



HAL
open science

Transfer between long-term and short-term memory using Conceptors

Anthony Strock, Nicolas P. Rougier, Xavier Hinaut

► **To cite this version:**

Anthony Strock, Nicolas P. Rougier, Xavier Hinaut. Transfer between long-term and short-term memory using Conceptors. 2020. hal-02494493v1

HAL Id: hal-02494493

<https://inria.hal.science/hal-02494493v1>

Preprint submitted on 10 Mar 2020 (v1), last revised 13 Nov 2020 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Transfer between long-term and short-term memory using Conceptors

Anthony Strock^{2,1,3}, Nicolas P. Rougier^{1,2,3,†} and Xavier Hinaut^{1,2,3,†,*}

¹Inria Bordeaux Sud-Ouest, Talence, France

²LaBRI, Université de Bordeaux, CNRS UMR 5800, Talence, France

³IMN, Université de Bordeaux, CNRS UMR 5293, Bordeaux, France

† Equal contribution, *Corresponding author: xavier.hinaut@inria.fr

Abstract

We introduce a recurrent neural network model of working memory combining short-term and long-term components. The short-term component is modelled using a gated reservoir model that is trained to hold a value from an input stream when a gate signal is on. The long-term component is modelled using conceptors in order to store inner temporal patterns (that corresponds to values). We combine these two components to obtain a model where information can go from long-term memory to short-term memory and vice-versa and we show how standard operations on conceptors allow to combine long-term memories and describe their effect on short-term memory.

1 Introduction

The reservoir computing (RC) paradigm [9] is a peculiar and economic way to train a recurrent neural network (RNN) because only the output layer is modified while the input and recurrent layers are kept unmodified. Such RNNs are called *reservoirs* because they provide a pool of non-linear computations based on inputs. Many variants (such as Echo State Networks [8] and Liquid State Machine [15]), along with specific extensions of this RC paradigm have been proposed since its initial stance by [8] (for a review see [14]), including implementations in various hardware like DNA- or laser-based ones (see [25] for a recent review on physical reservoirs). A recent and major enhancement of the RC paradigm has been proposed by Jaeger [10], called *Conceptors* (see Figure 1 that introduces the main concepts). Intuitively, a conceptor represents a subspace of internal states of a RNN, e.g. the trajectory of a reservoir when fed by some input. This representation can later be used for extending the capacity of the original model. For instance it can be used to recognize temporal patterns [10, 1, 2, 6], or to store and retrieve multiple arbitrary temporal patterns from a single RNN [10, 11]. These conceptors have been recently used in a number of different works. For instance, Mossakowski et al [20] proposed an implementation of fuzzy logic based on conceptors, Liu et al [13] used conceptors for online learning of sentence representations and He and Jaeger [7] proposed a general way to use conceptors during the learning of multiple tasks thanks to a reduction of interference between tasks by virtue of internal state space segregation.

Conceptors yield several advantages when compared to classical reservoirs, as Jaeger demonstrated it in his seminal paper [10]: conceptors provides symbolic operations on the latent space of the different input patterns. It is worth to be mentioned that such property were also exploited in the deep learning community and provided surprising results. For instance, in natural language processing (NLP),

Mikolov et al [18] showed that arithmetic operations such as “*king – men + woman*” give a vector similar to “*queen*”. More recently, Brock et al [4] proposed a method to edit global image features based on operations performed on the latent space of generative adversarial networks (GANs). Conceptors provide similar logical operations but in the framework of the reservoir computing paradigm. For instance, Jaeger [10] proposes an operator that quantifies if a stimulus is similar to an already known conceptor. By associating one conceptor per class, it is possible to measure if a stimulus belongs to a class (positive evidence) or none (negative evidence). Beyond logical operations, linear combinations of conceptors allow to implement continuous morphing between set of states: they were used to create morphing between two time series corresponding to the extended interpolation of the time series (e.g. a morphing between two sine-waves with different frequency is a sine-wave with an intermediate frequency).

Hypotheses on the interaction between working memory (WM) and long-term memory (LTM) has been explored in computational neuroscience models [22]. Moreover, the ability to store transient internal dynamics in long-term memory and to be able to retrieve these dynamics later on when needed is an attractive principle for biology. It is to be related to some recent experimental observations suggesting that, in some specific cases¹ [23], the maintenance of information is not uniform between the time of acquisition and the time of use. More specifically, it has been shown that the information cannot be reliably decoded from neural activity between these two times while it can be decoded at time of use. Some authors [19, 17, 16] have thus suggested the existence of a mechanism to temporarily store information in synaptic weights (instead of being stored in neural activity). In this context, conceptors might provide a plausible explanation of such a transfer.

Based on the original RC paradigm, we have shown in [24] how a reservoir with feedback connections can implement a gated working memory, i.e. a generic mechanism to maintain information at a given time (corresponding to when the gate is on). This study shows that a reservoir, using a gating signal, can faithfully memorize triggered inputs for a while based on a stream of continuous values (i.e. working memory property). This model gives account on two important facts from neurosciences working memory (WM) studies: (1) the model is functionally a closed system when maintaining an information but it is physically an open system. This means the model can maintain information even when fed with a strong and continuous disturbing input. (2) Memory is encoded in the dynamics of the models and information can be maintained without trace of sustained activity. In this model, the maintenance of information in output unit(s) is remarkably precise for long periods of time.

However, this model suffers from its cardinal property: it is able to maintain any information with great precision because it is not backed up by a long term memory component; while in biology, what we hold in WM memory is directly influenced by our experience and perception. Said differently, the current model is a quasi-perfect line-attractor, while it could be interesting to build a discrete line attractor model. For example, if we fed the model with a series of random scalar values with 10^{-3} precision, it may be desirable for the model to approximate (or discretize) the value to be maintained with only 10^{-1} precision. This might seem a trivial operation to be performed but it is actually harder than it seems because the robustness of the model is measured to the extent it is close to a formal line attractor. This is one of the reasons why we turned ourselves towards conceptors, to enable our WM model to be influenced by long-term memory.

In the present work, we introduce a link between short-term and long-term memory by combining two approaches: (1) a gated reservoir maintaining short-term information and (2) several conceptors maintaining long-term information. First, we introduce the transfer mechanisms between long-term and short-term memory. Then, we study how these mechanisms allow for a more stable version of the

¹For instance, when some information to be maintained in working memory is known to be useful only later on.

short-term memory and how it allows to add a prior to the values that must be maintained in short-term memory. Finally, we explore the nature of operations carried on by conceptors, the different ways to combine memories and how this modifies short-term memory.

