.

Claim 4a: $\mathcal{R}(C \wedge B) = \mathcal{R}(C) \cap \mathcal{R}(B)$. By definition we have

$$C \wedge B = \mathbf{B}_{\mathcal{R}(C)\cap\mathcal{R}(B)} \left(\mathbf{B}'_{\mathcal{R}(C)\cap\mathcal{R}(B)}(C^\dagger + B^\dagger - I)\mathbf{B}_{\mathcal{R}(C)\cap\mathcal{R}(B)}\right)^{-1} \mathbf{B}'_{\mathcal{R}(C)\cap\mathcal{R}(B)}.$$

Since $\mathbf{B}'_{\mathcal{R}(C)\cap\mathcal{R}(B)}(C^\dagger + B^\dagger - I)\mathbf{B}_{\mathcal{R}(C)\cap\mathcal{R}(B)}$ is invertible and $\mathcal{R}(\mathbf{B}'_{\mathcal{R}(C)\cap\mathcal{R}(B)}) = \mathbb{R}^{\dim(\mathcal{R}(C)\cap\mathcal{R}(B))}$, we have $\mathcal{R}\left((\mathbf{B}'_{\mathcal{R}(C)\cap\mathcal{R}(B)}(C^\dagger + B^\dagger - I)\mathbf{B}_{\mathcal{R}(C)\cap\mathcal{R}(B)})^{-1}\mathbf{B}'_{\mathcal{R}(C)\cap\mathcal{R}(B)}\right) = \mathbb{R}^{\dim(\mathcal{R}(C)\cap\mathcal{R}(B))}$. From this it follows that $\mathcal{R}(C \wedge B) = \mathcal{R}(\mathbf{B}_{\mathcal{R}(C)\cap\mathcal{R}(B)}) = \mathcal{R}(C) \cap \mathcal{R}(B)$.

Claim 5a: $\mathcal{I}(C \wedge B) = \mathcal{I}(C) \cap \mathcal{I}(B)$. We have, by definition,

$$C \wedge B = \mathbf{B}_{\mathcal{R}(C)\cap\mathcal{R}(B)} \left(\mathbf{B}'_{\mathcal{R}(C)\cap\mathcal{R}(B)}(C^\dagger + B^\dagger - I)\mathbf{B}_{\mathcal{R}(C)\cap\mathcal{R}(B)}\right)^{-1} \mathbf{B}'_{\mathcal{R}(C)\cap\mathcal{R}(B)}.$$

For shorter notation put $\mathbf{B} = \mathbf{B}_{\mathcal{R}(C)\cap\mathcal{R}(B)}$ and $X = \mathbf{B}'_{\mathcal{R}(C)\cap\mathcal{R}(B)}(C^\dagger + B^\dagger - I)\mathbf{B}_{\mathcal{R}(C)\cap\mathcal{R}(B)}$. Note (from proof of Proposition 6) that X is invertible. We characterize the unit eigenvectors of $C \wedge B$. It holds that $\mathbf{B}X^{-1}\mathbf{B}'\,x = x$ if and only if $\mathbf{B}X\mathbf{B}'\,x = x$. We need to show that the conjunction $Cx = x$ and $Bx = x$ is equivalent to $(C \wedge B)\,x = \mathbf{B}X^{-1}\mathbf{B}\,x = x$.

First assume that $Cx = x$ and $Bx = x$. This implies $x \in \mathcal{R}(C) \cap \mathcal{R}(B)$ and $C^\dagger\,x = x$ and $B^\dagger\,x = x$, and hence $\mathbf{P}_{\mathcal{R}(C)\cap\mathcal{R}(B)}(C^\dagger + B^\dagger - I)\mathbf{P}_{\mathcal{R}(C)\cap\mathcal{R}(B)}\,x = x$. But $\mathbf{P}_{\mathcal{R}(C)\cap\mathcal{R}(B)}(C^\dagger + B^\dagger - I)\mathbf{P}_{\mathcal{R}(C)\cap\mathcal{R}(B)} = \mathbf{B}X\mathbf{B}'$, thus $(C \wedge B)\,x = x$.

Now assume conversely that not $Cx = x$ or not $Bx = x$.

Case 1: $x \notin \mathcal{R}(C) \cap \mathcal{R}(B)$. Then $\mathbf{P}_{\mathcal{R}(C)\cap\mathcal{R}(B)}(C^\dagger + B^\dagger - I)\mathbf{P}_{\mathcal{R}(C)\cap\mathcal{R}(B)}\,x \neq x$ and hence $\mathbf{B}X\mathbf{B}'\,x \neq x$, which implies $(C \wedge B)\,x \neq x$.

Case 2: $x \in \mathcal{R}(C) \cap \mathcal{R}(B)$. We first show an auxiliary claim: $\|(C^\dagger + B^\dagger - I)\,x\| > \|x\|$. Let $C_0 = C^\dagger - CC^\dagger, B_0 = B^\dagger - BB^\dagger$. C_0 and B_0 are positive semidefinite because the nonzero singular values of $C^\dagger, B^\dagger$ are greater or equal to 1. Furthermore, $CC^\dagger\,x = BB^\dagger\,x = x$. Thus, $(C^\dagger + B^\dagger)\,x = 2Ix + C_0x + B_0x$, i.e. $(C^\dagger + B^\dagger - I)\,x = Ix + C_0x + B_0x$. From not $Cx = x$ or not $Bx = x$ it follows that $C_0x \neq 0$ or $B_0x \neq 0$. We infer

$$\begin{aligned} C_0x \neq 0 \text{ or } B_0x \neq 0 &\implies x'C_0x > 0 \text{ or } x'B_0x > 0 \\ &\implies x'(C_0 + B_0)x > 0 \quad\implies\quad x'(C_0 + B_0)^2x > 0. \end{aligned}$$

This implies $\|Ix + C_0x + B_0x\|^2 = \|x\|^2 + 2x'(C_0 + B_0)x + x'(C_0 + B_0)^2x > \|x\|^2$, or equivalently, $\|(C^\dagger + B^\dagger - I)\,x\| > \|x\|$, the auxiliary claim.

Since $\mathbf{P}_{\mathcal{R}(C)\cap\mathcal{R}(B)}$ preserves vector norm on $\mathcal{R}(C) \cap \mathcal{R}(B)$ and $\mathcal{R}(C^\dagger + B^\dagger - I)\mathbf{P}_{\mathcal{R}(C)\cap\mathcal{R}(B)} \subseteq \mathcal{R}(C) \cap \mathcal{R}(B)$, it follows that $\|\mathbf{P}_{\mathcal{R}(C)\cap\mathcal{R}(B)}(C^\dagger + B^\dagger - I)\mathbf{P}_{\mathcal{R}(C)\cap\mathcal{R}(B)}\,x\| > \|x\|$, hence $(C \wedge B)\,x \neq x$.

Altogether we have that the conjunction $Cx = x$ and $Bx = x$ is equivalent to $(C \wedge B)\,x = x$, which is equivalent to the claim.

Claim 6a: $\mathcal{N}(C \wedge B) = \mathcal{N}(C) + \mathcal{N}(B)$. This follows from *4a* by $\mathcal{N}(C \wedge B) = (\mathcal{R}(C \wedge B))^\perp = (\mathcal{R}(C) \cap \mathcal{R}(B))^\perp = (\mathcal{N}(C)^\perp \cap \mathcal{N}(B)^\perp)^\perp = \mathcal{N}(C) + \mathcal{N}(B)$.

Claims 4b, 5b, 6b: The second statements in *4., 5., 6.* follow from the first statements and *3.*, exploiting de Morgan's rule.

Claim 7 follows from Equation (19).

Claim 8: Let $A = A \wedge C$. Then claim *4.* implies $\mathcal{R}(A) \cap \mathcal{R}(C) = \mathcal{R}(A)$. By Proposition 6 we can write

$$A \wedge C = \mathbf{B}_{\mathcal{R}(A)} \left(\mathbf{B}'_{\mathcal{R}(A)} (C^\dagger + B^\dagger - I) \mathbf{B}_{\mathcal{R}(A)}\right)^{-1} \mathbf{B}'_{\mathcal{R}(A)}.$$

Let $A = USU'$ be the SVD of A, and assume A has rank $k \leq N$, that is, exactly the first k singular values in S are nonzero. Let U_k be the $N \times k$ matrix consisting of the first k columns of U. It holds that $U_k = \mathbf{B}_{\mathcal{R}(A)}$. We obtain

$$\begin{aligned} S = U'AU = & \\ &= U' U_k \left(U'_k(A^\dagger + C^\dagger - I) U_k\right)^{-1} U'_k U \\ &= I_k \left(U'_k(A^\dagger + C^\dagger - I) U_k\right)^{-1} I'_k, \end{aligned} \tag{114}$$

where I_k is the $N \times k$ matrix consisting of the first k columns of I. Let S_k be the $k \times k$ upper left submatrix of S. Then $S_k = \left(U'_k(A^\dagger + C^\dagger - I) U_k\right)^{-1}$ and

$$\begin{aligned} S_k^{-1} &= U'_k(A^\dagger + C^\dagger - I) U_k \\ &= U'_k A^\dagger U_k + U'_k (C^\dagger - I) U_k \\ &= S_k^{-1} + U'_k (C^\dagger - I) U_k, \end{aligned}$$

hence $U'_k (C^\dagger - I) U_k = 0_{k \times k}$ or equivalently, $U'_k C^\dagger U_k = I_{k \times k}$. This implies $\mathcal{R}(A) \subseteq \mathcal{I}(C)$.

Conversely, assume $\mathcal{R}(A) \subseteq \mathcal{I}(C)$. Going through the above line of arguments in reverse order establishes again $S = I_k \left(U'_k(A^\dagger + C^\dagger - I) U_k\right)^{-1} I'_k$ which implies

$$A = \mathbf{B}_{\mathcal{R}(A)} \left(\mathbf{B}'_{\mathcal{R}(A)} (C^\dagger + B^\dagger - I) \mathbf{B}_{\mathcal{R}(A)}\right)^{-1} \mathbf{B}'_{\mathcal{R}(A)}.$$

$\mathcal{R}(A) \subseteq \mathcal{I}(C)$ implies $\mathcal{R}(A) \subseteq \mathcal{R}(C)$, which leads to

$$\begin{aligned} A &= \mathbf{B}_{\mathcal{R}(A) \cap \mathcal{R}(C)} \left(\mathbf{B}'_{\mathcal{R}(A) \cap \mathcal{R}(C)} (C^\dagger + B^\dagger - I) \mathbf{B}_{\mathcal{R}(A) \cap \mathcal{R}(C)}\right)^{-1} \mathbf{B}'_{\mathcal{R}(A) \cap \mathcal{R}(C)} \\ &= A \wedge C. \end{aligned}$$

The dual $A = A \vee C \Leftrightarrow \mathcal{I}(A)^\perp \subseteq \mathcal{N}(C)$ is easily obtained from $A = A \wedge C \Leftrightarrow \mathcal{R}(A) \subseteq \mathcal{I}(C)$ by applying de Morgan's rules and claim *3.*.