2 Methods

2.1 Conceptors overview

Compared to classical ESNs, conceptors represent a new training paradigm. In Figure 1, we show how conceptors can be used to store temporal patterns. The training is performed in two steps and uses two models (N and N'). Let us consider a reservoir N' that receives as input the sequence M and that is trained (readout weights) to produce the sequence M as output (auto-encoder). Let us now consider another reservoir N that does not receive any input but is trained (internal weights) such as to match N' internal activity (see figure 1). If we now read the internal activity of N using the read-out weights of N' , we obtain the sequence M . Said differently, N has learnt implicitly to spontaneously produce the sequence M . A very similar idea can actually be found in the full-FORCE training algorithm [5] where internal weights of a recurrent neural network are trained in order for the activity of its neurons to match the ones of another recurrent neural network that receives as input more information than the former one (e.g. the desired output) Jaeger introduced a second idea based on the following observation: if a recurrent neural network is periodically stimulated with a sequence M , it evolves in a different region of space compared to when it is periodically stimulated with another sequence M' (because we are in high dimensional space). Consequently, in order to have the possibility to generate two distinct patterns M and M' , Jaeger proposes to train the model N to match the activity of model N' when it receives sequence M or M' . In such a scenario, N will end up following a mean trajectory, in between the trajectory where N' receives M as input and where N' receives M' as input. However, in light of the preliminary observation concerning the segregation of spaces for sequence M and M' , it is possible to disentangle the activity within N by projecting it to relevant sub-spaces. These sub-spaces can be identified in N' when it receives M or M' respectively. A conceptor corresponds exactly to these projections. In that context, these conceptors can be considered as long-term memories of the temporal patterns because they can be stored and reactivated later with negligible loss of recall/precision. More generally, conceptors can be considered as long-term memories of sets/subspaces of internal states.

2.2 Models

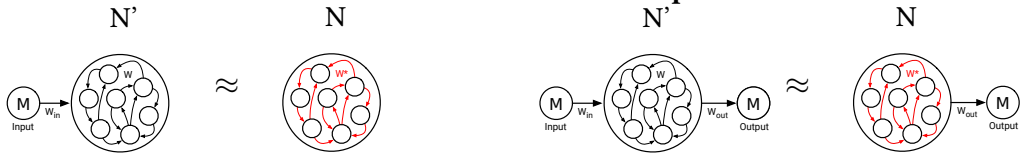
Echo State Networks (ESN)

In this work we consider Echo State Networks (ESN) with feedback from readout units to the reservoir. The system is described by the following update equations:

$$\begin{aligned} x[n] &= \tanh(W_{in}u[n] + W(x[n-1] + \xi) + W_{fb}(y[n-1])) \\ y[n] &= W_{out}x[n] \end{aligned} \tag{1}$$

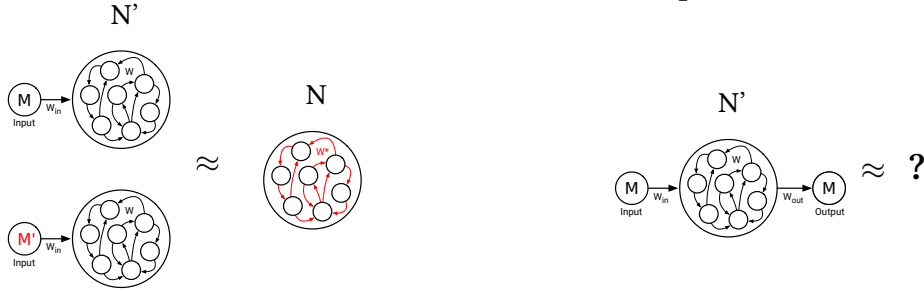
where $u[n]$, $x[n]$ and $y[n]$ are respectively the input, the reservoir and the output at time n . W , W_{in} , W_{fb} , W_{out} and C are respectively the recurrent, the input, the feedback, the output and the conceptor weight matrices and ξ is a uniform white noise term added to reservoir units.

How to load and retrieve one pattern ?

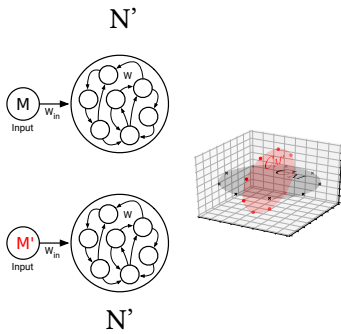


If N **mimic** N' (Load) then M can be retrieved from N as it would be from N' (Retrieve)

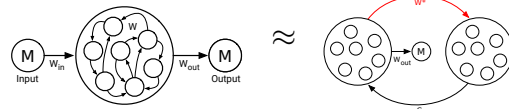
How to load and retrieve several patterns ?



If N **mimic** N' when N' is receiving either M or M' then M can't be retrieved in the same way from N (Load) (~~Retrieve~~)



then N can **mimic** N' receiving M (or M') by using an additional "projection" C_M (or $C_{M'}$)



But if M and M' make evolve N' in different spaces and M (or M') can be retrieved the same way from N (Retrieve)

Figure 1. How to use conceptors to handle long-term memories of temporal patterns? Let's assume N 's are recurrent neural networks and M 's are inputs or outputs. If N mimic N' then M can be retrieved from N as it would be from N' . If N mimic N' when N' is receiving either M or M' then M can't be retrieved the same way from N . But if M and M' make evolve N' in different spaces then N can mimic N' receiving M (or M') by using an additional "projection" C_M (or $C_{M'}$) and M (or M') can be retrieved the same way from N .

Controlling ESN dynamics using a conceptor

Following [10] notations, the equation for a conceptor C enforcing some particular dynamics can be written as:

$$x[n] = C \tanh (W x[n-1] + b)$$

where C is the conceptor (possibly changing over time), $x[n]$ is the state of the model at time n , W is the recurrent matrix and b is a constant bias. This can be extended to the general case where we also have an input $u[n]$ (with input matrix W_{in}) (or similarly a feedback), and writes:

$$x[n] = C \tanh (W x[n-1] + W_{in} u[n] + W_f b)$$

$$y[n] = W$$

Using a conceceptor C is similar to a change of W in $\tilde{W} = WC$ (and W_{out} in $W_{\text{out}}C$ if there is feedback). In our implementation, we thus consider:

$$x[n] = \tanh(WCx[n-1] + W_{in}u[n]) \quad (2)$$

Computing conceptors

In order to compute a conceceptor for some given dynamics, it is necessary to collect all the states of the reservoir and to concatenate them in a matrix X . The conceceptor C is then defined as:

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

where $R = XX^T$ is similar to a covariance matrix, and α (a.k.a the aperture) controls how close from the identity matrix C is.

Aperture adaptation

Intuitively, the aperture of a conceceptor controls the precision of the internal states representation. However, no information on internal states is lost, because it is possible to change the aperture of a conceceptor C without the need to recompute the conceceptor from scratch. To change the aperture, one only need to adapt the conceceptor C as follows:

$$\phi(C, \gamma) = C (C + \gamma^{-2}(I - C))^{-1} \quad (3)$$

where $\phi(C, \gamma)$ represents the same states than C with a different aperture, and γ is controlling how the aperture is modified. Intuitively, $\phi(C, \gamma)$ modifies the aperture of C by a factor of γ .

Linear combination

Given two conceptors C_1 and C_2 and $\lambda \in \mathbb{R}$, the linear combination of conceceptor C_1 and C_2 is defined as:

$$C = \lambda C_1 + (1 - \lambda)C_2$$

In the following when $\lambda \in [0, 1]$ we will talk about interpolation, when $\lambda > 1$ about right-extrapolation, and when $\lambda < 0$ about left-extrapolation.

Boolean operations

Boolean operations can be written as:

$$\begin{aligned} C \vee B &= \left(I + (C(I - C)^{-1} + B(I - B)^{-1})^{-1} \right)^{-1} \\ C \wedge B &= (C^{-1} + B^{-1} - I)^{-1} \\ \neg C &= I - C \end{aligned}$$

However, as highlighted in [21], \vee and \wedge are not idempotent (i.e. $C \vee C \neq C$ and $C \wedge C \neq C$). More precisely if C (resp. B) is a conceceptor built with the covariance matrix R (resp. Q), Jaeger proposes to build $C \vee B$ using the covariance matrix $R + Q$ that is by design not idempotent. What we propose

here is to consider instead the matrix $\beta R + (1 - \beta)Q$ with $\beta \in [0, 1]$ instead of $R + Q$, or if we want it to be symmetric $(R + Q)/2$. Similar calculation gives the following new \vee_β and \wedge_β .

$$C \vee_\beta B = \left(I + (\beta C(I - C)^{-1} + (1 - \beta)B(I - B)^{-1})^{-1} \right)^{-1}$$

$$C \wedge_\beta B = (\beta C^{-1} + (1 - \beta)B^{-1})^{-1}$$

This way of building the OR operation also has a data driven intuition. If we note $\beta = \frac{n}{n+p}$ where n (resp. p) is the number of data points used to build R (resp. Q) then $\beta R + (1 - \beta)Q$ is the "correlation matrix" obtained by taking the union of all the data points. Moreover, if we choose $\beta = 0.5$ then there is a direct link between the two way of defining the OR: $C \vee B = \phi(C \vee_{0.5} B, 2)$. In this study, the aperture was mostly not influencing the results, thus we show only the results for \vee .

2.3 Tasks

We consider the gating task described in [24] and two additional variants. In this task the model receives an input V that is continuously varying over time and another input being either 0 or 1 (trigger or gate T). To complete the task, the output has to be updated to the value of the input when the trigger is active and to remain constant otherwise (similarly to a line attractor). In other words, the trigger acts as a gate that controls the entry of the value in the memory (the output). Figure 2 describes this task and the two variants we consider in this work. In both variants we consider 11 values uniformly spread between -1 and 1: these values are used to discretize the input value (V) (when a trigger occurs; $T=1$) stored in the associated output (M). In the first variant (C2D task) we discretize only the associated output whereas in the second variant (D2D task) both are discretized. On a concrete example, if the model was trained to maintain 0.41 ($M = 0.41$), that means it was receiving a trigger ($T = 1$) along with the value 0.41 ($V = 0.41$). In the first approach we change both V and M to 0.4, whereas in the second approach we change only M to 0.4.

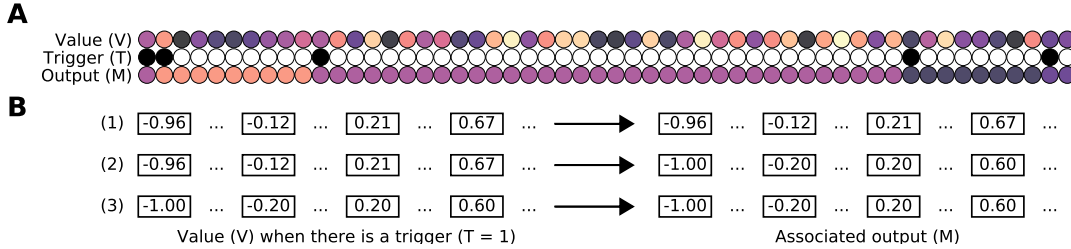


Figure 2. Gating tasks and its two variants. **A.** The gating task. Each column represents a time step (time increases from left to right), colored discs represent inputs (V and T) and the output (M). **B.** Variant tasks: **(1)** Original task: from continuous to continuous values. **(2)** First variant (C2D task): from Continuous to Discrete values. **(3)** Second variant (D2D task): from Discrete to Discrete values.

2.4 Implementation details

We consider a reservoir of 1000 neurons that has been trained to solve a gating task described in [24]. The overall dynamics of the network we consider are described by the following equations:

$$x[n] = C \tanh(W_{in}u[n] + W(x[n - 1] + \xi) + W_{fb}(y[n - 1]))$$

$$y[n] = W_{out}x[n] \tag{4}$$

where $u[n]$, $x[n]$ and $y[n]$ are respectively the input, the reservoir and the output at time n . W , W_{in} , W_{fb} , W_{out} and C are respectively the recurrent, the input, the feedback, the output and the conceceptor weight matrices and ξ is a uniform white noise term added to reservoir units. W , W_{in} , W_{fb} are uniformly sampled between -1 and 1 . Only W is modified to have sparsity level equal to 0.5 and a spectral radius of 0.1 . When W_{out} is computed to solve the gating task, the conceceptor C is considered to be fixed and equal to the identity matrix ($C = I$). In normal mode, the conceceptor C is equal to a conceceptor C_m that is generated and associated to a constant value m . In order to compute this conceceptor C_m , we impose a trigger ($T = 1$) as well as the input value ($V = m$) at the first time step, such that the reservoir has to maintain this value for 100 time steps. During these 100 time steps, we use the identity matrix in place of the conceceptor. The conceceptor C_m is then computed according to $C_m = XX^T (XX^T + \frac{I}{a})^{-1}$, where X corresponds to the concatenation of all the 100 reservoir states after the trigger, each row corresponding to a time step, I the identity matrix and a the aperture. In all the experiments the aperture has been fixed to $a = 10$. For the conceceptors pre-computed in Figure 3 and 6, the reservoir have been initialised with its last training state.