Claims 9 – 13 follow from Equation (19). For *Claim 14.* use *11.* and *4.* and *6.* to first establish that $\mathcal{R}(H \wedge G) = \mathcal{I}(H) \cap \mathcal{I}(G)$ and $\mathcal{N}(H \wedge G) = (\mathcal{I}(H) \cap \mathcal{I}(G))^\perp$, from which *14.* follows. *Claim 15* is analog.

5.6 Proof of Proposition 10 (Section 3.9.5)

Claim 1: $\neg\varphi(C,\gamma) = \varphi(\neg C, \gamma^{-1})$. Notation: All of the matrices $C, \neg C, \varphi(C,\gamma)$, $\neg\varphi(C,\gamma), \varphi(\neg C,\gamma^{-1})$ are positive semidefinite and have SVDs with identical principal component matrix U. For any matrix X among these, let US^XU' be its SVD. We write s_i^X for the ith singular value in S^X. We have to show that $s_i^{\neg\varphi(C,\gamma)} = s_i^{\varphi(\neg C,\gamma^{-1})}$. In the derivations below, we use various facts from Proposition 9 and Equation (19) without explicit reference.

Case $0 < \gamma < \infty, 0 < s_i^C < 1$:

$$\begin{aligned} s_i^{\neg\varphi(C,\gamma)} &= 1 - \frac{s_i^C}{s_i^C + \gamma^{-2}\,(1 - s_i^C)} = \frac{1 - s_i^C}{(1 - s_i^C) + \gamma^2\, s_i^C} \\ &= \frac{s_i^{\neg C}}{s_i^{\neg C} + (\gamma^{-1})^{-2}\,(1 - s_i^{\neg C})} = s_i^{\varphi(\neg C,\gamma^{-1})}. \end{aligned}$$

Case $0 < \gamma < \infty, s_i^C = 0$: Using $s_i^C = 0 \Leftrightarrow s_i^{\neg C} = 1$ we have

$$s_i^{\neg\varphi(C,\gamma)} = 1 - s_i^{\varphi(C,\gamma)} = 1 = s_i^{\varphi(\neg C,\gamma^{-1})}.$$

Case $0 < \gamma < \infty, s_i^C = 1$: dual of previous case.

Case $\gamma = 0$: We show $\neg\varphi(C,\gamma) = \varphi(\neg C,\gamma^{-1})$ directly.

$$\begin{aligned} \neg\varphi(C,\gamma) &= \neg\varphi(C,0) = I - \varphi(C,0) = I - \mathbf{P}_{\mathcal{I}(C)} = \mathbf{P}_{\mathcal{I}(C)^\perp} \\ &= \mathbf{P}_{\mathcal{N}(\neg C)^\perp} = \varphi(\neg C,\infty) = \varphi(\neg C,\gamma^{-1}). \end{aligned}$$

Case $\gamma = \infty$: the dual analog.

Claim 2: $\varphi(C,\gamma) \vee \varphi(B,\gamma) = \varphi(C \vee B,\gamma)$.

Case $0 < \gamma < \infty$: Using concepts and notation from Proposition 7, it is easy to check that any conceptor A can be written as

$$A = \lim_{\delta\downarrow 0} R_A^{(\delta)}(R_A^{(\delta)} + I)^{-1}, \tag{115}$$

and its aperture adapted versions as

$$\varphi(A,\gamma) = \lim_{\delta\downarrow 0} R_A^{(\delta)}(R_A^{(\delta)} + \gamma^{-2} I)^{-1}. \tag{116}$$

Using Proposition 7 and (115) we thus have

$$\begin{aligned} C \vee B &= \lim_{\delta\downarrow 0} (R_C^{(\delta)} + R_B^{(\delta)})\,(R_C^{(\delta)} + R_B^{(\delta)} + I)^{-1} && (117) \\ &= \lim_{\delta\downarrow 0} R_{C\vee B}^{(\delta)}(R_{C\vee B}^{(\delta)} + I)^{-1}. && (118) \end{aligned}$$

Furthermore, again by Proposition 7 and by (116),

$$\varphi(C,\gamma) \vee \varphi(B,\gamma) = \lim_{\delta\downarrow 0} (R_{\varphi(C,\gamma)}^{(\delta)} + R_{\varphi(B,\gamma)}^{(\delta)})\,(R_{\varphi(C,\gamma)}^{(\delta)} + R_{\varphi(B,\gamma)}^{(\delta)} + I)^{-1} \tag{119}$$

and

$$\varphi(C \vee B, \gamma) = \lim_{\delta \downarrow 0} (R^{(\delta)}_{C \vee B})(R^{(\delta)}_{C \vee B} + \gamma^{-2} I)^{-1}. \tag{120}$$

Using (17), it follows for any conceptor A that

$$R^{(\delta)}_{\varphi(A,\gamma)} = \gamma^2 \, R_A^{(\delta/(\delta + \gamma^{-2}(1-\delta)))}. \tag{121}$$

Applying this to (119) and observing that $\lim_{\delta \downarrow 0} \delta = 0 = \lim_{\delta \downarrow 0} \delta/(\delta + \gamma^{-2}(1-\delta))$ yields

$$\begin{aligned} \varphi(C,\gamma) \vee \varphi(B,\gamma) &= \lim_{\delta \downarrow 0} (\gamma^2 R^{(\delta)}_C + \gamma^2 R^{(\delta)}_B)\,(\gamma^2 R^{(\delta)}_C + \gamma^2 R^{(\delta)}_B + I)^{-1} \\ &= \lim_{\delta \downarrow 0} (R^{(\delta)}_C + R^{(\delta)}_B)\,(R^{(\delta)}_C + R^{(\delta)}_B + \gamma^{-2} I)^{-1}. \end{aligned} \tag{122}$$

We now exploit the following auxiliary fact which can be checked by elementary means: If $(X^{(\delta)})_\delta$ is a δ-indexed family of positive semidefinite matrices whose eigenvectors are identical for different δ, and similarly the members of the familiy $(Y^{(\delta)})_\delta$ have identical eigenvectors, and if the limits $\lim_{\delta \downarrow 0} X^{(\delta)}(X^{(\delta)} + I)^{-1}$, $\lim_{\delta \downarrow 0} Y^{(\delta)}(Y^{(\delta)} + I)^{-1}$ exist and are equal, then the limits $\lim_{\delta \downarrow 0} X^{(\delta)}(X^{(\delta)} + \gamma^{-2} I)^{-1}$, $\lim_{\delta \downarrow 0} Y^{(\delta)}(Y^{(\delta)} + \gamma^{-2} I)^{-1}$ exist and are equal, too. Putting $X^{(\delta)} = R^{(\delta)}_C + R^{(\delta)}_B$ and $Y^{(\delta)} = R^{(\delta)}_{C \vee B}$, combining (117), (118), (120) and (122) with this auxiliary fact yields $\varphi(C \vee B, \gamma) = \varphi(C,\gamma) \vee \varphi(B,\gamma)$.

Case $\gamma = 0$: Using various findings from Proposition 9 we have

$$\begin{aligned} \varphi(C,0) \vee \varphi(B,0) &= \mathbf{P}_{\mathcal{I}(C)} \vee \mathbf{P}_{\mathcal{I}(B)} = \mathbf{P}_{\mathcal{I}(\mathbf{P}_{\mathcal{I}(C)}) + \mathcal{I}(\mathbf{P}_{\mathcal{I}(B)})} \\ &= \mathbf{P}_{\mathcal{I}(C) + \mathcal{I}(B)} = \mathbf{P}_{\mathcal{I}(C \vee B)} = \varphi(C \vee B, 0). \end{aligned}$$

Case $\gamma = \infty$:

$$\begin{aligned} \varphi(C,\infty) \vee \varphi(B,\infty) &= \mathbf{P}_{\mathcal{R}(C)} \vee \mathbf{P}_{\mathcal{R}(B)} = \mathbf{P}_{\mathcal{I}(\mathbf{P}_{\mathcal{R}(C)}) + \mathcal{I}(\mathbf{P}_{\mathcal{R}(B)})} \\ &= \mathbf{P}_{\mathcal{R}(C) + \mathcal{R}(B)} = \mathbf{P}_{\mathcal{R}(C \vee B)} = \varphi(C \vee B, \infty). \end{aligned}$$

Claim 3: $\varphi(C,\gamma) \wedge \varphi(B,\gamma) = \varphi(C \wedge B, \gamma)$: follows from Claims *1.* and *2.* with de Morgan's law.

Claim 4: $\varphi(C,\gamma) \vee \varphi(C,\beta) = \varphi(C, \sqrt{\gamma^2 + \beta^2})$:

Case $0 < \gamma, \beta < \infty$: Using Proposition 9 and Equations (121), (117), (116),

we obtain

$$\begin{aligned}
\varphi(C,\gamma)\vee\varphi(C,\beta) &= \lim_{\delta\downarrow 0}\left(R^{(\delta)}_{\varphi(C,\gamma)}+R^{(\delta)}_{\varphi(C,\beta)}\right)\left(R^{(\delta)}_{\varphi(C,\gamma)}+R^{(\delta)}_{\varphi(C,\beta)}+I\right)^{-1}\\
&= \lim_{\delta\downarrow 0}\left(\gamma^2 R_C^{(\delta/(\delta+\gamma^{-2}(1-\delta)))}+\beta^2 R_C^{(\delta/(\delta+\beta^{-2}(1-\delta)))}\right)\cdot\\
&\qquad\cdot\left(\gamma^2 R_C^{(\delta/(\delta+\gamma^{-2}(1-\delta)))}+\beta^2 R_C^{(\delta/(\delta+\beta^{-2}(1-\delta)))}+I\right)^{-1}\\
&\overset{(*)}{=} \lim_{\delta\downarrow 0}(\gamma^2+\beta^2)R_C^{(\delta)}((\gamma^2+\beta^2)R_C^{(\delta)}+I)^{-1}\\
&= \lim_{\delta\downarrow 0} R_C^{(\delta)}(R_C^{(\delta)}+(\gamma^2+\beta^2)^{-1}I)^{-1}\\
&= \varphi(C,\sqrt{\gamma^2+\beta^2}),
\end{aligned}$$

where in step (*) we exploit the fact that the singular values of $R_C^{(\delta/(\delta+\gamma^{-2}(1-\delta)))}$ corresponding to eigenvectors whose eigenvalues in C are less than unity are identical to the singular values of $R_C^{(\delta)}$ at the analog positions.