3 Results

Transfer between long-term and short-term memory

Figure 3 displays the two core ideas of our approach: **(1)** How to transfer short-term to long-term memory and **(2)** How to retrieve (in short-term memory) an information stored in long-term memory. **(1)** The long-term memory we consider is the conceceptor C_m associated to the value m maintained in short-term memory. To compute C_m we use the 100 first time steps after a trigger. Meanwhile no conceceptor is applied (i.e. $C = I$). After that we update C with C_m . On figure 3B we can see that it doesn't seem to cause any interference in the short-term memory. However, the memory currently lies both in the conceceptor C (long-term) and in the output unit y (short-term). **(2)** Now, the long-term memory we consider are only conceceptors C_m^D associated to discrete values between -1 and 1 (11 values uniformly spread between -1 and 1). Similarly as before, after a trigger we compute a new conceceptor C_m using the 100 first time steps after a trigger and without conceceptors ($C = I$). Then, we search for the closest conceceptor C_m^i among the conceceptors with discrete values C_m^D using a distance between conceceptors and we update C with this conceceptor. On figure 3C, we see the following behavior: after a trigger, the value is correctly updated in short-term memory and remains stable until C is updated (after 100 time steps) and then the output jumps to the closest discrete representation of the memory.

Extended maintenance (well beyond learning)

In Figure 4, we show how conceceptors allow to better stabilize the short-term memory, even in the presence of noise. Let us start by reminding that both the model and the conceceptors have been trained to maintain information in short-term memory only for few hundreds of time step, and that in the constant presence of a disturbing input (V). However, in the absence of noise ($\xi = 0$), we can clearly see that even without conceceptors the short-term memory can be maintained for several thousands of time steps. Nevertheless the ability to maintain information in short-term memory is not infinite: if we go further in time we can note that after approximately 100,000 time steps the short-term memory will slowly degrade (Figure 4A). The first thing noticeable is that, with conceceptors, this slow degradation vanishes (Figure 4B). Moreover, we tried the same analysis with noise inside the reservoir. Even 10^{-4}

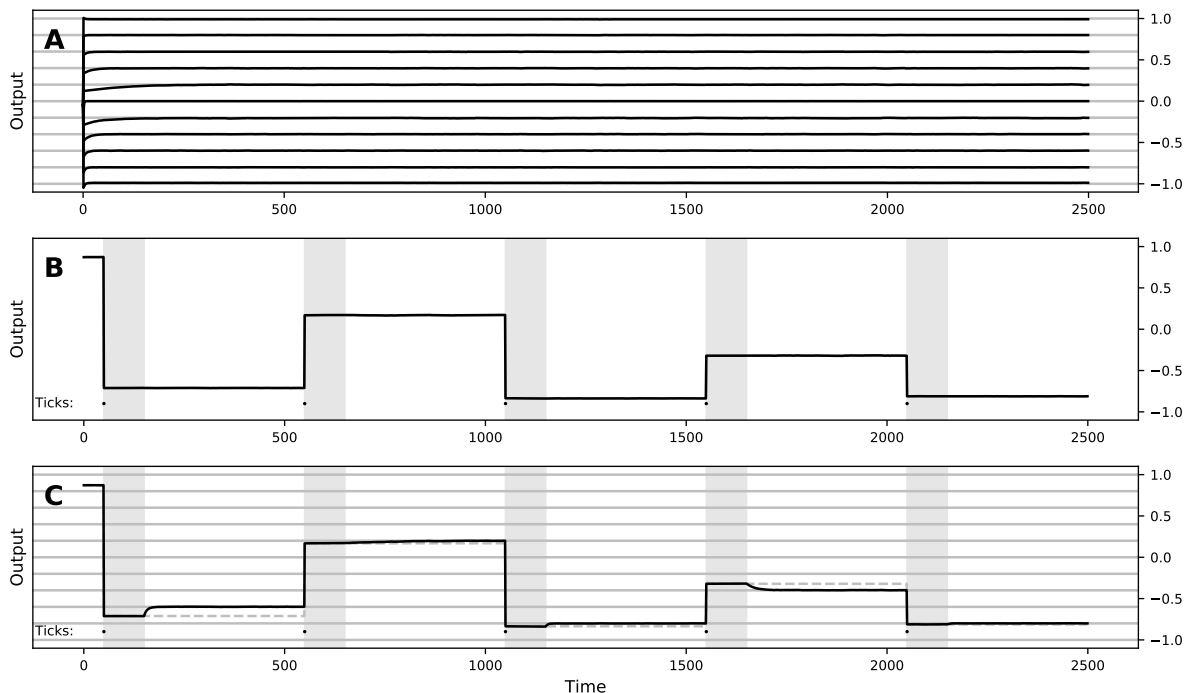


Figure 3. Approximation with conceptors or discrete conceptors. Black: Evolution of the short-term memory (i.e. the readout y). Gray lines: the discrete value considered. Light gray areas: time when conceptors are computed for the current value stored in short-term memory. **A.** Discrete conceptors are applied from a random state. **B-C.** Conceptor C_m are computed using 100 time steps after a trigger while $C = I$. **B.** C_m is directly applied to the following time steps. **C.** The closest conceptor among the discretized conceptor is applied for the following time steps. Dashed lines represents the memory that should have been kept if not discretized.

noise ($std(\xi) = 10^{-4}$) prevents the model to maintain longer than it has been trained to (Figure 4C). Interestingly in the noisy case, the benefit of conceptors is even more visible since they allow to maintain information in memory as if there were no noise (Figure 4D).

Using constant-memory conceptor stabilizes faster and more accurately to discretized values.

Another benefit of augmenting the WM model with conceptors is that while keeping the ability to maintain possibly everything, it is possible to add priors on the values the model is more likely to maintain. In Figure 5, we show the outcome of two complementary approaches discretizing memory. As we had already noticed in [24], only by being trained to maintained few discrete memory the model seems to generalize to all real values between -1 and 1. Thus the first task (C2D task) does not seem to allow to discretize the memory (Figure 5A). Concerning the second task (D2D task), it seems able to discretize the value memorized, (offline training Figure 5B, online training Figure 5C). But, in case it converges, the convergence towards a discrete value is way slower than with conceptors (Figure 5D).

Aperture adaptation of a constant-memory conceptors

The first operation we considered on constant-memory conceptors was the aperture adaptation. We did not notice an apparent influence of the aperture on the ability to maintain the constant values as

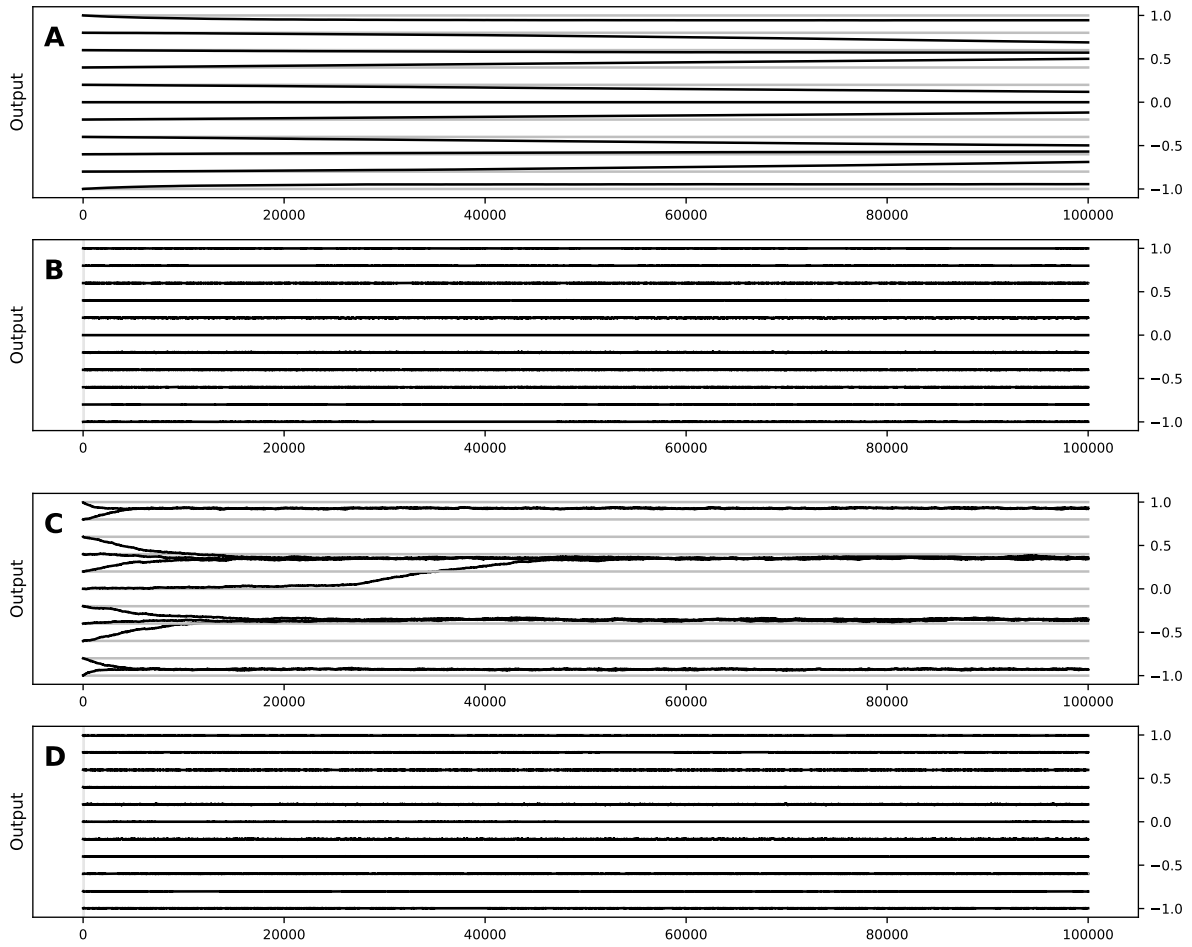


Figure 4. Stability comparison with or without conceptr, with or without noise. Black: Evolution of the short-term memory (i.e. the readout y). Gray lines: the discrete value considered. Light gray areas: time when conceptrs are computed for the current value stored in short-term memory. **A-B** No noise. **C-D** 0.0001 noise. **A** and **C** No conceptr used. **B** and **D** Discrete conceptr are applied.

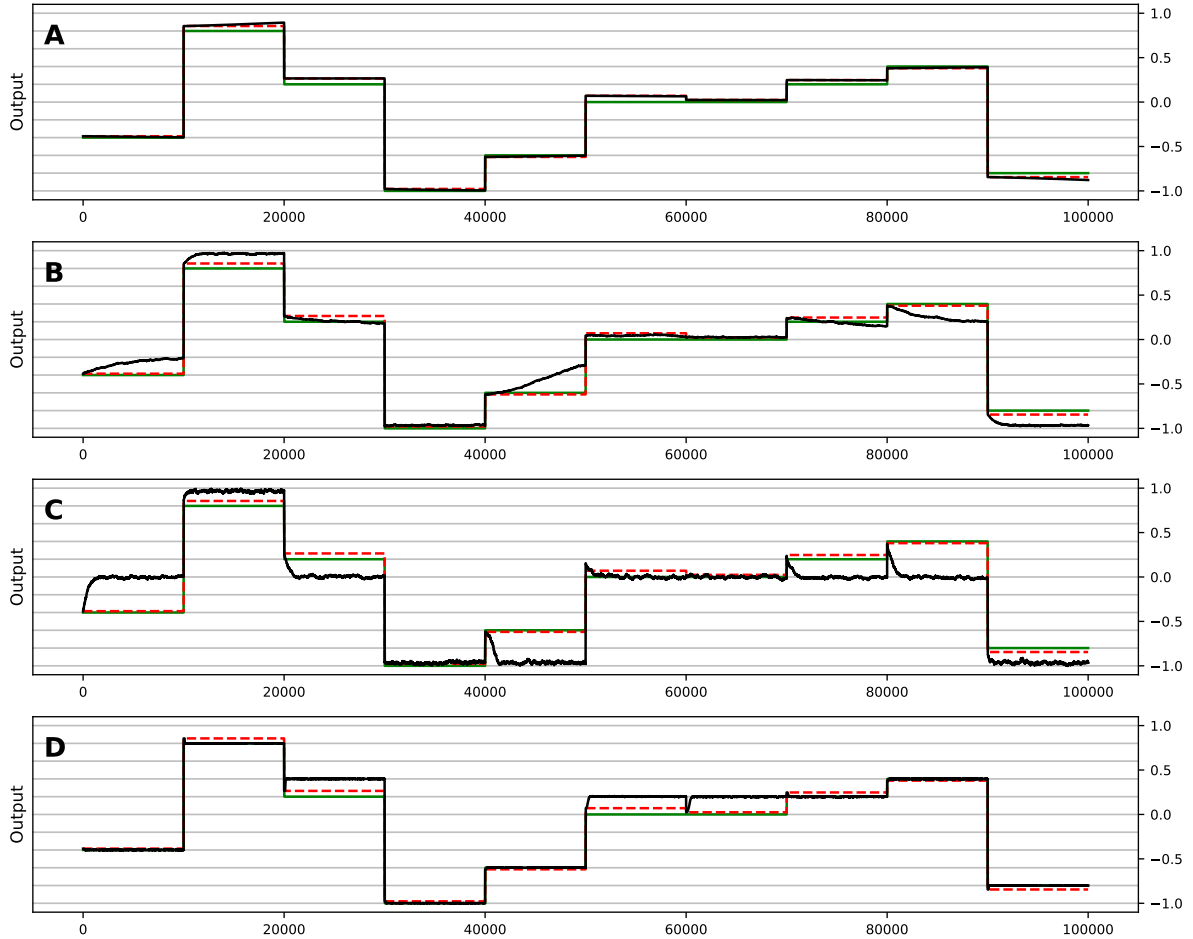


Figure 5. Different training procedures to discretize the output. **Black lines:** Evolution of the short-term memory (i.e. the readout y). **Gray lines:** the discrete value considered. **Green lines** represent the memory that should be kept when discretized. **Red dashed lines** represent the memory that should have been kept if not discretized. **A D2D task:** Only discrete values have been trained to be maintained. **B-C C2D task:** The value maintained is a discretized version of the value coming along with the trigger and no more the value itself. **B** Offline learning. **C** Online learning (ridge = 0.0001). **D Constant-memory conceceptor for discretized values:** Conceptors are computed using the 100 time steps after a trigger while $C = I$ and then the closest conceceptor among the discretized conceceptor is applied for the following time steps. The period of time during which conceptors are computed cannot be shown because they are too short compared to the figure's timescale.

long as it is not too small.