Case $\gamma=0, 0<\beta<\infty$: Using Proposition 7, facts from Proposition 9, and Equation (115), we obtain

$$\begin{aligned}
\varphi(C,0)\vee\varphi(C,\beta) &= \mathbf{P}_{\mathcal{I}(C)}\vee\varphi(C,\beta)\\
&= \lim_{\delta\downarrow 0}\left(R^{(\delta)}_{\mathbf{P}_{\mathcal{I}(C)}}+R^{(\delta)}_{\varphi(C,\beta)}\right)\left(R^{(\delta)}_{\mathbf{P}_{\mathcal{I}(C)}}+R^{(\delta)}_{\varphi(C,\beta)}+I\right)^{-1}\\
&\overset{(*)}{=} \lim_{\delta\downarrow 0} R^{(\delta/(2-\delta))}_{\varphi(C,\beta)}\left(R^{(\delta/(2-\delta))}_{\varphi(C,\beta)}+I\right)^{-1} = \lim_{\delta\downarrow 0} R^{(\delta)}_{\varphi(C,\beta)}\left(R^{(\delta/)}_{\varphi(C,\beta)}+I\right)^{-1}\\
&= \varphi(C,\beta) = \varphi(C,\sqrt{0^2+\beta^2}),
\end{aligned}$$

where step (*) is obtained by observing $\mathcal{I}(C)=\mathcal{I}(\varphi(C,\beta))$ and applying the definition of $R_A^{(\delta)}$ given in the statement of Proposition 7.

Case $\gamma=\infty, 0<\beta<\infty$: the dual analog to the previous case:

$$\begin{aligned}
\varphi(C,\infty)\vee\varphi(C,\beta) &= \mathbf{P}_{\mathcal{R}(C)}\vee\varphi(C,\beta)\\
&= \lim_{\delta\downarrow 0}\left(R^{(\delta)}_{\mathbf{P}_{\mathcal{R}(C)}}+R^{(\delta)}_{\varphi(C,\beta)}\right)\left(R^{(\delta)}_{\mathbf{P}_{\mathcal{R}(C)}}+R^{(\delta)}_{\varphi(C,\beta)}+I\right)^{-1}\\
&\overset{(*)}{=} \lim_{\delta\downarrow 0} R^{(\delta)}_{\mathbf{P}_{\mathcal{R}(C)}}\left(R^{(\delta/)}_{\mathbf{P}_{\mathcal{R}(C)}}+I\right)^{-1}\\
&= \mathbf{P}_{\mathcal{R}(C)} = \varphi(C,\infty) = \varphi(C,\sqrt{\infty^2+\beta^2}),
\end{aligned}$$

where in step (*) I have omitted obvious intermediate calculations.

The cases $0<\gamma<\infty, \beta\in\{0,\infty\}$ are symmetric to cases already treated, and the cases $\gamma,\beta\in\{0,\infty\}$ are obvious.

Claim 5: $\varphi(C,\gamma)\wedge\varphi(C,\beta)=\varphi(C,(\gamma^{-2}+\beta^{-2})^{-2})$: an easy exercise of applying de Morgan's rule in conjunction with Claims *1.* and *4.*

5.7 Proof of Proposition 11 (Section 3.9.6)

1. De Morgan's rules: By Definition 4 and Proposition 8. *2. Associativity:* From Equations (117) and (118) it follows that for any conceptors B, C it holds that $\lim_{\delta \downarrow 0} R^{(\delta)}_{B \vee C} = \lim_{\delta \downarrow 0} R^{(\delta)}_B + R^{(\delta)}_C$. Employing this fact and using Proposition 7 yields associativity of OR. Applying de Morgan's law then transfers associativity to AND. *3. Commutativity and 4. double negation* are clear. *5. Neutrality:* Neutrality of I: Observing that $\mathcal{R}(C) \cap \mathcal{R}(I) = \mathcal{R}(C)$ and $I^\dagger = I$, starting from the definition of $\vee$ we obtain

$$\begin{aligned} C \vee I &= (\mathbf{P}_{\mathcal{R}(C)} C^\dagger \mathbf{P}_{\mathcal{R}(C)})^\dagger \\ &= (\mathbf{P}_{\mathcal{R}(C^\dagger)} C^\dagger \mathbf{P}_{\mathcal{R}(C^\dagger)})^\dagger \quad \text{(by Prop. 9 Nr. 2)} \\ &= (C^\dagger)^\dagger \\ &= C. \end{aligned}$$

Neutrality of 0 can be obtained from neutrality of I via de Morgan's rules.

6. Globality: $C \wedge 0 = 0$ follows immediately from the Definition of $\wedge$ given in 4, observing that $\mathbf{P}_{\mathcal{R}(C) \cap \mathcal{R}(0)} = 0$. The dual $C \vee I = I$ is obtained by applying de Morgan's rule on $C \wedge 0 = 0$.

7. and 8. weighted absorptions follows from Proposition 10 items *4.* and *5.*

5.8 Proof of Proposition 13 (Section 3.9.6)

Let $\leq$ denote the well-known *Löwner* ordering on the set of real $N \times N$ matrices defined by $A \leq B$ if $B - A$ is positive semidefinite. Note that a matrix C is a conceptor matrix if and only if $0 \leq C \leq I$. I first show the following

Lemma 2 *Let $A \leq B$. Then $A^\dagger \geq \mathbf{P}_{\mathcal{R}(A)} B^\dagger \mathbf{P}_{\mathcal{R}(A)}$.*

Proof of Lemma. $A^\dagger$ and $B^\dagger$ can be written as

$$A^\dagger = \lim_{\delta \to 0} \mathbf{P}_{\mathcal{R}(A)} (A + \delta I)^{-1} \mathbf{P}_{\mathcal{R}(A)} \text{ and } B^\dagger = \lim_{\delta \to 0} \mathbf{P}_{\mathcal{R}(B)} (B + \delta I)^{-1} \mathbf{P}_{\mathcal{R}(B)}. \quad (123)$$

From $A \leq B$ it follows that $\mathcal{R}(A) \subseteq \mathcal{R}(B)$, that is, $\mathbf{P}_{\mathcal{R}(A)} \mathbf{P}_{\mathcal{R}(B)} = \mathbf{P}_{\mathcal{R}(A)}$, which in turn yields

$$\mathbf{P}_{\mathcal{R}(A)} B^\dagger \mathbf{P}_{\mathcal{R}(A)} = \lim_{\delta \to 0} \mathbf{P}_{\mathcal{R}(A)} (B + \delta I)^{-1} \mathbf{P}_{\mathcal{R}(A)}. \quad (124)$$

$A \leq B$ entails $A + \delta I \leq B + \delta I$, which is equivalent to $(A + \delta I)^{-1} \geq (B + \delta I)^{-1}$ (see [9], fact 8.21.11), which implies

$$\mathbf{P}_{\mathcal{R}(A)} (B + \delta I)^{-1} \mathbf{P}_{\mathcal{R}(A)} \leq \mathbf{P}_{\mathcal{R}(A)} (A + \delta I)^{-1} \mathbf{P}_{\mathcal{R}(A)},$$

see Proposition 8.1.2 (xii) in [9]. Taking the limits (123) and (124) leads to the claim of the lemma (see fact 8.10.1 in [9]).

Proof of Claim 1. Let A, B be conceptor matrices of size $N \times N$. According to Proposition 15 (which is proven independently of the results stated in Proposition 13), it holds that $A \leq A \vee B$, which combined with the lemma above establishes

$$\mathbf{P}_{\mathcal{R}(A)}(A^\dagger - (A \vee B)^\dagger)\mathbf{P}_{\mathcal{R}(A)} \geq 0,$$

from which it follows that $I + \mathbf{P}_{\mathcal{R}(A)}(A^\dagger - (A \vee B)^\dagger)\mathbf{P}_{\mathcal{R}(A)}$ is positive semidefinite with all singular values greater or equal to one. Therefore, $\mathbf{P}_{\mathcal{R}(A)}(I + A^\dagger - (A \vee B)^\dagger)\mathbf{P}_{\mathcal{R}(A)}$ is positive semidefinite with all nonzero singular values greater or equal to one. Hence $C = \left(\mathbf{P}_{\mathcal{R}(A)} \left(I + A^\dagger - (A \vee B)^\dagger\right) \mathbf{P}_{\mathcal{R}(A)}\right)^\dagger$ is a conceptor matrix. It is furthermore obvious that $\mathcal{R}(C) = \mathcal{R}(A)$.

From $A \leq A \vee B$ it follows that $\mathcal{R}(A) \subseteq \mathcal{R}(A \vee B)$, which together with $\mathcal{R}(C) = \mathcal{R}(A)$ leads to $\mathcal{R}(A \vee B) \cap \mathcal{R}(C) = \mathcal{R}(A)$. Exploiting this fact, starting from the definition of AND in Def. 4, we conclude

$$\begin{aligned}
(A \vee B) \wedge C &= \left(\mathbf{P}_{\mathcal{R}(A \vee B) \cap \mathcal{R}(C)} \left((A \vee B)^\dagger + C^\dagger - I\right) \mathbf{P}_{\mathcal{R}(A \vee B) \cap \mathcal{R}(C)}\right)^\dagger \\
&= \left(\mathbf{P}_{\mathcal{R}(A)} \left((A \vee B)^\dagger + C^\dagger - I\right) \mathbf{P}_{\mathcal{R}(A)}\right)^\dagger \\
&= \left(\mathbf{P}_{\mathcal{R}(A)} \left((A \vee B)^\dagger + \mathbf{P}_{\mathcal{R}(A)} \left(I + A^\dagger - (A \vee B)^\dagger\right) \mathbf{P}_{\mathcal{R}(A)} - I\right) \mathbf{P}_{\mathcal{R}(A)}\right)^\dagger \\
&= (\mathbf{P}_{\mathcal{R}(A)} A^\dagger \mathbf{P}_{\mathcal{R}(A)})^\dagger \;=\; A.
\end{aligned}$$

Proof of Claim 2. This claim is the Boolean dual to claim 1 and can be straightforwardly derived by transformation from claim 1, using de Morgan's rules and observing that $\mathcal{R}(\neg A) = \mathcal{I}(A)^\perp$ (see Prop. 9 item 3), and that $\neg A = I - A$.

5.9 Proof of Proposition 14 (Section 3.10)

Claim 1: $0 \leq A$ is equivalent to A being positive semidefinite, and for a positive semidefinite matrix A, the condition $A \leq I$ is equivalent to all singular values of A being at most one. Both together yield the claim.

Claim 2: Follows from claim 1.

Claim 3: $A \leq B$ iff $-A \geq -B$ iff $I - A \geq I - B$, which is the same as $\neg A \geq \neg B$.

Claim 4: We first show an auxiliary, general fact:

Lemma 3 *Let X be positive semidefinite, and $\mathbf{P}$ a projector matrix. Then*

$$(\mathbf{P}(\mathbf{P}X\mathbf{P} + I)^{-1}\mathbf{P})^\dagger = \mathbf{P}X\mathbf{P} + \mathbf{P}.$$

Proof of Lemma. Let $U = \mathcal{R}(\mathbf{P})$ be the projection space of $\mathbf{P}$. It is clear that $\mathbf{P}X\mathbf{P} + I : U \to U$ and $\mathbf{P}X\mathbf{P} + I : U^\perp \to U^\perp$, hence $\mathbf{P}X\mathbf{P} + I$ is a bijection on U. Also $\mathbf{P}$ is a bijection on U. We now call upon the following well-known property of the pseudoinverse (see [9], fact 6.4.16):

For matrices K, L of compatible sizes it holds that $(KL)^\dagger = L^\dagger K^\dagger$ if and only if $\mathcal{R}(K'KL) \subseteq \mathcal{R}(L)$ and $\mathcal{R}(LL'K) \subseteq \mathcal{R}(K')$.

Observing that $\mathbf{P}$ and $\mathbf{P}X\mathbf{P}+I$ are bijections on $U=\mathcal{R}(\mathbf{P})$, a twofold application of the mentioned fact yields $(\mathbf{P}(\mathbf{P}X\mathbf{P}+I)^{-1}\mathbf{P})^\dagger = \mathbf{P}^\dagger(\mathbf{P}X\mathbf{P}+I)\mathbf{P}^\dagger = \mathbf{P}X\mathbf{P}+\mathbf{P}$ which completes the proof of the lemma.

Now let $A,B\in\mathcal{C}_N$ and $B\leq A$. By Lemma 2, $B^\dagger - \mathbf{P}_{\mathcal{R}(B)}A^\dagger\mathbf{P}_{\mathcal{R}(B)}$ is positive semidefinite, hence $B^\dagger - \mathbf{P}_{\mathcal{R}(B)}A^\dagger\mathbf{P}_{\mathcal{R}(B)}+I$ is positive definite with singular values greater or equal to one, hence invertible. The singular values of $(B^\dagger - \mathbf{P}_{\mathcal{R}(B)}A^\dagger\mathbf{P}_{\mathcal{R}(B)}+I)^{-1}$ are thus at most one, hence $C=\mathbf{P}_{\mathcal{R}(B)}(B^\dagger - \mathbf{P}_{\mathcal{R}(B)}A^\dagger\mathbf{P}_{\mathcal{R}(B)}+I)^{-1}\mathbf{P}_{\mathcal{R}(B)}$ is a conceptor matrix. Obviously $\mathcal{R}(C)=\mathcal{R}(B)$.

Using these findings and Lemma 3 we can now infer

$$\begin{aligned}
A\wedge C &= \left(\mathbf{P}_{\mathcal{R}(A)\cap\mathcal{R}(C)}\left(A^\dagger + C^\dagger - I\right)\mathbf{P}_{\mathcal{R}(A)\cap\mathcal{R}(C)}\right)^\dagger \\
&= \left(\mathbf{P}_{\mathcal{R}(B)}\left(A^\dagger + \left(\mathbf{P}_{\mathcal{R}(B)}\left(B^\dagger - \mathbf{P}_{\mathcal{R}(B)}A^\dagger\mathbf{P}_{\mathcal{R}(B)}+I\right)^{-1}\mathbf{P}_{\mathcal{R}(B)}\right)^\dagger - I\right)\mathbf{P}_{\mathcal{R}(B)}\right)^\dagger \\
&= \left(\mathbf{P}_{\mathcal{R}(B)}\left(A^\dagger + \mathbf{P}_{\mathcal{R}(B)}\left(B^\dagger - \mathbf{P}_{\mathcal{R}(B)}A^\dagger\mathbf{P}_{\mathcal{R}(B)}\right)\mathbf{P}_{\mathcal{R}(B)}\right)\mathbf{P}_{\mathcal{R}(B)}\right)^\dagger \\
&= B.
\end{aligned}$$

Claim 5: This claim is the Boolean dual to the previous claim. It follows by a straightforward transformation applying de Morgan's rules and observing that $\mathcal{R}(\neg B)=\mathcal{I}(B)^\perp$ (see Prop. 9 item 3), and that $\neg A = I-A$.

Claim 6: Let $A\wedge C=B$. Using the notation and claim from Prop. 6 rewrite $A\wedge C = \lim_{\delta\to 0}(C_\delta^{-1}+A_\delta^{-1}-I)^{-1}$. Similarly, obviously we can also rewrite $A=\lim_{\delta\to 0}A_\delta$. Since $C_\delta^{-1}\geq I$, conclude

$$\begin{aligned}
& C_\delta^{-1}+A_\delta^{-1}-I \geq A_\delta^{-1} \\
\iff\ & (C_\delta^{-1}+A_\delta^{-1}-I)^{-1}\leq A_\delta \\
\implies\ & \lim_{\delta\to 0}(C_\delta^{-1}+A_\delta^{-1}-I)^{-1}\leq \lim_{\delta\to 0}A_\delta \\
\iff\ & A\wedge C\leq A,
\end{aligned}$$

where use is made of the fact that taking limits preserves $\leq$ (see [9] fact 8.10.1).

Claim 7: Using the result from the previous claim, infer $A\vee C=B \implies \neg A\wedge\neg C=\neg B \implies \neg A\geq\neg B \implies A\leq B$.

Claim 8: Let $\gamma=\sqrt{1+\beta^2}\geq 1$, where $\beta\geq 0$. By Proposition 10 *4.* we get $\varphi(A,\gamma)=\varphi(A,1)\vee\varphi(A,\beta)=A\vee\varphi(A,\beta)$, hence $A\leq\varphi(A,\gamma)$. The dual version is obtained from this result by using Proposition 5: let $\gamma\leq 1$, hence $\gamma^{-1}\geq 1$. Then $\phi(A,\gamma)\leq\phi(\phi(A,\gamma),\gamma^{-1})=A$.

Claim 9: If $\gamma=0$, then from Proposition 3 it is clear that $\varphi(A,0)$ is the projector matrix on $\mathcal{I}(A)$ and $\varphi(B,0)$ is the projector matrix on $\mathcal{I}(B)$. From $A\leq B$ and the fact that A and B do not have singular values exceeding 1 it is clear that $\mathcal{I}(A)\subseteq\mathcal{I}(B)$, thus $\varphi(A,0)\leq\varphi(B,0)$.

If $\gamma=\infty$, proceed in an analog way and use Proposition 3 to conclude that $\varphi(A,\infty),\varphi(B,\infty)$ are the projectors on $\mathcal{R}(A),\mathcal{R}(B)$ and apply that $A\leq B$ implies $\mathcal{R}(A)\subseteq\mathcal{R}(B)$.

It remains to treat the case $0<\gamma<\infty$. Assume $A\leq B$, that is, there exists a positive semidefinite matrix D such that $A+D=B$. Clearly D cannot have

singular values exceeding one, so D is a conceptor matrix. For (small) $\delta > 0$, let $A^{(\delta)} = (1-\delta)A$. Then $A^{(\delta)}$ can be written as $A^{(\delta)} = R^{(\delta)}(R^{(\delta)}+I)^{-1}$ for a positive semidefinite $R^{(\delta)}$, and it holds that

$$A = \lim_{\delta \to 0} A^{(\delta)},$$

and furthermore

$$A^{(\delta)} \leq A^{(\delta')} \leq A \text{ for } \delta \geq \delta'.$$

Similarly, let $D^{(\delta)} = (1-\delta)D$, with $D^{(\delta)} = Q^{(\delta)}(Q^{(\delta)}+I)^{-1}$, and observe again

$$D = \lim_{\delta \to 0} D^{(\delta)} \quad \text{and} \quad D^{(\delta)} \leq D^{(\delta')} \leq D \text{ for } \delta \geq \delta'.$$

Finally, define $B^{(\delta)} = (1-\delta)B$, where $B^{(\delta)} = P^{(\delta)}(P^{(\delta)}+I)^{-1}$. Then

$$B = \lim_{\delta \to 0} B^{(\delta)} \quad \text{and} \quad B^{(\delta)} \leq B^{(\delta')} \leq B \text{ for } \delta \geq \delta' \quad \text{and} \quad B^{(\delta)} = A^{(\delta)} + D^{(\delta)}.$$

Because of $B^{(\delta)} = A^{(\delta)} + D^{(\delta)}$ we have

$$R^{(\delta)}(R^{(\delta)}+I)^{-1} \leq P^{(\delta)}(P^{(\delta)}+I)^{-1}. \tag{125}$$

We next state a lemma which is of interest in its own right too.

Lemma 4 *For correlation matrices R, P of same size it holds that*

$$R(R+I)^{-1} \leq P(P+I)^{-1} \quad \textit{iff} \quad R \leq P. \tag{126}$$

Proof of Lemma. Assume $R(R+I)^{-1} \leq P(P+I)^{-1}$. By claim *4.* of this proposition, there is a conceptor matrix C such that $P(P+I)^{-1} = R(R+I)^{-1} \vee C$. Since $P(P+I)^{-1} < I$, C has no unit singular values and thus can be written as $S(S+I)^{-1}$, where S is a correlation matrix. Therefore, $P(P+I)^{-1} = R(R+I)^{-1} \vee S(S+I)^{-1} = (R+S)(R+S+I)^{-1}$, hence $P = R+S$, that is, $R \leq P$.