Linear interpolation of two constant-memory conceptors

In Figure 6, we show two main ideas: **(1)** how a linear interpolation between two conceptors can allow to generalize the gating of other values, and **(2)** a representation of the space in which lies the conceptors and their link to the memory they encode. **(1)** Interpolation and extrapolation C of conceptor $C_{0.1}$ and conceptor $C_{1.0}$ has been computed as $C = \lambda C_{1.0} + (1 - \lambda)C_{0.1}$ with 31 λ values uniformly spread between -1 and 2. Even though the interpolated ($\lambda \in [0, 1]$) conceptors obtained are not exactly equivalent to C_m conceptors obtained in Figure 3, they seem to also correspond to a retrieved long-term memory value to be maintained. The mapping between λ and the value is non-linearly encoded. For right-extrapolation ($\lambda \in [1, 2]$) the conceptor seems to be linked to a noisy version of a C_m conceptor. A value seems still to be retrieved from long-term memory and maintained in short-term memory: the output activity is not constant, but its moving average is constant. For left-extrapolation ($\lambda \in [-1, 0]$), the conceptor obtained does not seem to encode any information anymore: all the output activities collapse to zero. **(2)** Principal Component Analysis (PCA) have been performed using 201 pre-computed conceptors associated to values uniformly spread between -1 and 1. The first three components already explain approximately 85% of the variance. The first component seems to non-linearly encode the absolute value of the memory (Figure 6B) whereas the second component seems to non-linearly encode the memory itself (Figure 6C). The straight line conceptor C_m (Figure 6E-G), that might be an explanation why extrapolation does not work as we expected.

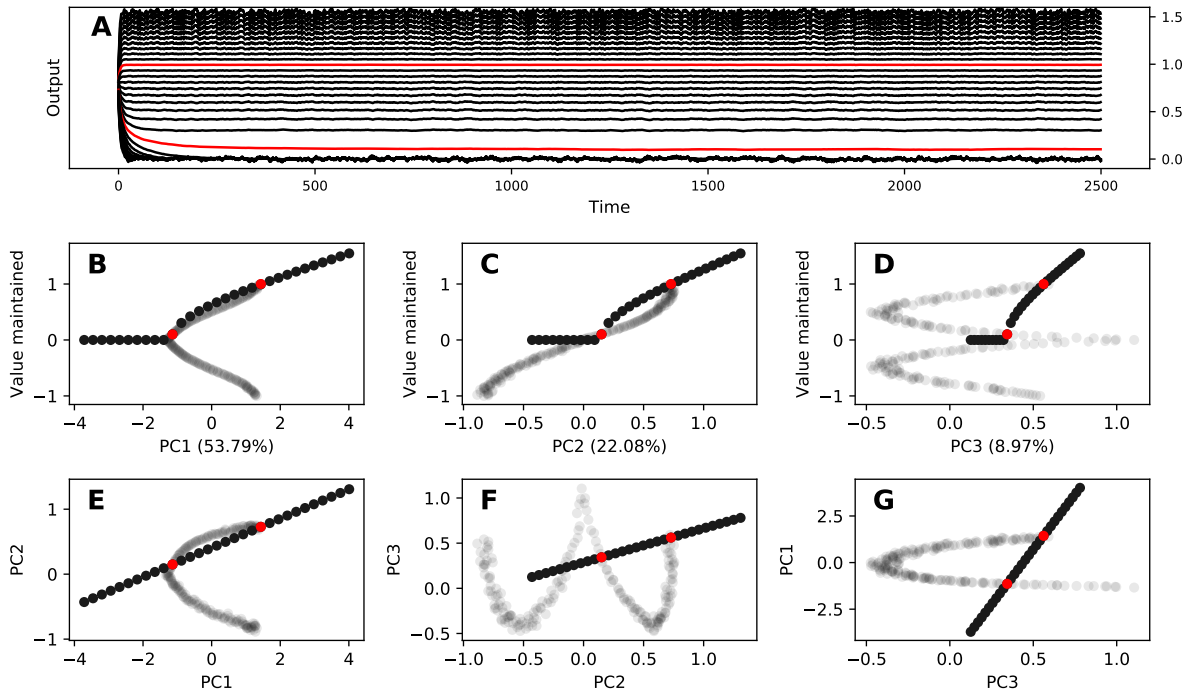


Figure 6. Generalisation of constant-memory conceptors C_m . **Red:** two constant-memory conceptors: $C_{0.1}$ and $C_{1.0}$. **Black:** Inferred conceptors, i.e. linear interpolation and extrapolation between $C_{0.1}$ and $C_{1.0}$. **A** Evolution of the short-term memory against time for different conceptors. **B-G Gray:** constant-memory conceptors C_m for 201 values of m uniformly spread between -1 and 1 . **B-D** Link between principal components of the conceptors and the memory they are encoding. For the interpolated conceptors, the memory is considered as the mean in the last 1000 time steps. **E-G** Representation of the conceptors in the three principal components of the C_m conceptors.

Intersection and union of constant-memory conceptors

We studied both how the conjunction and the disjunction of constant-memory conceptors were influencing the dynamics. In both cases, when a trigger occurs the output jumps towards the value to be maintained and then relaxes to another value. In the conjunction case, the value towards which it relaxes is easy to describe, it is always almost zero. In the disjunction case, it is harder to describe. In Figure 7, we show the values towards which the output relaxes (i.e. relaxation values) when the disjunction of two constant-memory conceptors is applied. First, as the disjunction of twice the same conceptor is either the same conceptor or an aperture adaptation of it (i.e. $C_m \vee C_m = \phi(C_m, 2)$ and $C_m \vee_{\beta} C_m = C_m$), the value towards which it relaxes is the value of the conceptor itself. Then, we realized that we could predict what would be the relaxation values in different cases: in general the relaxation value was mostly either almost zero or the maximum of the absolute values of the two conceptors multiplied by the sign of the new value to maintain. We propose the following formula to predict the value towards which it relaxes:

$$v_{\text{lim}}(c_1, c_2, v) = \begin{cases} c_1 & \text{if } c_1 = c_2 \\ \text{sign}(v) \times \max(|c_1|, |c_2|) & \text{if } \min(|c_1|, |c_2|) < |v| \text{ or } c_1 = -c_2 \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

where v is the initial value (V) proposed along with the trigger, c_1 (resp. c_2) is the constant associated to conceptor C_1 (resp. C_2), $v_{\text{lim}}(c_1, c_2, v)$ is the ultimate value reached while applying conceptor $C_1 \vee C_2$.