Next assume $R \leq P$, that is, $P = R+S$ for a correlation matrix S. This implies $P(P+I)^{-1} = (R+S)(R+S+I)^{-1} = R(R+I)^{-1} \vee S(S+I)^{-1}$. By claim *6.* of this proposition, $R(R+I)^{-1} \leq P(P+I)^{-1}$ follows. This concludes the proof of the lemma.

Combining this lemma with (125) and the obvious fact that $R^{(\delta)} \leq P^{(\delta)}$ if and only if $\gamma^2 R^{(\delta)} \leq \gamma^2 P^{(\delta)}$ yields

$$\gamma^2 R^{(\delta)}(\gamma^2 R^{(\delta)}+I)^{-1} \leq \gamma^2 P^{(\delta)}(\gamma^2 P^{(\delta)}+I)^{-1}. \tag{127}$$

Another requisite auxiliary fact is contained in the next

Lemma 5 *Let $0 < \gamma < \infty$. If $A = USU' = \lim_{\delta \to 0} R^{(\delta)}(R^{(\delta)}+I)^{-1}$ and for all δ, $R^{(\delta)}$ has a SVD $R^{(\delta)} = U\Sigma^{(\delta)}U'$, then $\varphi(A,\gamma) = \lim_{\delta \to 0} \gamma^2 R^{(\delta)}(\gamma^2 R^{(\delta)}+I)^{-1}$.*

Proof of Lemma. Since all $R^{(\delta)}$ (and hence, all $R^{(\delta)}(R^{(\delta)}+I)^{-1}$ and $\gamma^2 R^{(\delta)}(\gamma^2 R^{(\delta)}+I)^{-1}$) have the same eigenvectors as A, it suffices to show the convergence claim on the level of individual singular values of the concerned matrices. Let $s, s_\gamma, \sigma^{(\delta)}, s^{(\delta)}, s_\gamma^{(\delta)}$ denote a singular value of $A, \varphi(A,\gamma), R^{(\delta)}, R^{(\delta)}(R^{(\delta)}+I)^{-1}, \gamma^2 R^{(\delta)}(\gamma^2 R^{(\delta)}+I)^{-1}$, respectively (all these versions referring to the same eigenvector in U). For convenience I restate from Proposition 3 that

$$s_\gamma = \begin{cases} s/(s+\gamma^{-2}(1-s)) & \text{for} \quad 0<s<1, \\ 0 & \text{for} \quad s=0, \\ 1 & \text{for} \quad s=1. \end{cases}$$

It holds that $s^{(\delta)} = \sigma^{(\delta)}/(\sigma^{(\delta)}+1)$ and $\lim_{\delta\to 0} s^{(\delta)} = s$, and similarly $s_\gamma^{(\delta)} = \gamma^2\sigma^{(\delta)}/(\gamma^2\sigma^{(\delta)}+1)$. It needs to be shown that $\lim_{\delta\to 0} s_\gamma^{(\delta)} = s_\gamma$.
Case $s=0$:

$$\begin{aligned} s=0 \implies \lim_{\delta\to 0} s^{(\delta)} = 0 \implies \lim_{\delta\to 0}\sigma^{(\delta)} = 0 \\ \implies \lim_{\delta\to 0} s_\gamma^{(\delta)} = 0. \end{aligned}$$

Case $s=1$:

$$\begin{aligned} s=1 \implies \lim_{\delta\to 0} s^{(\delta)} = 1 \implies \lim_{\delta\to 0}\sigma^{(\delta)} = \infty \\ \implies \lim_{\delta\to 0} s_\gamma^{(\delta)} = 1. \end{aligned}$$

Case $0<s<1$:

$$\begin{aligned} s=\lim_{\delta\to 0} s^{(\delta)} \implies s=\lim_{\delta\to 0}\sigma^{(\delta)}/(\sigma^{(\delta)}+1) \implies \lim_{\delta\to 0}\sigma^{(\delta)} = s/(1-s) \\ \implies \lim_{\delta\to 0} s_\gamma^{(\delta)} = \frac{\gamma^2 s/(1-s)}{\gamma^2 s/(1-s)+1} = s/(s+\gamma^{-2}(1-s)). \end{aligned}$$

This concludes the proof of the lemma.

After these preparations, we can finalize the proof of claim *9.* as follows. From Lemma 5 we know that

$$\varphi(A,\gamma) = \lim_{\delta\to 0}\gamma^2 R^{(\delta)}(\gamma^2 R^{(\delta)}+I)^{-1}$$

and

$$\varphi(B,\gamma) = \lim_{\delta\to 0}\gamma^2 P^{(\delta)}(\gamma^2 P^{(\delta)}+I)^{-1}.$$

From (127) and the fact that $\leq$ is preserved under limits we obtain $\varphi(A,\gamma) \leq \varphi(B,\gamma)$.

5.10 Proof of Proposition 16 (Section 3.13.4)

We use the following notation for matrix-vector transforms. We sort the entries of an $N \times N$ matrix M into an N^2-dimensional vector $vec\, M$ row-wise (!). That is, $vec\, M(\mu) =$
$M(\lceil \mu/N \rceil, mod_1(\mu, N))$, where the ceiling $\lceil x \rceil$ of a real number x is the smallest integer greater or equal to x, and $mod_1(\mu, N)$ is the modulus function except for arguments of the form (lk, k), where we replace the standard value $mod(lm, m) = 0$ by $mod_1(lm, m) = m$. Conversely, $M(i, j) = vec\, M((i-1)N + j)$.

The Jacobian J_C can thus be written as a $N^2 \times N^2$ matrix $J_C(\mu, \nu) = \partial\, vec\, \dot{C}(\mu) / \partial\, vec\, C(\nu)$. The natural parametrization of matrices C by their matrix elements does not lend itself easily to an eigenvalue analysis. Assuming that a reference solution $C_0 = USU'$ is fixed, any $N \times N$ matrix C is uniquely represented by a parameter matrix P through $C = C(P) = U\,(S + P)\,U'$, with $C(P) = C_0$ if and only if $P = 0$. Conversely, any parameter matrix P yields a unique $C(P)$.

Now we consider the Jacobian $J_P(\mu, \nu) = \partial\, vec\, \dot{P}_C(\mu) / \partial\, vec\, P_C(\nu)$. By using that (i) $\dot{P} = U'\dot{C}U$, (ii) $vec\,(X'YX) = (X \otimes X)'\, vec\, Y$ for square matrices X, Y, and (iii) $\partial\, A\dot{x} / \partial\, Ax = A(\partial\, \dot{x} / \partial\, x)A^{-1}$ for invertible A, (iv) $(U \otimes U)^{-1} = (U \otimes U)'$, one obtains that $J_P = (U \otimes U)' J_C (U \otimes U)$ and hence J_P and J_C have the same eigenvalues.

Using $C_0 = USU'$ and $\dot{C} = (I - C)CDC' - \alpha^{-2}C$ and the fact that diagonal entries in S of index greater than k are zero, yields

$$\begin{aligned} \left.\frac{\partial\, \dot{p}_{lm}}{\partial\, p_{ij}}\right|_{P=0} \;=\; & e_l'\,(I_{ij}U'DUS + SU_k'DUI_{ji} \\ & -I_{ij}SU_k'DUS - SI_{ij}U'DUS - S^2U_k'DUI_{ji} - \alpha^{-2}I_{ij})\; e_m \end{aligned} \quad (128)$$

where e_l is the l-th unit vector and $I_{ij} = e_i\, e_j'$. Depending on how l, m, i, j relate to k and to each other, calculating (128) leads to numerous case distinctions. Each of the cases concerns entries in a specific subarea of J_P. These subareas are depicted in Fig. 48, which shows J_P in an instance with $N = 5, k = 3$.

I will demonstrate in detail only two of these cases (subareas A and B in Fig. 48) and summarize the results of the others (calculations are mechanical).

The case A concerns all entries $J_P(\mu, \nu)$ with $\mu < \nu, \nu \le kN, \mathrm{mod}_1(\nu, N) \le k$. Translating indices μ, ν back to indices l, m, i, j via $J_P(\mu, \nu) = \partial\, vec\, \dot{P}_C(\mu)\,/\,\partial\, vec\, P_C(\nu) = \partial\, \dot{p}_{(\lceil \mu/N \rceil, mod_1(\mu, N))}\,/\,\partial\, p_{(\lceil \nu/N \rceil, mod_1(\nu, N))} = \partial\, \dot{p}_{lm}\,/\,\partial p_{ij}$ yields conditions (i) $i \le k$ (from $i = \lceil \nu/N \rceil$ and $\nu \le kN$), (ii) $j \le k$ (from $j = mod_1(\nu, N) \le k$) and (iii.a) $l < i$ or (iii.b) $l = i \wedge m < j$ (from $\mu < \nu$).

I first treat the subcase (i), (ii), (iii.a). Since $l \neq i$ one has $e_l'\, I_{ij} = 0$ and eqn. (128) reduces to the terms starting with S, leading to

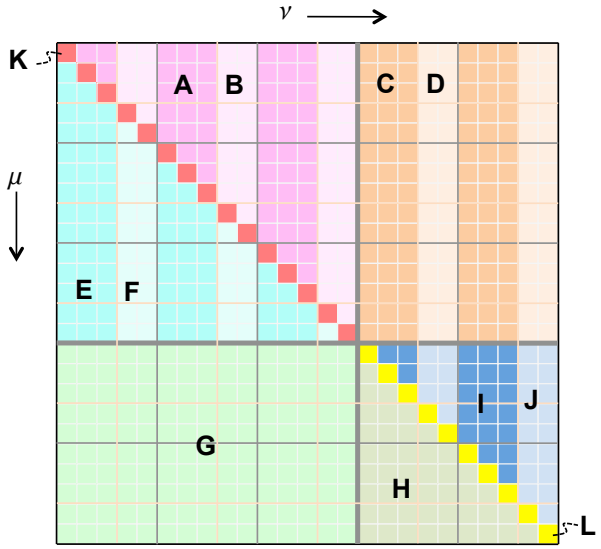

Figure 48: Main case distinction areas for computing values in the matrix J_P. An instance with $N = 5, k = 3$ is shown. Areas are denoted by A, ..., L; same color = same area. J_P has size $N^2 \times N^2$. Its structure is largely organized by a $kN \times kN$-dimensional and a $(N-k)N \times (N-k)N$ submatrix on the diagonal (areas ABEFK and HIJL, respectively). Column/row indices are denoted by μ, ν. Area specifications: A: $\mu < \nu, \nu \leq kN, \mathrm{mod}_1(\nu, N) \leq k$. B: $\mu < \nu, \nu \leq kN, \mathrm{mod}_1(\nu, N) > k$. C: $\nu > kN, \mu \leq kN, \mathrm{mod}_1(\nu, N) \leq k$. D: $\nu > kN, \mu \leq kN, \mathrm{mod}_1(\nu, N) > k$. E: $\mu > \nu, \mu \leq kN, \mathrm{mod}_1(\nu, N) \leq k$. F: $\mu > \nu, \mu \leq kN, \mathrm{mod}_1(\nu, N) > k$. G: $\mu > kN, \nu \leq kN$. H: $\nu > kN, \nu < \mu$. I: $\mu > kN, \mu < \nu, \mathrm{mod}_1(\nu, N) \leq k$. J: $\mu > kN, \mu < \nu, \mathrm{mod}_1(\nu, N) > k$. K: $\mu = \nu \leq kN$. L: $\mu = \nu > kN$.

$$
\begin{aligned}
\left.\frac{\partial \dot{p}_{lm}}{\partial p_{ij}}\right|_{P=0} &= e_l' \left(SU_k' DUI_{ji} - SI_{ij}U'DUS - S^2U_k'DUI_{ji}\right) e_m \\
&= s_l u_l' Du_j \delta_{im} - s_l e_l' I_{ij} U'DUSe_m - s_l^2 u_l' Du_j \delta_{im} \\
&= s_l u_l' Du_j \delta_{im} - s_l^2 u_l' Du_j \delta_{im} \\
&= \begin{cases} 0, & \text{if } i \neq m \quad \text{(subcase A1)} \\ 0, & \text{if } i = m, j \neq l \quad \text{(A2)} \\ \alpha^{-2}, & \text{if } i = m, j = l \quad \text{(A3)}, \end{cases}
\end{aligned}
$$

where u_l is the l-th column in U and $\delta_{im} = 1$ if and only if $i = m$ (else 0) is the Kronecker delta. The value α^{-2} noted for subcase A3 is obtained through $(s_l - s_l^2)u_l' Du_l = (s_l - s_l^2)\tilde{d}_l = \alpha^{-2}$. Note that since $l = i \leq k$ in subcase A3 it holds that $s_l > 1/2$.

Next, in the subcase (i), (ii), (iii.b) one has

$$\left.\frac{\partial\,\dot{p}_{lm}}{\partial\,p_{ij}}\right|_{P=0} = u'_j Du_m s_m + s_l u'_l Du_j \delta_{im} - s_j u'_j Du_m s_m$$
$$-s_l u'_j Du_m s_m - s_l^2 u'_l Du_j \delta_{im} - \alpha^{-2} e'_j e_m$$
$$= (s_l - s_l^2) u'_l Du_j \delta_{im} \quad (\text{since } u'_j Du_m = 0 \text{ and } j \neq m)$$
$$= 0, \quad \text{(A4)} \qquad (129)$$

because assuming $i \neq m$ or $j \neq l$ each null the last expression, and $i = m \wedge j = l$ is impossible because condition (iii.b) would imply $m = j$ contrary to (iii.b).

The case B concerns all entries $J_P(\mu, \nu)$ with $\mu < \nu, \nu \leq kN, mod_1(\nu, N) > k$. Like in case A above, this yields conditions on the P-matrix indices: (i) $i \leq k$, (ii) $j > k$, (iii.a) $l < i$ or (iii.b) $l = i \wedge m < j$.

Again we first treat the subcase (i), (ii), (iii.a). Since $l \neq i$ one has $e'_l I_{ij} = 0$ and eqn. (128) reduces to the terms starting with S, leading to

$$\left.\frac{\partial\,\dot{p}_{lm}}{\partial\,p_{ij}}\right|_{P=0} = e'_l \left(SU'_k DUI_{ji} - SI_{ij}U'DUS - S^2 U'_k DUI_{ji}\right) e_m$$
$$= s_l u'_l Du_j \delta_{im} - s_l e'_l I_{ij} U'DUSe_m - s_l^2 u'_l Du_j \delta_{im}$$
$$= s_l u'_l Du_j \delta_{im} - s_l^2 u'_l Du_j \delta_{im}$$
$$= \begin{cases} 0, & \text{if } i \neq m \quad \text{(B1)} \\ (s_l - s_l^2) u'_l Du_j & \text{if } i = m \quad \text{(B2).} \end{cases}$$

where u_l is the l-th column in U and $\delta_{im} = 1$ if and only if $i = m$ (else 0) is the Kronecker delta. Note that since $l < i \leq k$ it holds that $s_l > 1/2$.

In the subcase (i), (ii), (iii.b) from (128) one obtains

$$\left.\frac{\partial\,\dot{p}_{lm}}{\partial\,p_{ij}}\right|_{P=0} = u'_j Du_m s_m + s_l u'_l Du_j \delta_{im} - s_j u'_j Du_m s_m$$
$$-s_l u'_j Du_m s_m - s_l^2 u'_l Du_j \delta_{im} - \alpha^{-2} e'_j e_m$$
$$= s_m (1 - s_l) u'_j Du_m + (s_l - s_l^2) u'_l Du_j \delta_{im}$$
$$= \begin{cases} s_m(1 - s_l) u'_j Du_m & \text{if } i \neq m \text{ and } m \leq k \quad \text{(B3)} \\ 0 & \text{if } i \neq m \text{ and } m > k \quad \text{(B4)} \\ s_m(1 - s_l) u'_j Du_m + (s_l - s_l^2) u'_l Du_j & \text{if } i = m \quad \text{(B5),} \end{cases}$$

where in the step from the first to the second line one exploits $j > k$, hence $s_j = 0$; and $m < j$, hence $e'_j e_m = 0$. Note that since $l = i \leq k$ it holds that $s_l > 0$; that $m > k$ implies $s_m = 0$ and that $m = i \leq k$ implies $s_m > 0$.

Most of the other cases C – L listed in Fig. 48 divide into subcases like A and B. The calculations are similar to the ones above and involve no new ideas. Table 1 collects all findings. It only shows subcases for nonzero entries of J_P. Fig. 49 depicts the locations of these nonzero areas.

Subcase	**Index range**	**Cell value**
A3	$l = j \leq k;\ i = m \leq k;$	α^{-2}
B2	$l < i = m \leq k < j$	$(s_l - s_l^2)\, u_l' D u_j$
B3	$l = i \leq k < j;\ m \leq j;\ m \neq i$	$s_m(1 - s_l) u_j' D u_m$
B5	$l = i \leq k < j;\ m \leq j;\ m = i$	$s_l(1 - s_l) u_j' D u_l + (s_l - s_l^2) u_l' D u_j$
C1	$l = j \leq k;\ i = m > k$	α^{-2}
D1	$l \leq k;\ i, j > k;\ i = m$	$(s_l - s_l^2)\, u_l' D u_j$
E1	$i = m < l = j \leq k$	α^{-2}
F1	$i = m < l \leq k < j$	$(s_l - s_l^2)\, u_l' D u_j$
J1	$i = l > k;\ m \leq k < j$	$s_m u_j' D u_m$
K1	$i = j = l = m \leq k$	$\alpha^{-2}(1 - 2s_l)/(1 - s_l)$
K2	$i = l;\ j = m > k;\ l \neq m$	$-\alpha^{-2}$
K3	$i = l;\ j = m \leq k;\ l \neq m$	$-\alpha^{-2}\, s_l/(1 - s_m)$
L1	$i = l > k;\ m = j > k$	$-\alpha^{-2}$

Table 1: Values in nonzero areas of the Jacobian J_P.

The eigenvalues of J_P are now readily obtained. First observe that the eigenvalues of a matrix with block structure

$$\begin{pmatrix} K & L \\ 0 & M \end{pmatrix},$$

where K and M are square, are the eigenvalues collected from the principal submatrices K and M. In J_P we therefore only need to consider the leading $kN \times kN$ and the trailing $(N-k)N \times (N-k)N$ submatrices; call them K and M.

M is upper triangular, its eigenvalues are therefore its diagonal values. We thus can collect from M $(N-k)k$ times eigenvalues 0 and $(N-k)^2$ times eigenvalues $-\alpha^{-2}$.

By simultaneous permutations of rows and columns (which leave eigenvalues unchanged) in K we can bring it to the form K_{perm} shown in Fig. 49**B**. A block structure argument as before informs us that the eigenvalues of K_{perm} fall into three groups. There are $k(N-k)$ eigenvalues $-\alpha^{-2}$ (corresponding to the lower right diagonal submatrix of K_{perm}, denoted as area K2), k eigenvalues $\alpha^{-2}(1 - 2s_l)/(1 - s_l)$, where $l = 1, \ldots, k$ earned from the k leading diagonal elements of K_{perm} (stemming from the K1 entries in J_P), plus there are the eigenvalues of $k(k-1)/2$ twodimensional submatrices, each of which is of the form