The predictions made by the formula are less accurate for extreme values such as for $v = 1.00$ (see Figure 7). We hypothesize a similar formula for relaxation values of n constant-memory conceptors:

$$v_{\text{lim}}(c_1, \dots, c_n, v) = \begin{cases} c_1 & \text{if } c_1 = c_2 = \dots = c_n \\ \text{sign}(v) \times \max(|c_1|, |c_2|, \dots, |c_n|) & \text{if } \min(|c_1|, |c_2|, \dots, |c_n|) < |v| \\ & \text{or } (\forall i, j |c_i| = |c_j| \\ & \text{and } \exists i, j \text{ such that } i > j \text{ and } c_i = -c_j) \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

where v is the initial value (V) proposed along with the trigger, c_i is the constant associated to conceptor C_i , $v_{\text{lim}}(c_1, c_2, \dots, c_n, v)$ is the ultimate value reached while applying conceptor $\bigvee_{i=1}^n C_i$.

4 Discussion

This study introduces the basis for establishing the link between long-term and short-term working memory in echo state networks using conceptors. This allowed us to show how a short-term working memory can be formed under the influence of long-term memory: i.e. how working memory can be biased towards predefined discrete values stored in long-term memory. In most working memory models (e.g. [12, 3]), it is implicitly assumed that the working memory is faithful to the value(s) to be maintained, that is, without any influence from the long-term memory. However, biological and psychological observations suggest that working memory is influenced by long-term memory. First, perceptions may be perturbed by noisy inputs or other processes, thus the associated working memory might end up being different from the ground truth. Second, our past experience may influence the information that will be maintained. This is exactly what we have shown using conceptors and an ad-hoc method for updating the working memory. Future work will concentrate on removing the (less

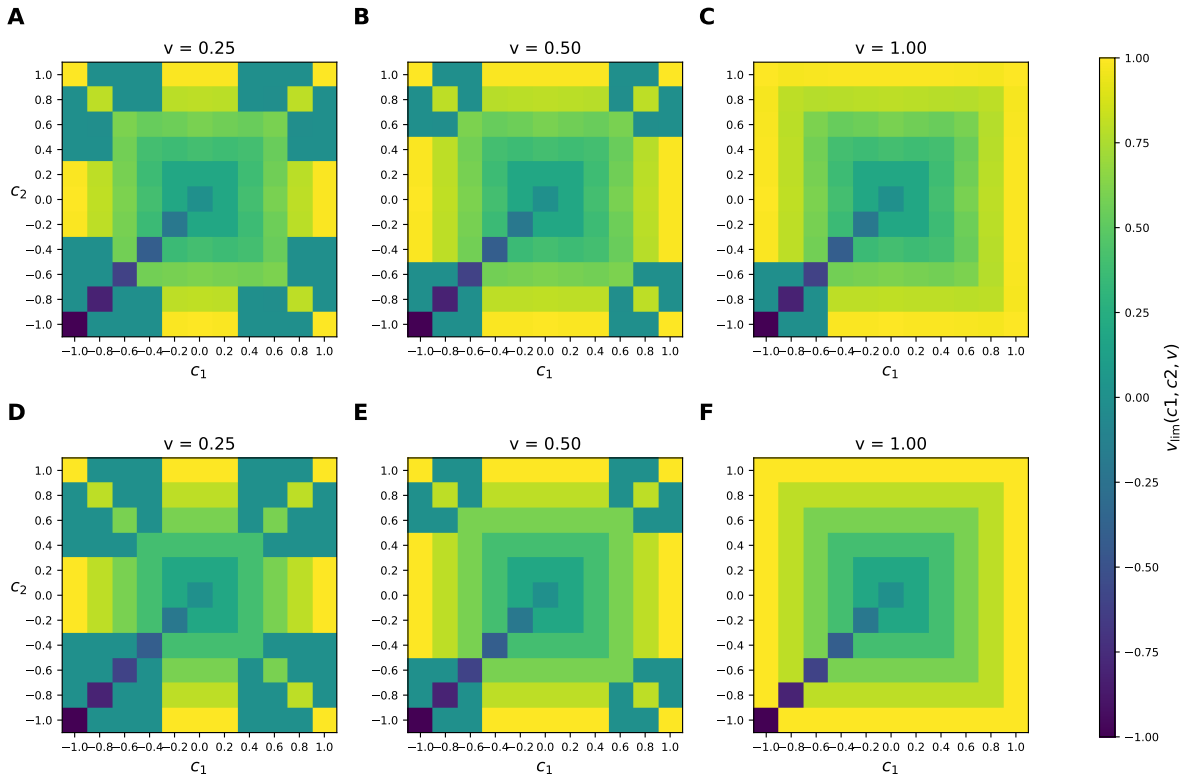


Figure 7. Relaxation values (i.e. values towards which the output relaxes) when applying the disjunction of two constant-memory concepts. In other words, it corresponds to the final values reached when applying the conceptor $C_1 \vee C_2$. **A-C** Empirical results from experiments. **D-F** Predictions based on equation 5.

biologically plausible) engineered steps, namely, the offline computation and selection of the closest conceptor. Such processes could be implemented using the auto-conceptors introduced in [10].

There are however theoretical difficulties when combining conceptors together: the result is difficult to predict because it largely differs from what we would naturally expect. For instance, a linear interpolation of constant-memory conceptors does not create another constant-memory conceptor. The reason being that the space of constant-memory conceptors is not a straight line. Hence, a mere linear combination of constant-memory conceptors could not lead to another constant-memory conceptor. Nevertheless, we have shown empirically that in all scenarios a linear combination of two constant-memory conceptors lead to a value that is maintained. However, this new memory is oscillating around the combination of the constant values (see Figure 6). This oscillation being a direct consequence of the perturbation of the system (i.e. the input). Moreover, the disjunction of conceptors is not implementing what we were expecting. For two conceptors with two constant values v_1 and v_2 such that $0 \leq v_1 \leq v_2$, we would expect that the disjunction encodes the two values simultaneously. More specifically, we expected such disjunction to implement a choice function between the two values stored in long-term memory. Instead, we obtained a conceptor that does not converge towards v_1 but only towards 0 or v_2 depending on the given input value. To some extent, v_1 and v_2 influence the disjunction with however different qualitative roles. In the general case of a disjunction of $n > 2$ constant-memory conceptors, only the extreme value seems to matter in the composite conceptor.