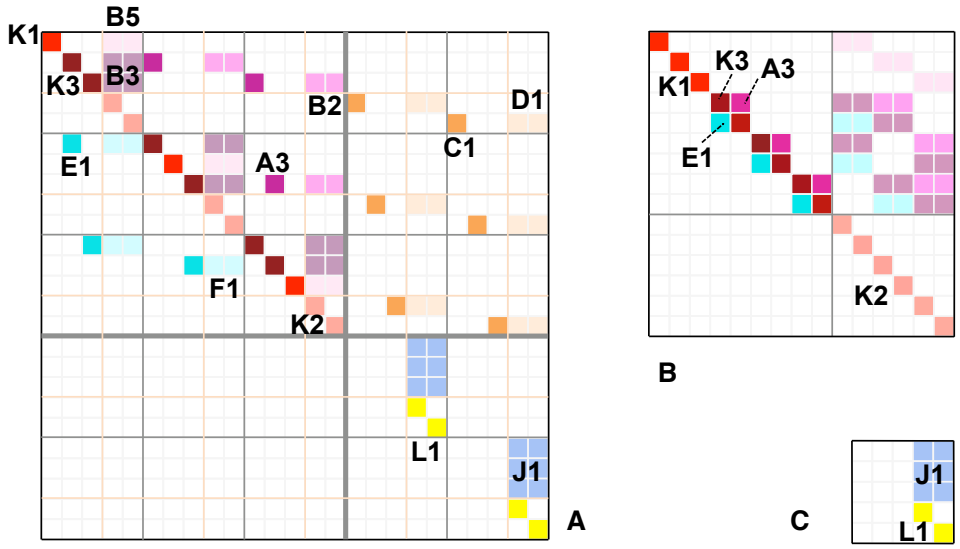

Figure 49: **A.** Nonzero areas in the Jacobian matrix J_P. An instance with $N = 5, k = 3$ is shown. Areas denotations correspond to Table 1. Same color = same area. Values: areas A3, C1, E1: α^{-2}; B2, D1, F1: $(s_l - s_l^2)\, u_l' D u_j$; B3: $s_m(1-s_l)u_j' D u_m$; B5: $s_l(1-s_l)u_j' D u_l + (s_l - s_l^2) u_l' D u_j$; J1: $s_m u_j' D u_m$; K1: $\alpha^{-2}(1-2s_l)/(1-s_l)$; K2, L1: $-\alpha^{-2}$; K3: $-\alpha^{-2}\, s_l/(1-s_m)$. **B.** The left upper principal submatrix re-arranged by simultaneous row/column permutations. **C.** One of the $N \times N$ submatrices C_r from the diagonal of the $(N-k)N \times (N-k)N$ right bottom submatrix of J_P. For explanations see text.

$$K_{l,m} = -\alpha^{-2} \begin{pmatrix} s_l/(1-s_m) & 1 \\ 1 & s_m/(1-s_l) \end{pmatrix},$$

where $l = 1, \ldots, k$; $m < l$. By solving the characteristic polynomial of $K_{l,m}$ its eigenvalues are obtained as

$$\lambda_{1,2} = \frac{\alpha^{-2}}{2} \left(\frac{s_l}{s_m - 1} + \frac{s_m}{s_l - 1} \pm \sqrt{\left(\frac{s_l}{s_m - 1} - \frac{s_m}{s_l - 1} \right)^2 + 4} \right). \tag{130}$$

Summarizing, the eigenvalues of J_P are constituted by the following multiset:

1. $k(N-k)$ instances of 0,
2. $N(N-k)$ instances of $-\alpha^{-2}$,
3. k eigenvalues $\alpha^{-2}(1-2s_l)/(1-s_l)$, where $l = 1, \ldots, k$,
4. $k(k-1)$ eigenvalues which come in pairs of the form given in Eqn. (130).

5.11 Proof of Proposition 18 (Section 3.17)

$$\begin{aligned}
\zeta \models_{\Sigma^{(n)}} \xi \quad &\text{iff} \quad \forall z^{\cdot\wedge 2} : z^{\cdot\wedge 2} \models_{\Sigma^{(n)}} \zeta \to z^{\cdot\wedge 2} \models_{\Sigma^{(n)}} \xi \\
&\text{iff} \quad \forall z^{\cdot\wedge 2} : z^{\cdot\wedge 2} .* (z^{\cdot\wedge 2} + 1)^{-1} \leq \iota(\zeta) \to z^{\cdot\wedge 2} .* (z^{\cdot\wedge 2} + 1)^{-1} \leq \iota(\xi) \\
&\text{iff} \quad \iota(\zeta) \leq \iota(\xi),
\end{aligned}$$

where the last step rests on the fact that all vector components of ζ, ξ are at most 1 and that the set of vectors of the form $z^{\cdot\wedge 2} .* (z^{\cdot\wedge 2}+1)^{-1}$ is the set of nonnegative vectors with components less than 1.

References

[1] The Human Brain Project. Report to the European Commission, HBP Consortium, EPFL Lausanne, 2012. URL: http://www.humanbrainproject.eu/.

[2] L. F. Abbott. Theoretical neuroscience rising. *Neuron*, 60(November 6):489–495, 2008.

[3] D.H. Ackley, G.E. Hinton, and T.J. Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

[4] D. J. Amit, H. Gutfreund, and H. Sompolinsky. Spin-glass models of neural networks. *Phys. Rev. A*, 32:1007–1018, 1985.

[5] J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, and Y . Qin. An integrated theory of the mind. *Psychological Review*, 111(4):1036–1060, 2004.

[6] L. Appeltant, M. C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer. Information processing using a single dynamical node as complex system. *Nature Communications*, 2(468), 2011. DOI: 10.1038/ncomms1476.

[7] A. Babloyantz and C. Lourenço. Computation with chaos: A paradigm for cortical activity. *Proceedings of the National Academy of Sciences of the USA*, 91:9027–9031, 1994.

[8] S. Bader, P. Hitzler, and S. Hölldobler. Connectionist model generation: A first-order approach. *Neurocomputing*, 71(13):2420–2432, 2008.

[9] D. S. Bernstein. *Matrix Mathematics, 2nd Edition.* Princeton Univ. Press, 2009.

[10] R. V. Borges, A. Garcez, and L. C. Lamb. Learning and representing temporal knowledge in recurrent networks. *IEEE Trans. on Neural Networks*, 22(12):2409–2421, 2011.

[11] R.A. Brooks. The whole iguana. In M. Brady, editor, *Robotics Science*, pages 432–456. MIT Press, Cambridge, Mass., 1989.

[12] M. Brown and C. Harris. *Neurofuzzy Adaptive Modelling and Control.* Prentice Hall, 1994.

[13] M. Buehner and P. Young. A tighter bound for the echo state property. *IEEE Transactions on Neural Networks*, 17(3):820– 824, 2006.

[14] F. S. Chance and L. F. Abbott. Divisive inhibition in recurrent networks. *Network: Comput. Neural Syst.*, 11:119–129, 2000.

[15] S. P. Chatzis. Hidden Markov models with nonelliptically contoured state densities. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 32(12):2297 – 2304, 2010.

[16] A. Clark. Whatever next? Predictive brains, situated agents, and the future of cognitive science. *Behavioral and Brain Sciences*, pages 1–86, 2012.

[17] A. Coates, P. Abbeel, and A. Y. Ng. Learning for control from multiple demonstrations. In *Proc. 25th ICML, Helsinki*, 2008.

[18] A. M. Collins and M. r. Quillian. Retrieval time from semantic memory. *Journal of verbal learning and verbal behavior*, 8(2):240–247, 1969.

[19] P. Dominey, M. Arbib, and J.-P. Joseph. A model of corticostriatal plasticity for learning oculomotor associations and sequences. *Journal of Cognitive Neuroscience*, 7(3):311–336, 1995.

[20] P. F. Dominey. From sensorimotor sequence to grammatical construction: Evidence from simulation and neurophysiology. *Adaptive Behaviour*, 13(4):347–361, 2005.

[21] R. Douglas and T. Sejnowski. Future challenges for the sciene and engineering of learning: Final workshop report. Technical report, National Science Foundation, 2008.

[22] G.L. Drescher. *Made-up Minds: A Constructivist Approach to Artificial Intelligence*. MIT Press, Cambridge, Mass., 1991.

[23] D. Durstewitz, J. K. Seamans, and T. J. Sejnowski. Neurocomputational models of working memory. *Nature Neuroscience*, 3:1184–91, 2000.

[24] C. Eliasmith. A unified approach to building and controlling spiking attractor networks. *Neural Computation*, 17:1276–1314, 2005.

[25] C. Eliasmith. Attractor network. *Scholarpedia*, 2(10):1380, 2007.

[26] C. Eliasmith, Stewart T. C., Choo X., Bekolay T., Tang Y. DeWolf T., and D. Rasmussen. A large-scale model of the functioning brain. *Science*, 338(6111):1202–1205, 2012.

[27] K. Fan and G. Pall. Imbedding conditions for Hermitian and normal matrices. *Canad. J. Math.*, 9:298–304, 1957.

[28] B. Farhang-Boroujeny. *Adaptive Filters: Theory and Applications*. Wiley, 1998.

[29] J.A. Fodor and Z.W. Pylyshin. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28:3–71, 1988.

[30] W. J. Freeman. Definitions of state variables and state space for brain-computer interface. part 1: Multiple hierarchical levels of brain function. *Cognitive Neurodynamics*, 1(1):3–14, 2007.

[31] W. J. Freeman. Definitions of state variables and state space for brain-computer interface. part 2. extraction and classification of feature vectors. *Cognitive Neurodynamics*, 1(2):85–96, 2007.

[32] R. M. French. Catastrophic interference in connectionist networks. In L. Nadel, editor, *Encyclopedia of Cognitive Science*, volume 1, pages 431–435. Nature Publishing Group, 2003.

[33] K. Friston. A theory of cortical response. *Phil. Trans. R. Soc. B*, 360:815–836, 2005.

[34] M. Galtier, O. D. Faugeras, and P. C. Bressloff. Hebbian learning of recurrent connections: a geometrical perspective. *Neural Computation*, 24(9):2346–2383, 2012.

[35] T. Gedeon and D. Arathorn. Convergence of map seeking circuits. *J. Math. Imaging Vis.*, 29:235–248, 2007.

[36] W. Gerstner, H. Sprekeler, and D. Deco. Theory and simulation in neuroscience. *Science*, 338(5 Oct):60–65, 2012.

[37] J. Goguen and R. Burstall. Institutions: Abstract model theory for specification and programming. *J. of the ACM*, 39(1):95–146, 1992.

[38] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):855 – 868, 2009.

[39] A. Graves and J. Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Proc. NIPS 2008*. MIT Press, 2008.

[40] S. Grillner. Biological pattern generation: The cellular and computational logic of networks in motion. *Neuron*, 52:751–766, 2006.

[41] C. Gros and G. Kaczor. Semantic learning in autonomously active recurrent neural networks. *Logic Journal of the IGPL*, 18(5):686–704, 2010.

[42] S. Grossberg. Linking attention to learning, expectation, competition, and consciousness. In L. Itti, G. Rees, and J. Tsotsos, editors, *Neurobiology of attention*, chapter 107, pages 652–662. San Diego: Elsevier, 2005.

[43] S. Grossberg. Adaptive resonance theory. *Scholarpedia*, 8(5):1569, 2013.

[44] S. Harnad. The symbol grounding problem. *Physica*, D42:335–346, 1990.

[45] D. O. Hebb. *The Organization of Behavior*. New York: Wiley & Sons, 1949.

[46] M. Hermans and B. Schrauwen. Recurrent kernel machines: Computing with infinite echo state networks. *Neural Computation*, 24(1):104–133, 2012.

[47] G. E. Hinton and R. R. Salakuthdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(July 28):504–507, 2006.

[48] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc. NatL Acad. Sci. USA*, 79:2554–2558, 1982.

[49] O. Houdé and N. Tzourio-Mazoyer. Neural foundations of logical and mathematical cognition. *Nature Reviews Neuroscience*, 4(June 2003):507–514, 2003.

[50] A. J. Ijspeert. Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks*, 21:642–653, 2008.

[51] T. R. Insel, S. C. Landis, and F. S. Collins. The NIH BRAIN initiative. *Science*, 340(6133):687–688, 2013.

[52] M. Ito and J. Tani. Generalization in learning multiple temporal patterns using RNNPB. In *Neural Information Processing*, number 3316 in LNCS, pages 592–598. Springer Verlag, 2004.

[53] H. Jaeger. Identification of behaviors in an agent's phase space. Arbeitspapiere der GMD 951, GMD, St. Augustin, 1995.

[54] 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.

[55] H. Jaeger. Reservoir self-control for achieving invariance against slow input distortions. technical report 23, Jacobs University Bremen, 2010.

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

[57] H. Jaeger, M. Lukosevicius, D. Popovici, and U. Siewert. Optimization and applications of echo state networks with leaky integrator neurons. *Neural Networks*, 20(3):335–352, 2007.

[58] T. Kohonen and T. Honkela. Kohonen network. In *Scholarpedia*, volume 2, page 1568. 2007.

[59] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1106–1114, 2012.

[60] M. Kudo, J. Toyama, and M. Shimbo. Multidimensional curve classification using passing-through regions. *Pattern Recognition Letters*, 20(11):1103–1111, 1999.

[61] R. Laje and D. V. Buonomano. Robust timing and motor patterns by taming chaos in recurrent neural networks. *Nature Neuroscience*, 16(7):925–933, 2013.

[62] G. Lakoff. *Women, fire, and dangerous things: What categories reveal about the mind*. University of Chicago, 1987.

[63] G. Lakoff. Cognitive models and prototype theory. In I. Margolis and S. Laurence, editors, *Concepts: Core Readings*, chapter 18, pages 391–422. Bradford Books / MIT Press, 1999.

[64] G. Lakoff and R. E. Nunez. *Where mathematics comes from: How the embodied mind brings mathematics into being*. Basic Books, 2000.

[65] L. C. Lamb. The grand challenges and myths of neural-symbolic computation. In L. De Raedt, B. Hammer, P. Hitzler, and W. Maass, editors, *Recurrent Neural Networks- Models, Capacities, and Applications*, number 08041 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2008. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

[66] Y. LeCun, L. Bottou, J. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf.

[67] F. Lehmann. *Semantic networks in artificial intelligence*. Elsevier Science, 1992.

[68] L. Lukic, J. Santos-Victor, and A. Billard. Learning coupled dynamical systems from human demonstration for robotic eye-arm-hand coordination. In *Proc. IEEE-RAS International Conference on Humanoid Robots, Osaka 2012*, 2012.

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

[70] W. Maass, P. Joshi, and E. Sontag. Computational aspects of feedback in neural circuits. *PLOS Computational Biology*, 3(1):1–20, 2007.

[71] G. Manjunath and H. Jaeger. Echo state property linked to an input: Exploring a fundamental characteristic of recurrent neural networks. *Neural Computation*, 25(3):671–696, 2013.

[72] N. M. Mayer and M. Browne. Echo state networks and self-prediction. In *Biologically Inspired Approaches to Advanced Information Technology*, volume 3141 of *LNCS*, pages 40–48. Springer Verlag Berlin / Heidelberg, 2004.

[73] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bull. of Mathematical Biophysics*, 5:115–133, 1943.

[74] D. L. Medin and L. J. Rips. Concepts and categories: Memory, meaning, and metaphysics. In K. J. Holyoak and R. G. Morrison, editors, *The Cambridge Handbook of Thinking and Reasoning*, chapter 3, pages 37–72. Cambridge University Press, 2005.

[75] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[76] M. Negrello and F. Pasemann. Attractor landscapes and active tracking: The neurodynamics of embodied action. *Adaptive Behaviour*, 16:196 – 216, 2008.

[77] K. Obermayer, H. Ritter, and K. Schulten. A principle for the formation of the spatial structure of cortical feature maps. *Proc. of the National Academy of Sciences of the USA*, 87:8345–8349, 1990.

[78] E. Oja. A simplified neuron model as a principal component analyzer. *J. Math. Biol.*, 15:267–273, 1982.

[79] C. Orsenigo and C. Vercellis. Combining discrete SVM and fixed cardinality warping distances for multivariate time series classification. *Pattern Recognition*, 43(11):3787–3794, 2010.

[80] G. Palm. On associative memory. *Biol. Cybernetics*, 36(1):19–31, 1980.

[81] R. Pfeifer and Ch. Scheier. *Understanding Intelligence*. MIT Press, 1999.

[82] G. Pinkas. Propositional non-monotonic reasoning and inconsistency in symmetric neural networks. In *Proc. 12th international joint conference on Artificial intelligence - Volume 1*, pages 525–530, 1991.

[83] J. B. Pollack. Recursive distributed representations. *Artificial Intelligence*, 46(1-2):77–105, 1990.

[84] L. Qi. Symmetric nonnegative tensors and copositive tensors. arxiv.org/pdf/1211.5642, 2012.

[85] M. R. Quillain. Word concepts: A theory and simulation of some basic semantic capabilities. *Behavioral Science*, 12(5):410–430, 1967.

[86] F. Rabe. *Representing Logics and Logic Translations*. Phd thesis, School of Engineering and Science, Jacobs University Bremen, 2008.

[87] M. I. Rabinovich, R. Huerta, P. Varona, and V. S. Afraimovich. Transient cognitive dynamics, metastability, and decision making. *PLOS Computational Biology*, 4(5):e1000072, 2008.

[88] F. R. Reinhart and J. J. Steil. Recurrent neural associative learning of forward and inverse kinematics for movement generation of the redundant pa-10 robot. In A. Stoica, E. Tunsel, T. Huntsberger, T. Arslan, S. Vijayakumar, and A. O. El-Rayis, editors, *Proc. LAB-RS 2008, vol. 1*, pages 35–40, 2008.

[89] R. F. Reinhart, A. Lemme, and J. J. Steil. Representation and generalization of bi-manual skills from kinesthetic teaching. In *Proc. of IEEE-RAS International Conference on Humanoid Robots, Osaka, 2012*, in press.

[90] R. F. Reinhart and J. J. Steil. A constrained regularization approach for input-driven recurrent neural networks. *Differential Equations and Dynamical Systems*, 19(1–2):27–46, 2011 (2010 online pre-publication). DOI 10.1007/s12591-010-0067-x.

[91] J. J. Rodriguez, C. J. Alonso, and J. A. Maestro. Support vector machines of interval-based features for time series classification. *Knowledge-Based Systems*, 18(4-5):171–178, 2005.

[92] J. S. Rothman, L. Cathala, V. Steuber, and R. A. Silver. Synaptic depression enables neuronal gain control. *Nature*, 457(19 Feb):1015–1018, 2009.

[93] G. Schöner, M. Dose, and C. Engels. Dynamics of behavior: theory and applications for autonomous robot architectures. *Robotics and Autonomous Systems*, 16(2):213–246, 1995.

[94] G. Schöner and J. A. Kelso. Dynamic pattern generation in behavioral and neural systems. *Science*, 239(4847):1513–1520, 1988.

[95] J.R. Searle. Minds, brains, and programs. *The Behavioral and Brain Sciences*, 3:417–457, 1980.

[96] L. Shastri. Advances in Shruti – a neurally motivated model of relational knowledge representation and rapid inference using temporal synchrony. *Artificial Intelligence*, 11:79–108, 1999.

[97] K. R. Sivaramakrishnan, K. Karthik, and C. Bhattacharyya. Kernels for large margin time-series classification. In *Proc. IJCNN 2007*, pages 2746 – 2751, 2007.

[98] L.B. Smith and E. Thelen, editors. *A Dynamic Systems Approach to Development: Applications*. Bradford/MIT Press, Cambridge, Mass., 1993.

[99] T. Strauss, W. Wustlich, and R. Labahn. Design strategies for weight matrices of echo state networks. *Neural Computation*, 24(12):3246–3276, 2012.

[100] D. Sussillo and L. Abbott. Transferring learning from external to internal weights in echo-state networks with sparse connectivity. *PLoS ONE*, 7(5):e37372, 2012.

[101] D. Sussillo and O Barak. Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Computation*, 25(3):626–649, 2013.

[102] M. Timme, F. Wolf, and Th. Geisel. Unstable attractors induce perpetual synchronization and desynchronization. *Chaos*, 13:377–387, 2003. http://arxiv.org/abs/cond-mat/0209432.

[103] I. Tsuda. Towards an interpretation of dynamic neural activity in terms of chaotic dynamical systems. *Behavioural and Brain Sciences*, 24(5):793–810, 2001.

[104] F. van der Velde and M. de Kamps. Neural blackboard architectures of combinatorial structures in cognition. *Behavioural and Brain Sciences*, 29(1):37–70, 2006.

[105] T. van Gelder. The dynamical hypothesis in cognitive science. *Behavioural and Brain Sciences*, 21(5):615–628, 1998.

[106] D. Verstraeten. *Reservoir Computing: Computation with Dynamical Systems*. Phd thesis, Electronics and Information Systems, University of Ghent, 2009.

[107] D.M. Wolpert and M. Kawato. Multiple paired forward and inverse models for motor control. *Neural Networks*, 11(7-8):1317–1330, 1998.

[108] F. wyffels, J. Li, T. Waegeman, B. Schrauwen, and H. Jaeger. Frequency modulation of large oscillatory neural networks. *Biological Cybernetics*, published online Feb 2014.

[109] Y. Yao and W.J. Freeman. A model of biological pattern recognition with spatially chaotic dynamics. *Neural Networks*, 3(2):153–170, 1990.

[110] I. B. Yildiz, H. Jaeger, and S. J. Kiebel. Re-visiting the echo state property. *Neural Networks*, 35:1–20, 2012.

[111] S. K. U. Zibner, C. Faubel, I. Iossifidis, and G. Schöner. Dynamic neural fields as building blocks of a cortex-inspired architecture for robotic scene representation. *IEEE Trans. on Autonomous Mental Development*, 3(1):74–91, 2011.