More interestingly, this work opens the door to another form of working memory: procedural (or functional) working memory. Instead of temporarily memorizing declarative information, this kind of working memory would be able to memorize procedural information (e.g. how a task should be

performed, which processes should be applied, etc.). For instance, imagine you are given some instructions which are to sum up a series of numbers. In order to complete this task, it is necessary to keep track of the current sum (e.g. in a classical short-term declarative working memory) that needs to be updated each time a new number is given. However, it is also necessary to remember the preliminary instruction (i.e summing up) in another form of working memory which is long-term (it needs to span the whole experiment) and which is procedural. This procedural nature makes this working memory quite peculiar because instead of memorizing a given information, it needs to memorize a procedure – here, a sequence of operations depending on the context – that needs to be applied each time an input is given. It is not yet clear how such memory could be encoded in the brain (e.g. sustained activity, dynamic activity, transient weights) and we think conceptors might be key in answering this question, but more experimental and theoretical work will be needed before answering this question.

Compliance with Ethical Standards

Authors declare they have no conflict of interest. Ethical approval: This article does not contain any studies with human participants or animals performed by any of the authors.

References

- [1] Bao J, Pei L, Ye M, Zhao X (2016) Action recognition based on conceptors of skeleton joint trajectories. *Rev Fac Ing* 31(4):11–22
- [2] Bartlett M, Garcia DH, Thill S, Belpaeme T (2019) Recognizing human internal states: A conceptor-based approach. [1909.04747](https://doi.org/10.1009.04747)
- [3] Bouchacourt F, Buschman TJ (2019) A flexible model of working memory. *Neuron* 103(1):147–160.e8, DOI [10.1016/j.neuron.2019.04.020](https://doi.org/10.1016/j.neuron.2019.04.020)
- [4] Brock A, Lim T, Ritchie JM, Weston N (2016) Neural photo editing with introspective adversarial networks. arXiv preprint [arXiv:160907093](https://arxiv.org/abs/160907093)
- [5] DePasquale B, Cueva CJ, Rajan K, Escola GS, Abbott LF (2018) full-FORCE: A target-based method for training recurrent networks. *PLOS ONE* 13(2):e0191527, DOI [10.1371/journal.pone.0191527](https://doi.org/10.1371/journal.pone.0191527), URL <https://doi.org/10.1371/journal.pone.0191527>
- [6] Gast R, Faion P, Standvoss K, Suckro A, Lewis B, Pipa G (2017) Encoding and decoding dynamic sensory signals with recurrent neural networks: An application of conceptors to birdsongs. *BioRxiv* DOI [10.1101/131052](https://doi.org/10.1101/131052), URL <https://doi.org/10.1101/131052>
- [7] He X, Jaeger H (2018) Overcoming catastrophic interference using conceptor-aided backpropagation. In: International Conference on Learning Representations, URL <https://openreview.net/forum?id=B1a17jg0b>
- [8] Jaeger H (2001) The "echo state" approach to analysing and training recurrent neural networks. Tech. Rep. 148, German National Research Center for Information Technology GMD, Bonn, Germany
- [9] Jaeger H (2004) Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science* 304(5667):78–80, DOI [10.1126/science.1091277](https://doi.org/10.1126/science.1091277), URL <https://doi.org/10.1126/science.1091277>

- [10] Jaeger H (2014) Controlling recurrent neural networks by conceptors. arXiv preprint arXiv:14033369
- [11] Jaeger H (2017) Using conceptors to manage neural long-term memories for temporal patterns. *Journal of Machine Learning Research* 18(13):1–43
- [12] Lim S, Goldman MS (2013) Balanced cortical microcircuitry for maintaining information in working memory. *Nature Neuroscience* 16(9):1306–1314, DOI 10.1038/nn.3492
- [13] Liu T, Ungar L, Sedoc J (2019) Continual learning for sentence representations using conceptors. CoRR abs/1904.09187, URL <http://arxiv.org/abs/1904.09187>, 1904.09187
- [14] Lukoševičius M, Jaeger H (2009) Reservoir computing approaches to recurrent neural network training. *Computer Science Review* 3(3):127–149, DOI 10.1016/j.cosrev.2009.03.005, URL <https://doi.org/10.1016/j.cosrev.2009.03.005>
- [15] Maass W, Natschläger T, Markram H (2002) Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation* 14(11):2531–2560, DOI 10.1162/089976602760407955, URL <https://doi.org/10.1162/089976602760407955>
- [16] Manohar SG, Zokaei N, Fallon SJ, Vogels TP, Husain M (2019) Neural mechanisms of attending to items in working memory. *Neuroscience & Biobehavioral Reviews* 101:1–12, DOI 10.1016/j.neubiorev.2019.03.017
- [17] Masse NY, Yang GR, Song HF, Wang XJ, Freedman DJ (2019) Circuit mechanisms for the maintenance and manipulation of information in working memory. *Nature Neuroscience* 22(7):1159–1167, DOI 10.1038/s41593-019-0414-3
- [18] Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: Proc. of NIPS, pp 3111–3119
- [19] Mongillo G, Barak O, Tsodyks M (2008) Synaptic theory of working memory. *Science* 319(5869):1543–1546, DOI 10.1126/science.1150769
- [20] Mossakowski T, Diaconescu R, Glauer M (2019) Towards logics for neural conceptors. *J of Applied Logics* 6(4):725–744
- [21] Mossakowski T, Diaconescu R, Glauer M (2019) Towards logics for neural conceptors. *J of Applied Logics* 6(4):725–744
- [22] Nachstedt T, Tetzlaff C (2017) Working memory requires a combination of transient and attractor-dominated dynamics to process unreliably timed inputs. *Scientific Reports* 7(1), DOI 10.1038/s41598-017-02471-z
- [23] Stokes MG (2015) ‘activity-silent’ working memory in prefrontal cortex: a dynamic coding framework. *Trends in Cognitive Sciences* 19(7):394–405, DOI 10.1016/j.tics.2015.05.004
- [24] Strock A, Hinaut X, Rougier NP (2020) A robust model of gated working memory. *Neural Computation* 32(1):153–181, DOI 10.1162/neco\._a\._01249
- [25] Tanaka G, Yamane T, Héroux JB, Nakane R, Kanazawa N, Takeda S, Numata H, Nakano D, Hirose A (2019) Recent advances in physical reservoir computing: A review. *Neural Networks* 115:100–123, DOI 10.1016/j.neunet.2019.03.005, URL <https://doi.org/10.1016/j.neunet.2019.03.005>